Systems Analysis and Design

Methodology

and

Supporting Processes

May 2009

Contents

1. APPROACH TO REQUIREMENTS VALIDATION, ANALYSIS AND DESIGN 11

1.1 INTRODUCTION 11
1.2 PROJECT ROLES AND RESPONSIBILITIES 12
1.2.1 Project Roles 12
1.2.2 Project Responsibilities 14
1.3 PROJECT TYPES 15
1.3.1 Feasibility/Assessment Project (FAP) 16
1.3.2 Small System Project (SSP) 17
1.3.3 Large System Project (LSP) 17
1.3.4 Purchased System Project (PSP) 17
1.3.5 Rapid Application Project (RAD) 18
1.3.6 Business Process Reengineering Project (BPR) 18
1.4 PROJECT PHASE TASKS FOR PROJECT TYPES 18
1.4.1 Project Initiation Phase 18
1.4.2 Analysis Phase 19
1.4.3 Design Phase 21
1.4.4 Construction Phase 24
1.4.5 Implementation Phase 25
1.4.6 Post Implementation Assessment Stage 26
1.4.7 System Modification Stage 26

2. PI - PROJECT INITIATION PHASE 27

2.1 INTRODUCTION 27
2.2 STEPS IN PROJECT INITIATION PHASE 28
2.2.1 PI1 - Set Initial Project Objectives and Scope 28
2.2.2 PI2 - Refine Project Scope 28
2.2.3 PI3 - Define Project Benefits 29
2.2.4 PI4 - Identify Sources of Business Knowledge 30
2.2.5 PI5 - Prepare Preliminary Project Timeline 30
2.2.6 PI6 - Determine Preliminary Project Costs/Resources 31
2.2.7 PI7 - Establish Business User Participation 31
2.2.8 PI8 - Identify Source of Project Funding/Resources 32
2.2.9 PI9 - Decide Whether to Continue With Project 32
2.2.10 PI10 - Prepare Project Plan 33
2.2.11 PI11 - Create Formal Project Plan Document 33
2.2.12 PI12 - Set Analysis Stage Standards 33
2.2.13 PI13 - Liaison With Control Function Groups 33
2.3 PROJECT INITIATION DELIVERABLES 35
2.3.1 PID1 - Information System Preliminary Requirements 37
2.3.2 PID2 - Project Scope Document 39
2.3.3 PID3 - Preliminary Project Plan 39
2.3.4 PID4 - Next Stage Project Plans 40
2.3.5 PID6 - Needs Analysis Report 41
2.3.6 PID7 - Decision As To Whether To Proceed With Project As Defined 42

3. EM - ENTERPRISE MODELLING PHASE 43

3.1 INTRODUCTION 43
3.2 STEPS IN ENTERPRISE MODELLING PHASE 43
3.2.1 EM1 - Plan the Enterprise Modelling Project 44
3.2.2 EM2 - Create Coarse Grain Enterprise Information Model 44
3.2.3 EM3 - Approve Coarse Grain Enterprise Information Model 45
3.2.4 EM4 - Refine Enterprise Information Model 45
3.2.5 EM5 - Approve Fine Grain Enterprise Information Model 46
3.2.6 EM6 - Inventory Existing Information Systems 46

3.2.7 EM7 - Develop Information Systems Architecture 47
3.2.8 EM8 - Develop Action Plan 47
3.2.9 EM9 - Approve Action Plan 47
3.2.10 EM10 - Identify Information Resource Management Principles 48
3.2.11 EM11 - Characterise Current Technology Environment 48
3.2.12 EM12 - Characterise Target Technology Environment 48
3.2.13 EM13 - Define Technology Migration Plan 48
3.2.14 EM14 - Manage the Introduction of New Technology 49
3.3 Enterprise Modelling Phase Deliverables 50
3.3.1 EMD1 - Enterprise Modelling Project Plan 52
3.3.2 EMD2 - Inventory of Enterprise Directions and Concerns 52
3.3.3 EMD3 - Inventory of Enterprise Business Locations and Organisation Units ... 52
3.3.4 EMD4 - Inventory of Business Functions 52
3.3.5 EMD5 - Inventory of Subject Areas 52
3.3.6 EMD6 - Association of Business Functions with Subject Areas, Enterprise Directions, Concerns, Business Locations and Organisation Units 52
3.3.7 EMD7 - Association of Subject Areas with Enterprise Directions, Concerns, Business Locations and Organisation Units 52
3.3.8 EMD8 - Scope of Fine Grain Information Model 53
3.3.9 EMD9 - Inventory of Business Processes 53
3.3.10 EMD10 - Inventory of Entity Types 53
3.3.11 EMD11 - Association of Business Processes with Enterprise Business Locations, Organisation Units, Entity Types, Enterprise Directions and Concerns 53
3.3.12 EMD12 - Association of Entity Types with Enterprise Directions, Concerns, Business Locations and Organisation Units 53
3.3.13 EMD13 - Inventory of Mechanisms and Data Collections 53
3.3.14 EMD14 - Assessments of Mechanisms and Data Collections 54
3.3.15 EMD15 - Associations of Existing Information Systems to Enterprise Activities, Data, Business Locations and Organisation Units 54
3.3.16 EMD16 - Information Systems Architecture 54
3.3.17 EMD17 - Data Architecture 54
3.3.18 EMD18 - Prioritised Business Areas 54
3.3.19 EMD19 - Strategic Enterprise Information Plan 54
3.3.20 EMD20 - Information Resource Management Principles 54
3.3.21 EMD21 - Characterisation of Current Technology Environment 55
3.3.22 EMD22 - Characterisation of Target Technology Environment 55
3.3.23 EMD23 - Technology Migration Plan 55
3.3.24 EMD24 - Inventory of Emerging Technologies 55

4. ANALYSIS PHASE 57

4.1 Introduction 57
4.2 Steps in Analysis Stage 60
4.2.1 A1 - Acquire Initial Understanding 60
4.2.2 A2 - Produce Entity Relationship Diagram 63
4.2.3 A3 - Produce Process Model 66
4.2.4 A4 - Analyse Involvement Matrices 70
4.2.5 A5 - Confirm Outline Business Model 72
4.2.6 A6 - Analyse Distribution 79
4.2.7 A7 - Define Design Areas 80
4.2.8 A8 - Consider Purchased Software Solution 86
4.2.9 A9 - Analyse Entity Type Life Cycles 94
4.2.10 A10 - Analyse Process Logic 97
4.2.11 A11 - Analyse Business Rules 102
4.2.12 A12 - Analyse Current Systems 104
4.2.13 A13 - Confirm Correctness of Detailed Business Model 112
4.2.14 A14 - Reaffirm Design Area Evaluation 119
4.3 Analysis Phase Deliverables 120
4.3.1 AD1 - Project Control 122

4.3.2 AD2 - Data 123
4.3.3 AD3 - Processes 125
4.3.4 AD4 - Process/Data Interaction 127
4.3.5 AD5 - Business Model 129
4.3.6 AD6 - Distribution and Frequency 129
4.3.7 AD7 - Design Preparation 130
4.3.8 AD8 - Current System Information 132
4.3.9 AD9 - BAA Report 134
4.3.10 AD10 - System Plans 135

5. DESIGN PHASE 137

5.1 INTRODUCTION 137
5.2 BUSINESS SYSTEM DESIGN 137
5.2.1 BD1 - Design System Structure 139
5.2.2 BD2 - Design for Implementation 143
5.2.3 BD3 - Design Preliminary Data Structures 146
5.2.4 BD4 - Define Data and Procedure Interactions 151
5.2.5 BD5 - Design Procedures 154
5.2.6 BD6 - Check Design Completeness 164
5.2.7 BD7 - Check Design Correctness 165
5.2.8 BD8 - Plan for Technical Design 170
5.3 TECHNICAL DESIGN 176
5.3.1 TD1 - Define Technical Context and Requirements 178
5.3.2 TD2 - Design the Data Structures 182
5.3.3 TD3 - Develop Program Specifications 185
5.3.4 TD4 - Prepare for Testing 190
5.3.5 TD5 - Assess Performance 201
5.3.6 TD6 - Design for System Testing 202
5.3.7 TD7 - Plan Construction and Implementation Stages 204
5.4 DESIGN PHASE DELIVERABLES 206
5.4.1 DD1 - Procedures 209
5.4.2 DD2 - Implementation Issues 211
5.4.3 DD3 - Interface Definition 212
5.4.4 DD4 - Training Requirements 212
5.4.5 DD5 - Data Structures 215
5.4.6 DD6 - Data Access Diagram 217
5.4.7 DD7 - Dialog Flows 217
5.4.8 DD8 - Layout Designs 217
5.4.9 DD9 - Prototype 218
5.4.10 DD10 - Procedure/Data Matrices 220
5.4.11 DD11 - Quality Reports 221
5.4.12 DD13 - Procedure/Benefit Matrix 221
5.4.13 DD13 - Cost-Benefit Data 221
5.4.14 DD14 - Technical Design Phase Plan 222
5.4.15 DD15 - Design Infrastructure 222
5.4.16 DD16 - Load Matrices 228
5.4.17 DD17 - Software Structure 228
5.4.18 DD18 - Data Storage 230
5.4.19 DD19 - Integration Group 231
5.4.20 DD20 - Test Plans 231
5.4.21 DD21 - Security 236
5.4.22 DD22 - System Performance 237
5.4.23 DD23 - Construction Plan 237

6. CONSTRUCTION PHASE 238

6.1 INTRODUCTION 238
6.2 STEPS IN CONSTRUCTION PHASE 240
6.2.1 C1 - Construct Computing Environment 240

6.2.2 C2 - Prepare Development Procedures and Conventions 241
6.2.3 C3 - Establish Database Environment 242
6.2.4 C4 - Generate Implementable Module 244
6.2.5 C5 - Generate Test Data and System Documents 245
6.2.6 C6 - Finalise Implementation Plan 248
6.2.7 C7 - Perform Integration Tests 251
6.2.8 C8 - Perform System Tests 252
6.2.9 C9 - Perform Benchmark Tests 253
6.2.10 C10 - Perform Acceptance Tests 253
6.3 Construction Phase Deliverables 255
6.3.1 CD1 - Technology Environment 257
6.3.2 CD2 - Development Environment 257
6.3.3 CD3 - Production Mode 257
6.3.4 CD4 - Testing 258
6.3.5 CD5 - Training Deliverables 258
6.3.6 CD6 - User Deliverables 258
6.3.7 CD7 - Implementation Deliverables 258

7. IMPLEMENTATION PHASE 259

7.1 Introduction 259
7.2 Steps in Implementation Phase 261
7.2.1 I1 - Train Users 261
7.2.2 I2 - Perform Data Conversion 262
7.2.3 I3 - Install Production System 264
7.2.4 I4 - Accept System Installation 265
7.2.5 I5 - Support the System 268
7.2.6 I6 - Respond to Emergency Situations 269
7.3 Implementation Phase Deliverables 269
7.3.1 ID1 - Training 270
7.3.2 ID2 - Data Conversion 270
7.3.3 ID3 - Production Environment 271
7.3.4 ID4 - User Acceptance Agreement 272
7.3.5 ID5 - Project Initiation Request 273

8. POST IMPLEMENTATION ASSESSMENT PHASE 274

8.1 Introduction 274
8.2 Steps in Post Implementation Assessment Phase 274
8.2.1 PIA1 - Define Project(s) to Evaluate 274
8.2.2 PIA2 - Gather Evaluated Project's Information 275
8.2.3 PIA3 - Evaluate Methods and Techniques Employed in the Evaluated Project(s) 275
8.2.4 PIA4 - Publish Post Implementation Assessment 276
8.3 Post Implementation Assessment Phase Deliverables 277
8.3.1 PIAD1 - Post Implementation Assessment 277

9. SYSTEM MODIFICATION PHASE 278

9.1 Introduction 278
9.2 Steps in System Modification Phase 278
9.2.1 SM1 - Evaluate System 278
9.2.2 SM2 - Assess Changes or Enhancement Requests 278
9.2.3 SM3 - Analyse the Nature of the Change 278
9.2.4 SM4 - Analyse the Impact of the Change 278
9.2.5 SM5 - Execute the Change 279

10. RAD REQUIREMENTS PLANNING PHASE 280

10.1 Introduction 280
10.2 Steps in RAD Requirements Planning Phase 283
10.2.1 RADRP1 - Research Current Situation 283

10.2.2 RADRP2 - Define Requirements 284
10.2.3 RADRP3 - Finalise Requirements 288

11. RAD USER DESIGN PHASE 293

11.1 INTRODUCTION 293
11.2 STEPS IN RAD USER DESIGN PHASE 296
11.2.1 RADUD1 - Produce Detailed System Area Model 296
11.2.2 RADUD2 - Develop Outline System Design 300
11.2.3 RADUD3 - Refine System Design 303
11.2.4 RADUD4 - Prepare Implementation Plan 304
11.2.5 RADUD5 - Finalise System Design 307
11.2.6 RADUD6 - Obtain Approval for Construction 309

12. RAD RAPID CONSTRUCTION PHASE 311

12.1 INTRODUCTION 311
12.2 STEPS IN RAD RAPID CONSTRUCTION PHASE 315
12.2.1 RADRC1 - Prepare for Rapid Construction 315
12.2.2 RADRC2 - Construct the System 316
12.2.3 RADRC3 - Generate Test Data and System Documents 319
12.2.4 RADRC4 - Prepare for Implementation 322
12.2.5 RADRC5 - Verify System Construction 326

13. APPENDIX 1 – TERMS 329

13.1 A 329
13.2 B 330
13.3 C 331
13.4 D 332
13.5 E 334
13.6 F 335
13.7 I 335
13.8 J 336
13.9 L 336
13.10 M 337
13.11 N 337
13.12 O 337
13.13 P 338
13.14 Q 341
13.15 R 341
13.16 S 342
13.17 T 343
13.18 U 345
13.19 W 345

14. APPENDIX 2 - ENTITY RELATIONSHIP MODELLING 346

14.1 INTRODUCTION 346
14.2 CONCEPTS 346
14.2.1 Activity 346
14.2.2 Attribute Type 347
14.2.3 Cardinality 348
14.2.4 Condition 349
14.2.5 Decomposition 351
14.2.6 Domain 352
14.2.7 Elementary Process 352
14.2.8 Entity 354
14.2.9 Entity Type 354
14.2.10 Entity Type Horizon 355
14.2.11 External Object 356
14.2.12 Function 357

14.2.13 Identifier 358
14.2.14 Location 359
14.2.15 Normalisation 360
14.2.16 Optionality 361
14.2.17 Pairing 361
14.2.18 Partitioning 362
14.2.19 Predicate 364
14.2.20 Procedure 365
14.2.21 Process 366
14.2.22 Relationship Type 367
14.2.23 Reusability 370
14.2.24 Subject Area 371
14.2.25 Subtype 372
14.3 Rules 373
14.3.1 Diagram Conventions for Entity Types 373
14.3.2 Diagram Conventions for Relationship Types 373
14.3.3 Entity Types 379
14.3.4 Rules About Relationship Types 380
14.3.5 Rules About Entity Analysis Objects 381
14.4 Entity Relationship Modelling Techniques 381
14.4.1 Entity Types 381
14.4.2 Naming Relationship Types 382
14.4.3 Reading a Relationship Type 382
14.4.4 Redundant Relationships 384
14.4.5 Types of Relationship Types 385
14.4.6 Modelling Organisational Structure 385
14.4.7 Refinements to the Entity Relationship Model 386
14.4.8 Drawing Entity Relationship Diagrams 387
14.4.9 Presenting an Entity Relationship Diagram 389

15. APPENDIX 3 - SWOT ANALYSIS 390

15.1 Introduction 390
15.2 SWOT Template 390

16. APPENDIX 4 – CHANGE, ISSUE AND PROBLEM MANAGEMENT 393

16.1 Introduction 393
16.2 Change Management Process 394
16.3 Change and Problem Requests 398
16.4 Evaluating Change Requests 400
16.5 Analysing Change Requests 401
16.5.1 Project Manger Responsibilities 402
16.5.2 Consolidating CR/PR into Change Package(s) 402
16.6 Change Requests/Problem Reports and Change/Problem Log 403
16.6.1 Change Request/Problem Report 404
16.6.2 Change/Problem Log 406
16.6.3 CR/PR Progress and Status Reports 407
16.7 Change Control Process Components 408
16.7.1 Change Control Board 408
16.7.2 Change Control Office 409

17. APPENDIX 5 – RISK MANAGEMENT 410

17.1 Introduction 410
17.2 Risk Management Activities 410
17.2.1 Identify Risks 411
17.2.2 Analyse Risks 416
17.2.3 Prioritise Risks 417
17.2.4 Produce Risk Mitigation Plan 417
17.2.5 Mitigate Risks 421

17.2.6 Assess Mitigation Effectiveness 421
17.2.7 Reassess Exposure 422

18. APPENDIX 6 – SAMPLE TEST PLAN 424

18.1 Revision History 424
18.2 Introduction 424
18.3 Goal of Project and Feature Team 424
18.4 Primary Testing Concerns 424
18.5 Primary Testing Focus 424
18.6 References 424
18.7 Personnel 425
18.8 Testing Schedule 425
18.9 Feature History 425
18.10 Features 425
18.11 Files and Modules 426
18.11.1 Files List 426
18.11.2 Registry, INI Settings 426
18.11.3 Setup Procedures 426
18.11.4 De-installation Procedures 426
18.11.5 Database Setup and Procedures 426
18.11.6 Network Domain/Topologies Configuration Procedures 426
18.11.7 Performance Monitoring Counters Setup And Configurations 426
18.12 Operational Issues 426
18.12.1 Backup 427
18.12.2 Recovery 427
18.12.3 Archiving 427
18.12.4 Monitoring 427
18.12.5 Operational Problem Escalation/Alert Methods 427
18.13 Scope of Test Cases 427
18.14 Acceptance Criteria 427
18.15 Key Feature Issues 427
18.16 Test Approach 427
18.16.1 Design Validation 427
18.16.2 Data Validation 428
18.16.3 API Testing 428
18.16.4 Content Testing 428
18.16.5 Low-Resource Testing 428
18.16.6 Setup Testing 428
18.16.7 Modes and Runtime Options 428
18.16.8 Interoperability 429
18.16.9 Integration Testing 429
18.16.10 Compatibility: Clients 429
18.16.11 Compatibility: Servers 429
18.16.12 Beta Testing 429
18.16.13 Environment/System – General 429
18.16.14 Configuration 430
18.16.15 User Interface 430
18.16.16 Performance & Capacity Testing 430
18.16.17 Scalability 430
18.16.18 Stress Testing 430
18.16.19 Volume Testing 431
18.16.20 International Issues 431
18.16.21 Robustness 431
18.16.22 Error Testing 431
18.16.23 Usability 431
18.16.24 Accessibility 431
18.16.25 User Scenarios 432
18.16.26 Boundaries and Limits 432

18.16.27 Operational Issues 432
18.16.28 Special Code Profiling and Other Metrics..... 433
18.17 Test Environment 433
18.17.1 Operating Systems..... 433
18.17.2 Networks 434
18.17.3 Hardware..... 434
18.17.4 Software..... 434
18.18 Unique Testing Concerns For Specific Features 434
18.19 Area Breakdown..... 434
18.19.1 Feature Name..... 435
18.20 Test Case Structure 437
18.21 Spec Review Issues 437
18.22 Test Tools 438
18.23 Smoke Test (acceptance test, build verification, etc.)..... 438
18.24 Automated Tests 438
18.25 Manual Tests..... 438
18.26 Regression Tests 438
18.27 Bug Bashes..... 438
18.28 Bug Reporting 438
18.29 Plan Contingencies..... 439
18.30 External Dependencies 439
18.31 Headcount Requirements 439
18.32 Product Support 439
18.33 Testing Schedule 439
18.34 Drop Procedures..... 440
18.35 Release Procedures 440
18.36 Alias/Newsgroups and Communication Channels 440
18.37 Regular Meetings..... 440
18.38 Decisions Making Procedures 440

19. APPENDIX 7 – SAMPLE TEST CASE 441

19.1 Introduction..... 441
19.1.1 Definitions, Acronyms and Abbreviations 441
19.1.2 References..... 441
19.2 Testing Environments 442
19.2.1 Environment 1..... 442
19.2.2 Environment 2..... 442
19.3 Setup Information (General Pre-conditions) 443
19.4 Test Cases 444
19.4.1 Test Case 1: <Test Case Name> 444
19.4.2 Array of values..... 445
19.5 Test Case 2: <Test Case Name>..... 445

20. APPENDIX 8 – SAMPLE QUALITY ASSURANCE PLAN..... 446

20.1 Introduction..... 446
20.1.1 Purpose..... 446
20.1.2 Scope..... 446
20.1.3 Definitions, Acronyms and Abbreviations 446
20.1.4 References..... 446
20.1.5 Overview 447
20.2 Quality Objectives 447
20.3 Management 447
20.3.1 Organisation 447
20.3.2 Tasks and Responsibilities..... 447
20.4 Documentation 447
20.5 Standards and Guidelines 448
20.6 Metrics 448
20.7 Review and Audit Plan 448

20.8 Evaluation and Test 450
20.9 Problem Resolution and Corrective Action 450
20.10 Tools, Techniques, and Methodologies 450
20.11 Configuration Management 450
20.12 Supplier and Subcontractor Controls 450
20.13 Quality Records 450
20.14 Training 450
20.15 Risk Management 451

21. APPENDIX 9 – FUNCTION POINT ANALYSIS 452

21.1 Introduction 452
21.2 Components of Function Points 452
21.2.1 Internal Logical Files 453
21.2.2 External Interface Files 453
21.2.3 External Input 454
21.2.4 External Output 454
21.2.5 External Inquiries 455
21.3 Function Point Analysis 456
21.3.1 Calculating Unadjusted Function Point Count 456
21.3.2 Adjusting UFP for Complexity 456
21.3.3 Translating FPA Count Into Project Estimates 458
21.4 Other Users of FPA 459
21.4.1 Managing Project Scope 459
21.4.2 Estimating Support Requirements 460
21.4.3 Using FPA to Make Application Retain, Retire, Replace and Design Decisions 460
21.5 Other Factors Affecting Estimation 461

1. Approach to Requirements Validation, Analysis and Design

1.1 Introduction

For any systems implementation project the requirements validation, analysis and design phases are extremely important.

The information contained in this document is a generic methodology. This needs to be modified to meet the needs of any project during the project mobilisation phase. Some of the steps can be reduced or eliminated to accelerate the work because the work has already been done or the steps are redundant.

The advantages of using a structured and detailed analysis and design methodology such as this are:

- The analysis leads to a well-defined set of documented requirements and underlying processes.
- The analysis and design process generates understanding and insight.
- The comprehensive nature of the methodology will ensure that all requirements and issues are identified and addressed.
- The detail obtained and documented will ensure that all parties know what is being delivered.
- The analysis and design information will feed into the integration, system and user testing phases.
- It reduces the risk of project failure.
- It reduces the risk of system rejection based on user requirements having being met incompletely.

Any reasonably complex project such as the needs the foundation created by a detailed analysis and design phase.

The next sections describe in detail the analysis and design phases and their deliverables.

1.2 Project Roles and Responsibilities

1.2.1 Project Roles

The methodology contains references to the following logical roles. These roles can be combined and performed by a single person, if necessary and appropriate.

Role Title	Description
A	**Auditor** - An organisation unit or individual that performs a control function for the organisation.
BCA	**Business Contract Administrator** - A party or organisation unit that provides legal consultation, manages and/or enforces a user contractual obligation.
BE	**Business Expert** - An individual with in-depth knowledge of an organisation unit's overall business processes and data and who is able to identify the business needs of a particular organisational unit.
BPM	**Business Project Manager** - A person representing the business who has decision-making authority for the user community. They are responsible for assignment of users to the project team and to develop user guides, office procedures and user training. Further, they identify users to be interviewed, users to participate in review walkthroughs and inspections. They also will identify users who will participate in testing, development of test scripts, test cases and test data. Additionally, they will resolve user conflicts, be a focal point for change control, prioritise user requests and keep the user community informed of project status.
CCM	**Code Configuration Manager** - A person or organisation unit responsible for maintaining the library of code "objects" for a system and migrating/installing/configuring new and/or revised versions to appropriate environments.
DAD	**Data Administrator** - An organisation unit or individual that develops and maintains an unambiguous and consistent set of definitions of data, activities and their relationships to facilitate common understanding and to provide a consistent framework for information systems within the enterprise.
DBA	**Database Administrator** - The organisation unit or individual responsible for designing and maintaining the physical database(s) required by an information technology project and develops database rollback and recovery strategies.
DS	**Data Steward** - An organisation unit or individual that is responsible for the accuracy, timeliness, access and security of a segment of the organisation's business data.
DC	**Documentation Specialist** - A person who writes, reviews, maintains and releases all product documentation; including user guides, reference manuals, detailed system specifications; and

	designs and defines documentation standards.
EU	**End User** - An organisation unit or individual that supplies and/or utilises business information.
IRP	**Information Resources Planner** - An organisation unit or individual that plans the information systems technology environment and recommends the business areas and information systems needed by the organisation.
ITS	**Information Technology Steering Committee** - A decision making body of individuals accountable for the selection and prioritisation of the organisation's information technology projects.
MM	**Model Manager** - A person or organisation responsible for managing the organisation's library of models.
O	**Operations** - An organisation unit responsible for operating an information system, communications network and for hardware and communications facilities maintenance.
PD	**Project Director** - A person who resolves issues between the Business Project Manager and Technical Project Manager. They have decision authority for budget issues and resource issues beyond the scope of the project managers. Further, they act as a liaison between the project and the rest of the organisation community and resolve political issues beyond the scope of the project managers. The role of mediation is assumed in each task where it is not specified.
QA	**Quality Assurance/Quality Control** - A function responsible for defining standards, guidelines and maintaining a glossary of terms to facilitate common understanding among anyone who has to interact with the project. Additionally, Quality Assurance ensures all deliverables meet or exceed standards set for the deliverable. Standards are usually based on the intended use, target audience and impact on the community of the deliverable.
SAC	**Security Administrator** - The organisation unit(s) accountable for implementing the security and protection of the information and technology resources of the organisation.
S	**Sponsor** - An individual or group representing the organisation's administrative management that endorses the scope and goals of a project and represents that project in management councils. The sponsor has primary responsibility for marshalling the resources needed to fund the project.
SC	**Steering Committee** - Members of the academic and/or administrative staff having collective accountability for overseeing the welfare of an information technology project.
SA	**System Architect** - An individual responsible for the overall design of the system and for the successful integration of sub-systems. The System Architect provides technical leadership, plans and estimates, technical interfaces external to project; and manages overall test strategies. This individual also resolves technical issue/conflicts, and ensure adherence to standards.
SD	**System Developer** - The individual(s) responsible for the modelling analysis, programming, testing and/or implementation

	of an information system. (AKA: Modeller, Developer, Application Developer, or Programmer/Analyst)
SAD	**Systems Administrator** - An organisation unit or individual that provides and maintains that hardware and software that establishes and maintains the environment in which the application software operates.
TPM	**Technical Project Manager** - An individual having primary responsibility for the welfare of an information system project.
TSC	**Technology Support Contact** - A user department's representative who has been given responsibility as the first point of contact for computing related problems or questions. This person has the responsibility to communicate all unsolved problems and unanswered questions to appropriate personnel for resolution. An individual who installs and maintains all software, hardware and the networks for end-clients.
TS	**Training Specialist** - An organisation unit or individual that prepares and/or teaches a full curriculum of technical courses to the user community. They manage all aspects of courseware development process and work with management to insure proper course rollout and translation/localisation of course material.

1.2.2 Project Responsibilities

The methodology contains references to the following responsibilities.

Responsibility	**Description**
P = Primary Responsibility	The role of the party that has primary responsibility for consolidating all information and producing and publishing the deliverable. It includes coordination of all activities associated with the development and approval of the deliverable. Verification of the fulfilment of a deliverable component or of a decision point may be either verbal or in writing. It is the party having primary responsibility for the deliverable component to determine which mode is appropriate.
S = Support Responsibility	The role of the party that must be involved in the creation of the deliverable to ensure that its requirements and responsibilities are met. It may involve supplying significant input or providing a control function.
I = Input Responsibility	The role of the party that must provide necessary information (either written or verbal) to complete the deliverable.
C = Consultation Responsibility	The role of the party that may be required to provide assistance to complete the deliverable.
A = Approval	The role of the party who must formally approve the

Responsibility	contents of the deliverable before the deliverable can become baselined and the development process can proceed. Accountability for obtaining the approval of the designated party rests with the party having primary responsibility for that deliverable. Verification of the fulfilment of a deliverable component or of a decision point may be either verbal or in writing. It is the party having primary responsibility for the deliverable component to determine which mode is appropriate. NOTE: Once approved, any further changes require the use of the change management process. Issues not resolved are to be escalated through the designated approval party's management chain. Unresolved issues arising from a deliverable or decision point will stop the development process.
E = Endorse Responsibility	The role wherein a party must formally endorse or concur with the approval of the deliverable in order for the deliverable to be baselined and the development process to proceed. Responsibility for gaining the concurrence of the designated endorsement rests with the party having primary responsibility for that deliverable. Verification of the fulfilment of a deliverable or of a decision point may be either verbal or in writing. It is the party having primary responsibility for the deliverable to determine which mode is appropriate.
K = Uses Task Output	The role of using the output (deliverable) of an ADM task. All the responsibilities related to a deliverable occur during the creation of the deliverable or upon its creation.

1.3 Project Types

A project can consist of the following stages:

- Enterprise Planning Stage
- Initiate Stage
- Analysis Stage
- Design Stage
- Construction Stage
- Implement Stage
- Post Implementation Stage
- System Maintenance

The methodology can be used for a number of different types of project. Depending on the type of project, steps in various phases can be eliminated or performed in an accelerated manner.

The methodology can be used for the following types of project:

- Feasibility/Assessment Project (FAP)
- Small System Project (SSP)
- Large System Project (LSP)
- Purchased System Project (PSP)
- Rapid Application Project (RAD)
- Business Process Reengineering Project (BPR)

These project types have the following phases:

Stage/Task	**Small System**	**Large System**	**Purchased System**	**RAD System**	**Feasibility Assessment**
Enterprise Modelling					
Project Initiation	X	X	X	X	X
Analysis	X	X	X		X
Design	X	X	X		
Construction	X	X	X		
Requirements Planning				X	
User Design				X	
Rapid Construction				X	
Implementation	X	X	X	X	
Post Implementation Assessment	X	X	X	X	
System Modification	X	X	X	X	

1.3.1 Feasibility/Assessment Project (FAP)

This project has the following characteristics:

- It is a stand-alone project.
- It is a study.
- Its deliverable is a decision or an evaluation.
- It does not directly develop or buy a system.
- It includes estimating of costs and resources.
- It involves an application development issue.
- It may only include the project initiation stage or a part of it.

1.3.2 Small System Project (SSP)

This project has the following characteristics:

- It may not have budget assigned.
- It affects very few organisation units.
- It has an estimated to cost no more than €100,000.
- It may involve either in-house developed system or purchased software.
- It is expected to require few resources.
- It involves a small number of business processes.
- It involves a small number of entity types.
- It probably produces few outputs.
- It may have a flexible time-line.
- It may have a short-term life.
- It has few system interfaces.
- It is assumed to be new system development.
- It does not involve significant change/revision to existing systems.

1.3.3 Large System Project (LSP)

This project has the following characteristics:

- It has a substantial budget assigned.
- It has a project sponsor assigned.
- In house development is assumed, either with or without the use of external contractors or service providers.
- It affects either multiple organisational units and/or involves a large number of users.
- It may have a significant infrastructural component.
- It involves a large number of business processes.
- It involves a large number of entity types.

1.3.4 Purchased System Project (PSP)

This project has the following characteristics:

- Its intention is to buy software, not build it.
- It has budget assigned to the project.
- It has an estimated cost of more than €100,000.
- It affects a large number of organisational units.
- It affects a large number of business processes.
- It affects a large number of entity types.
- It represents purchased application systems software packages that may be modified by the user through software changes or enhancements prior to implementation and usage.

1.3.5 Rapid Application Project (RAD)

This project has the following characteristics:

- It has an abbreviated analysis stage.
- It combines design and construction stages.
- It involves prototyping.
- It does not update user models.
- It tends to be used for small, stand-alone system.

1.3.6 Business Process Reengineering Project (BPR)

This project has the following characteristics:

- It has executive involvement.
- It involves current physical process analysis.
- It includes logical future process modelling.
- It includes implementation planning.
- It is product/deliverable driven.
- It includes the design of new physical process.
- It frequently triggers an IT development project.

1.4 Project Phase Tasks For Project Types

1.4.1 Project Initiation Phase

Stage/Task	Small System	Large System	Purchased System	Feasibility Assessment
Set initial project objectives and scope	R	R	R	R
Refine project scope	R	R	R	R
Define project's benefits	R	R	R	R
Identify sources of business knowledge	R	R	R	R
Prepare preliminary project timeline	R	R	R	R
Determine preliminary project costs	R	R	R	R
Establish business user participation	R	R	R	R
Identify source of project funding/resources	O; PD	R	R	R
Decide whether to continue with project	R	R	R	R
Prepare project plan	R	R	R	R
Create formal project planning document	O; TPM	R	R	N/A
Set analysis stage standards	R	R	R	N/A
Liaison with control function groups	O; TPM and BPM	R	R	N/A

1.4.2 Analysis Phase

Stage/Task	Small System	Large System	Purchased System	Feasibility Assessment
Acquire initial understanding				
• Gather information sources	R	R	R	R
• List candidate entity types and processes	R	R	R	R
• Develop initial process model	R	R	R	R
• Develop initial entity relationship diagram	R	R	R	O; TPM
• Assess diagrams	R	R	R	O; TPM
• Identify knowledge workers	R	R	R	R
• Confirm business area scope	R	R	R	O; TPM, S and BPM
Produce entity relationship diagram:				
• Name and define entity types	O; SA and TPM	R	R	N/A
• Name and define relationship types	O; SA and TPM	R	R	N/A
• Name and define attribute types	O; SA and TPM	R	R	N/A
• Examine model for entity subtypes	O; SA and TPM	R	R	N/A
• Consider merging entity types	O; SA and TPM	R	R	N/A
• Assess entity relationship diagram	O; SA and TPM	R	R	N/A
Produce process model:				
• Decompose processes	O; TPM	R	R	N/A
• Produce elementary process definitions	O; TPM	R	R	N/A
• Identify business events	O; TPM	R	R	N/A
• Build business event models	O; SA and TPM	R	R	N/A
• Build process logic diagrams	O; SA and TPM	O; SA and TPM	O; SA and TPM	N/A
• Describe logical process connectors	O; SA and TPM	O; SA and TPM	O; SA and TPM	N/A
• Assess logical process models	O; SA and TPM	O; SA and TPM	O; SA and TPM	N/A
Analyse involvement matrices:				
• Develop information needs matrix	O; SA and TPM	O; SA and TPM	O; SA and TPM	N/A
• Develop process/entity type matrix	O; SA and TPM	R	R	N/A
• Refine business area model	O; TPM	R	R	N/A
• Select ADM path	R	R	R	N/A
Confirm outline business model:				
• Check process model correctness	O; QA	R	O; QA	N/A
• Check entity relationship diagram correctness	O; QA	R	O; QA	N/A
• Check conformity to rules and conventions	O; QA	R	O; QA	N/A

Stage/Task	**Small System**	**Large System**	**Purchased System**	**Feasibility Assessment**
• Confirm stability of the outline business model	O; QA	R	R	N/A
• Verify outline business model with users	O; TPM	R	R	N/A
• Refine outline business model	O; TPM	R	R	N/A
Analyse distribution:				
• Analyse business locations	R	R	R	N/A
• Analyse process distribution	O; BPM and TPM	O; BPM and TPM	O; BPM and TPM	N/A
• Analyse entity type distribution	O; BPM and TPM	O; BPM and TPM	O; BPM and TPM	N/A
• Estimate process frequencies	O; TPM	R	R	N/A
• Estimate entity type volumes	O; TPM	R	R	N/A
Define design areas:				
• Establish design areas	R	R	R	N/A
• Refine design area boundaries	O; TPM and SA	R	R	N/A
• Coordinate with business systems architecture	O; TPM, BPM, and SA	R	R	N/A
• Analyse process benefits	O; BPM	R	R	N/A
• Select process mechanisms	R	R	R	N/A
• Define implementation strategy	R	R	R	N/A
• Produce cost-benefit analysis of each design area	O; TPM and BPM	O; TPM and BPM	O; TPM and BPM	N/A
• Develop design area project plan	R	R	R	N/A
• Prepare OBAA stage report	O; TPM	R	R	N/A
Consider purchased software solution:				
• Survey for packaged software solution	O; TPM and BPM	O; TPM and BPM	R	R
• Identify criteria for evaluating software solutions	O; TPM and BPM	N/A	R	N/A
• Request vendor proposals	O; TPM and BPM	N/A	R	N/A
• Evaluate available packaged software solutions	O; TPM and BPM	N/A	R	N/A
• Authorise packaged software solution	O; TPM and BPM	N/A	R	N/A
• Negotiate agreement with packaged software solution vendor	O; TPM and BPM	N/A	R	N/A
• Acquire selected packaged software solution	O; TPM and BPM	N/A	R	N/A
• Reaffirm path selection	R	N/A	R	N/A
• Analyse entity type life cycles				
• Create entity state transition diagram	O; SA	R	O; SA	N/A
• Assess subtypes of entity types	O; SA	R	O; SA	N/A
• Refine entity state diagram	O; SA	R	O; SA	N/A
• Define classifying attribute	O; SA	R	O; SA	N/A

Stage/Task	Small System	Large System	Purchased System	Feasibility Assessment
types				
• Refine business model	O; SA	R	O; SA	N/A
• Analyse process logic				
• Assess process documentation	O; SA	R	O; SA	N/A
• Draw process logic diagram	O; SA	R	O; SA	N/A
• Prepare process action diagram	O; SA	R	O; SA	N/A
• Identify attribute value actions	O; SA	R	O; SA	N/A
• Define algorithms	O; SA	R	O; SA	N/A
• Analyse exception situations	O; SA	R	O; SA	N/A
• Assess process action diagrams	O; SA	R	O; SA	N/A
• Refine business model	O; SA	R	O; SA	N/A
Analyse business rules				
• Define domains	R	R	R	N/A
• Define attribute type properties	R	R	R	N/A
• Define permitted ranges	R	R	R	N/A
• Define integrity conditions	R	R	R	N/A
• Define security requirements	R	R	R	N/A
Analyse current systems:				
• Analyse current systems procedures	R	R	R	N/A
• Analyse current systems data	R	R	R	N/A
• Analyse current systems problems	R	R	R	N/A
• Compare business model with current systems	R	R	R	N/A
Confirm correctness of the detailed business model:				
• Check consistency of quantitative information	O; QA	O; QA	O; QA	N/A
• Check detailed process model correctness	O; QA	O; QA	O; QA	N/A
• Check entity relationship diagram correctness	O; QA	O; QA	O; QA	N/A
• Check conformity to rules and conventions	O; QA	R	O; QA	N/A
• Correct detailed business area model	R	R	R	N/A
• Confirm business model with users	R	R	R	N/A
Reaffirm design area evaluation:				
• Assess implementation strategy	R	R	R	N/A
• Assess cost-benefit analysis	O; BPM	O; BPM	O; BPM	N/A
• Enhance detailed analysis stage report	R	R	R	N/A

1.4.3 Design Phase

Stage/Task	Small System	Large System	Purchased System	Feasibility Assessment
Business System Design Phase				
Design system structure:				
• Define procedure	R	R	X	N/A
• Design procedure interaction	R	R	X	N/A
Design for implementation:				
• Revise implementation strategy	O; TPM	R	X	N/A
• Identify interface data	R	R	R	N/A
• Define implementation algorithms	R	R	R	N/A
• Design bridging procedures	R	R	R	N/A
• Define training requirements	R	R	R	N/A
• Plan rollout	R	R	R	N/A
Design preliminary data structures:				
• Summarise data model usage	O; BPM	O; BPM	O; BPM	N/A
• Design data code values	R	R	R	N/A
• Prepare preliminary data structure	R	R	X	N/A
Define data and procedure interactions:				
• Select procedures for definition	O; TPM	R	O; TPM	N/A
• Summarise data interactions	O; TPM	R	O; TPM	N/A
• Define procedure data access	O; TPM	R	O; TPM	N/A
• Refine data structure	O; TPM	R	O; TPM	N/A
Design procedures:				
• Define human interface procedures	R	R	X	N/A
• Design layouts	R	R	X	N/A
• Prototype the system behaviour	O; TPM & BPM	O; TPM & BPM	O; TPM & BPM	N/A
• Define procedure actions	R	R	X	N/A
Check design completeness	R	R	X	N/A
Check design correctness:				
• Conduct design walk-through	O; TPM	R	X	N/A
• Check data availability	R	R	R	N/A
• Modify lists of elements	R	R	X	N/A
• Check conformity to architectures and standards	R	R	O; TPM & QC	N/A
• Produce correctness quality report	O; TPM	R	X	N/A
Plan for technical design:				
• Analyse benefits	O; BPM	O; BPM	O; BPM	N/A
• Analyse costs	O; BPM	O; BPM	O; BPM	N/A
• Select procedures for implementation	O; TPM & BPM	R	X	N/A
• Group procedures into implementation areas	O; TPM	R	X	N/A
• Sequence the implementation areas	O; TPM	R	X	N/A
• Plan for technical design and construction	R	R	X	N/A
Technical Design Phase				

Stage/Task	**Small System**	**Large System**	**Purchased System**	**Feasibility Assessment**
Define technical context and requirements:				
• Define project standards	R	R	X	N/A
• Define system acceptance criteria	R	R	R	N/A
• Define technical objectives	O; TPM	R	R	N/A
• Evaluate capacity requirements	R	R	R	N/A
• Draw hardware context diagram	O; TPM	R	O; TPM & BPM	N/A
• Draw software context diagram	O; TPM	R	O; TPM	N/A
• Draw language evaluation matrix	O; SA	O; SA	O; SA	N/A
• Place orders for products	O; TPM	O; TPM	O; TPM	N/A
Design the data structures:				
• Prepare load matrices	R	R	R	N/A
• Refine the data structure	R	R	R	N/A
• Define the perceived data structures	R	R	R	N/A
• Design the data storage structure	R	R	R	N/A
• Estimate space requirements	R	R	R	N/A
• Design buffering	R	R	R	N/A
• Design non-database files	R	R	R	N/A
• Identify system performance problems	R	R	R	N/A
• Design on-line data storage recovery	R	R	R	N/A
Develop program specifications:				
• Define programs and modules	R	R	X	N/A
• Design programs and modules	R	R	X	N/A
• Conduct a program/module walk-through	O; TPM	R	X	N/A
Prepare for testing:				
• Define integration groups	R	R	R	N/A
• Define integration test requirements	R	R	R	N/A
Prepare for implementation:				
• Finalise design for implementation	R	R	R	N/A
• Refine training requirements	O; TPM & BPM	R	R	N/A
• Refine rollout plan	O; TPM & BPM	R	R	N/A
• Design operations procedures	R	R	R	N/A
Assess performance	R	R	R	N/A
Design for system testing:				
• Define the scope of system tests	O; TPM	R	R	N/A
• Design system test cycles	O; TPM & BPM	R	R	N/A
• Specify test conditions for	O; TPM &	R	R	N/A

Stage/Task	Small System	Large System	Purchased System	Feasibility Assessment
system testing	BPM			
• Define benchmark testing	O; TPM & BPM	O; TPM	O; TPM	N/A
• Define acceptance test cycles	O; TPM & BPM	R	R	N/A
• Specify test conditions for acceptance tests	O; TPM & BPM	R	R	N/A
Plan construction and implementation stages	R	R	R	N/A

1.4.4 Construction Phase

Stage/Task	Small System	Large System	Purchased System	Feasibility Assessment
Construct computing environment:				
• Establish computing environment	R	R	R	N/A
• Construct network links	R	R	R	N/A
Prepare development procedures and conventions				
• Establish reusable code sets	R	R	X	N/A
• Establish standard development procedures	R	R	X	N/A
Establish database environment:				
• Construct test database and files	R	R	R	N/A
• Construct system test database and files	O; TPM	R	R	N/A
• Prepare production database and files	R	R	R	N/A
Generate implementable module:				
• Write procedure	R	R	X	N/A
• Create code module	R	R	X	N/A
• Debug modules	R	R	R	N/A
• Load Modules into the project code library	R	R	R	N/A
Generate test data and system documents:				
• Generate test data and execution data	R	R	R	N/A
• Devise system test data	R	R	R	N/A
• Document system test case	O; SA	R	R	N/A
• Generate system documents	O; TPM	R	R	N/A
Finalise implementation plan:				
• Finalise implementation strategy	O; TPM and BPM	R	R	N/A
• Develop training plan	R	R	R	N/A
• Finalise contingency plan for implementation failure	R	R	R	N/A
• Complete implementation plan	O; TPM and BPM	R	R	N/A
• Develop training materials	R	R	R	N/A

Stage/Task	Small System	Large System	Purchased System	Feasibility Assessment
• Prepare user implementation procedures	R	R	R	N/A
• Develop data conversion routines	O; TPM and BPM	O; TPM and BPM	O; TPM and BPM	N/A
• Follow-up on outstanding organisational issues	O; BPM	R	R	N/A
• Test implementation arrangements	O; TPM	R	R	N/A
Perform integration tests:				
• Conduct integration tests	O; TPM	R	X	N/A
• Review integration testing	O; TPM	R	X	N/A
Perform system tests:				
• Verify scope of system tests	R	R	R	N/A
• Define system test cycles	R	R	R	N/A
• Confirm system test conditions	R	R	R	N/A
• Conduct system tests	R	R	R	N/A
Perform benchmark tests	O; BPM and TPM	R	O; BPM and TPM	N/A
Perform acceptance tests:				
• Prepare for acceptance testing	O; BPM	R	R	N/A
• Devise the acceptance test data	O; BPM	R	R	N/A
• Conduct acceptance test	O; BPM	R	R	N/A
• Approve the business system	R	R	R	N/A

1.4.5 Implementation Phase

Stage/Task	Small System	Large System	Purchased System	Feasibility Assessment
Train users:				
• Schedule user training sessions	R	R	R	N/A
• Conduct user training	R	R	R	N/A
• Develop permanent user training	O; BPM	R	R	N/A
Perform data conversion:				
• Suspend transaction processing	O; BPM and TPM	O; BPM and TPM	O; BPM and TPM	N/A
• Archive existing system components	R	R	R	N/A
• Execute conversion routines	O; BPM and TPM	O; BPM and TPM	O; BPM and TPM	N/A
• Confirm conversion accuracy	O; BPM and TPM	O; BPM and TPM	O; BPM and TPM	N/A
Install production system:				
• Adjust hardware and system software	R	R	R	N/A
• Train operations staff	R	R	R	N/A
• Migrate system components to production	R	R	R	N/A
• Update production code libraries	R	R	R	N/A

Stage/Task	Small System	Large System	Purchased System	Feasibility Assessment
Accept system installation:				
• Negotiate system/procedure startup schedule	R	R	R	N/A
• Refine agreement on pricing and charging	R	R	R	N/A
• Develop warranty and support agreement	R	R	R	N/A
Support the implementable module:				
• Ensure efficiency of system performance	O; BPM and TPM	R	R	N/A
• Ensure usefulness of system documentation	O; BPM and TPM	R	R	N/A
• Ensure system's operational effectiveness	O; BPM and TPM	R	R	N/A
• Generate project initiation request(s) to correct, extend or modify operational systems	R	R	R	N/A
Respond to emergency situations	R	R	R	N/A

1.4.6 Post Implementation Assessment Stage

Stage/Task	Small System	Large System	Purchased System	Feasibility Assessment
Define project(s) to evaluate	O; AC	R	O; AC	N/A
Gather evaluated project's information	O; AC	R	O; AC	N/A
Evaluate methods and techniques employed in the evaluated project(s)	O; AC	R	O; AC	N/A
Publish post implementation assessment	O; AC	R	O; AC	N/A

1.4.7 System Modification Stage

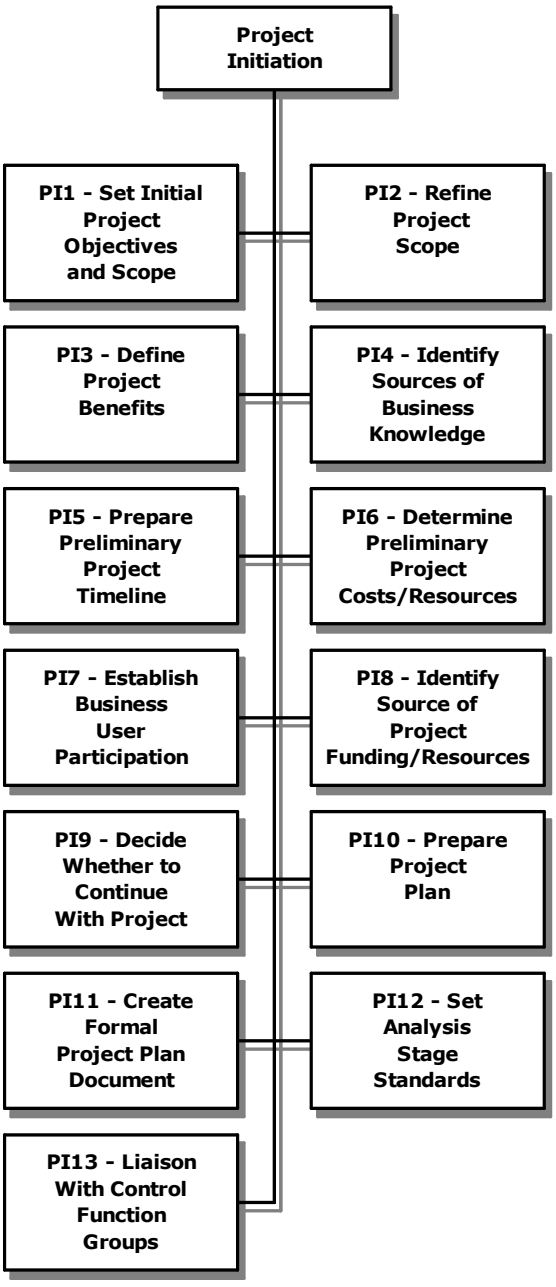
Stage/Task	Small System	Large System	Purchased System	Feasibility Assessment
Evaluate system	O; BPM	O; TPM and BPM	O; TPM and BPM	N/A
Assess changes or enhancement requests	O; BPM	O; TPM and BPM	O; TPM and BPM	N/A
Analyse the nature of the change	O; BPM	O; TPM and BPM	O; TPM and BPM	N/A
Analyse the impact of the change	O; BPM	O; TPM and BPM	O; TPM and BPM	N/A
Execute the change	O; BPM	O; TPM and BPM	O; TPM and BPM	N/A

2. PI - Project Initiation Phase

2.1 Introduction

This is when the individual project is initiated. The project is scoped; the project objectives and benefits are established; the end users identified, the project SWOT (see section 15 Appendix 3 - SWOT Analysis on page 390) is identified; risk analysis is performed; project sponsor are identified; project funding is obtained; and a project plan is approved.

Schematically, the structure of the steps in the project initiation phase is:



2.2 Steps in Project Initiation Phase

2.2.1 PI1 - Set Initial Project Objectives and Scope

Review project initiation request.

Identify project initiator.

Define the project's objectives.

Define the project scope in terms of business activities, data entities, locations and organisation units from the project initiation request submitter’s perspective.

This step creates the following deliverables:

- **Initial Definition Of Project's Scope And Objectives** - see section 2.3.1.1 PID1.1 - Initial Definition Of Project Scope And Objectives on page 37

Prepare Needs Analysis Report containing findings.

This step creates the following deliverables:

- **Needs Analysis Report** - see section 2.3.5 PID6 - Needs Analysis Report on page 41
- **Project Scope Document** - see section 2.3.2 PID2 - Project Scope Document on page 39

2.2.2 PI2 - Refine Project Scope

Identify business activities represented in the Company Information Model that might be impacted by the identified business need(s).

Identify and exclude those business activities impacted by the identified business need(s) that have previously been analysed.

Identify data entities represented in the data model that might be involved with the identified business need(s).

Identify and exclude those data entities involved with the identified business need(s) that have previously been analysed.

Identify the locations represented in the Company Information Model that might be involved with the identified business need(s).

Identify the organisation units represented in the Company Information Model that might be involved with the identified business need(s).

Determine whether any other project initiation requests in the Company queue address the same business area as this request and whether they might be combined with this request to address a broader set of business needs.

Identify existing information systems included in the proposed project scope.

Identify existing files and databases included in the proposed project scope.

Determine whether existing information systems, files and databases are potential candidates to utilise, modify or extend to satisfy the identified business need(s).

Determine the extent of the adjustments that need to be made to the proposed project scope and adjust the scope of the projected study or system as appropriate.

This step updates the following deliverables:

- **Refined Project Scope And Objectives** - see section 2.3.1.2 PID1.2 - Refined Project Scope And Objectives on page 37

Determine the dependencies of this project on other projects.

This step creates the following deliverables:

- **Dependencies Between Project Tasks Or Between This Project And Other Projects** - see section 2.3.3.5 PID3.5 - Dependencies Between Project Tasks Or Between This Project And Other Projects on page 40

Update the Needs Analysis Report.

This step updates the following deliverables:

- **Needs Analysis Report** - see section 2.3.5 PID6 - Needs Analysis Report on page 41

Based on the proposed project scope, evaluate and adjust the boundaries of the business area derived from the information system architecture.

This step updates the following deliverables:

- **Project Scope Document** - see section 2.3.2 PID2 - Project Scope Document on page 39

2.2.3 PI3 - Define Project Benefits

Reassess the project's benefits defined in the Needs Analysis Report in light of the altered project scope and objectives.

Assess the degree to which the projected study or system will satisfy the identified business need(s).

Utilise the Needs Analysis Report to determine the benefits to be gained by implementing the projected study or system.

This step creates the following deliverables:

- **Definition Of Project Benefits** - see section 2.3.1.3 PID1.3 - Definition Of Project Benefits on page 37

Adjust Needs Analysis Report to reflect benefit information.

This step updates the following deliverables:

- **Needs Analysis Report** - see section 2.3.5 PID6 - Needs Analysis Report on page 41

2.2.4 PI4 - Identify Sources of Business Knowledge

Identify the organisation units and business locations performing the business activities included within the scope of the project.

Identify the organisation units and business locations performing the business activities that maintain the data entities referenced within the scope of the project.

Identify users of the data as additional sources of business knowledge.

This step creates the following deliverables:

- **Identified Sources Of Authoritative Business Knowledge** - see section 2.3.1.4 PID1.4 - Identified Sources Of Authoritative Business Knowledge on page 37

2.2.5 PI5 - Prepare Preliminary Project Timeline

Review Analysis and Design Methodology and select appropriate deliverables and tasks.

Define resource assumptions for the next stage.

Utilise standard estimating techniques to project a timeline for the next stage.

Estimate resource assumptions for the design, construction and implementation stages if their scope and system technology is known.

Identify those factors that will hamper or enhance the pace of the project and adjust the timelines accordingly.

Identify the dependencies among tasks in the next stage of this project.

This step creates the following deliverables:

- **Dependencies Between Project Tasks Or Between This Project And Other Projects** - see section 2.3.3.5 PID3.5 - Dependencies Between Project Tasks Or Between This Project And Other Projects on page 40
- **Preliminary Project Timelines For The Next Stage** - see section 2.3.3.1 PID3.1 - Preliminary Project Timelines For The Next Stage on page 39
- **Preliminary Project Timelines For The Design, Construction and Implementation Stages** - see section 2.3.3.4 PID3.4 - Preliminary Project Timelines For The Design, Construction and Implementation Stages on page 40

2.2.6 PI6 - Determine Preliminary Project Costs/Resources

Determine the expected costs/resources associated with the scoped project.

Relate and assess preliminary project costs/resources to projected project benefits.

This step creates the following deliverables:

- **Preliminary Project Cost For Next Stage** - see section 2.3.1.5 PID1.5 - Preliminary Project Cost For Next Stage on page 37
- **Preliminary Project Cost For Design, Construction and Implementation Stages** - see section 2.3.1.6 PID1.6 - Preliminary Project Cost For Design, Construction, and Implementation Stages on page 37
- **Cost Benefit Analysis** - see section 2.3.1.8 PID1.8 - Cost Benefit Analysis on page 38

2.2.7 PI7 - Establish Business User Participation

Identify potential information users of the product of the study or system.

Identify the stakeholders in the project.

Identify the maintainers of the data entities referenced by the activities within the scope of the project.

Identify candidate end users of the projected system and determine the potential nature and degree of their involvement in the project.

Identify the information users and maintainers of the data entities referenced by the activities included in the project scope.

This step creates the following deliverables:

- **End User's Assignment Of Measurable Resources To The Project** - see section 2.3.3.4 PID3.4 - Preliminary Project Timelines For The Design, Construction and Implementation Stages on page 40

- **Project Participant's Commitment** - see section 2.3.3.3 PID3.3 - Project Participant's Commitment on page 40

2.2.8 PI8 - Identify Source of Project Funding/Resources

Identify possible sources of project funding.

Draft pricing agreement.

Prepare project funding request.

Prepare estimate of funding required for the project life cycle.

Secure agreement with project sponsor regarding project scope, projected resource requirements and required funding.

This step creates the following deliverables:

- **Refined Project Scope, Projected Resource Requirement's And Planned Funding Level for the Next Stage** - see section 2.3.4.1 PID4.1 - Refined Project Scope, Projected Resource Requirement's And Planned Funding Level for the Next Stage on page 40
- **Pricing Agreement** - see section 7.3.4.2 ID4.2 - Pricing Agreement on page 273

2.2.9 PI9 - Decide Whether to Continue With Project

Evaluate cost benefit analysis.

Evaluate risk analysis.

Evaluate SWOT.

Evaluate degree of end user participation.

Forecast required funds and timelines for each development stage along with a schedule for confirming or revising each Stage's funding.

This step creates the following deliverables:

- **Decision On Whether To Proceed With Project As Defined** - see section 2.3.6 PID7 - Decision As To Whether To Proceed With Project As Defined on page 42

2.2.10 PI10 - Prepare Project Plan

Prepare detailed project plan for the next stage of the project.

Prepare project overview for the design, construction and implementation stage.

This step creates the following deliverables:

- **Refined Project Plan For Next Stage** - see section 2.3.4.3 PID4.3 - Refined Project Plan For Next Stage on page 41

2.2.11 PI11 - Create Formal Project Plan Document

Create a project plan document containing all factors included in the project plan for the next stage.

2.2.12 PI12 - Set Analysis Stage Standards

Establish project standards for modelling that will be employed in the analysis stage.

2.2.13 PI13 - Liaison With Control Function Groups

Identify remaining control function groups that have an interest in or jurisdiction over this project (e.g., security, data stewardship, production control, assessing enabling technologies).

Advise the identified organisation control function groups of the project's objectives, projected benefits, costs/resources, participants, SWOT, risk factors, funding source(s) and time-lines.

Establish role for selected organisation control function groups in the project.

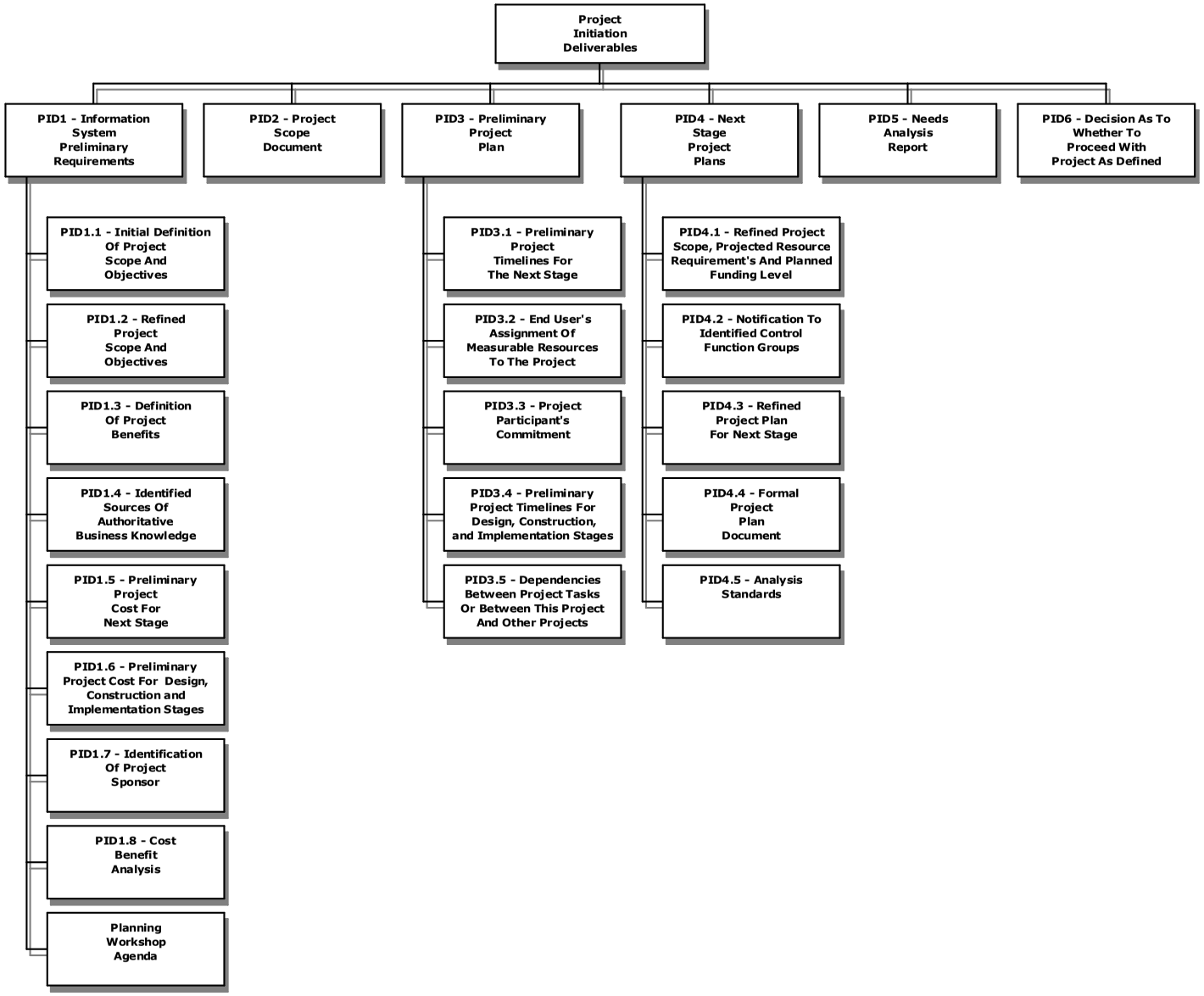
Update the Project Initiation Request status to reflect the completion o of the Project Initiation Stage.

This step creates the following deliverables:

- **Notification To Identified Organisation Control Function Groups** - see section 2.3.4.2 PID4.2 - Notification To Identified Control Function Groups on page 41

2.3 Project Initiation Deliverables

Schematically, the structure of the deliverables in the project initiation phase is:


Project Initiation Deliverables
PID1 - Information System Preliminary Requirements
PID2 - Project Scope Document
PID3 - Preliminary Project Plan
PID4 - Next Stage Project Plans
PID5 - Needs Analysis Report
PID6 - Decision As To Whether To Proceed With Project As Defined
PID1.1 - Initial Definition Of Project Scope And Objectives
PID1.2 - Refined Project Scope And Objectives
PID1.3 - Definition Of Project Benefits
PID1.4 - Identified Sources Of Authoritative Business Knowledge
PID1.5 - Preliminary Project Cost For Next Stage
PID1.6 - Preliminary Project Cost For Design, Construction and Implementation Stages
PID1.7 - Identification Of Project Sponsor
PID1.8 - Cost Benefit Analysis
Planning Workshop Agenda
PID3.1 - Preliminary Project Timelines For The Next Stage
PID3.2 - End User's Assignment Of Measurable Resources To The Project
PID3.3 - Project Participant's Commitment
PID3.4 - Preliminary Project Timelines For Design, Construction, and Implementation Stages
PID3.5 - Dependencies Between Project Tasks Or Between This Project And Other Projects
PID4.1 - Refined Project Scope, Projected Resource Requirement's And Planned Funding Level
PID4.2 - Notification To Identified Control Function Groups
PID4.3 - Refined Project Plan For Next Stage
PID4.4 - Formal Project Plan Document
PID4.5 - Analysis Standards

2.3.1 PID1 - Information System Preliminary Requirements

To provide a communication vehicle for end users and systems units that will address a business need using common terminology and format to generate development estimates.

2.3.1.1 PID1.1 - Initial Definition Of Project Scope And Objectives

Associated business activities, data entities, business locations, organisation units and business areas impacted by or involved with the identified business need(s).

Existing information systems, files and databases included in the proposed information project scope or which may be utilised, modified or extended to satisfy the identified business need(s).

2.3.1.2 PID1.2 - Refined Project Scope And Objectives

2.3.1.3 PID1.3 - Definition Of Project Benefits

The deliverable may contain all or part of the following elements:

- Assessment of the degree to which the projected information system will meet the identified business need(s).
- Specific projected benefits (e.g., head-count reduction, improved productivity).
- Dollar denominated value of the projected benefits.
- Time frame over which the projected benefits are expected to accrue.
- Time criticality of the benefit.

2.3.1.4 PID1.4 - Identified Sources Of Authoritative Business Knowledge

2.3.1.5 PID1.5 - Preliminary Project Cost For Next Stage

2.3.1.6 PID1.6 - Preliminary Project Cost For Design, Construction, and Implementation Stages

2.3.1.7 PID1.7 - Identification Of Project Sponsor

2.3.1.8 PID1.8 - Cost Benefit Analysis

A parameter-based estimate of the total cost to develop and cutover the proposed area, based on the present definition of the areas scope, against the quantitative estimate of benefits that could be achieved by the area. The analysis can be done for several scopes, depending on the task involved. Typical scopes are per business area, design area and/or procedure.

Costs accrue from the implementation of an activity or entity type. As with benefits, costs are either tangible or intangible. Tangible costs are directly caused by the implementation of an activity or entity type and can be attached to a direct monetary value.

Other tangible costs include things like training and implementation. Intangible costs are often more important than tangible costs. They include things like disruption to the enterprise and consequential loss of customer service, and non-availability of staff. Accurate costing of systems development from analysis information is generally not possible. Even so, estimates are essential. Most organisations have rules of thumb accumulated by experience, but development costs can change significantly for many reasons, including:

- The programmer or designer employed
- The software in use
- Database size
- Database complexity
- Transaction volumes
- Transaction complexity
- Experience with the technology to be used

Components of the Cost-Benefit Analysis

- Cost-Benefit Summary Report
- Benefit Analysis Matrix
- Benefit Analysis Table

2.3.1.9 PID1.9 - Planning Workshop Agenda

The Planning Workshop Agenda establishes the general topics to be addressed by the participants of the JRP Workshop.

A typical Planning Workshop Agenda will identify these topics:

- Opening Presentation: to explain the objectives of the workshop and the general rules and conventions that will be followed.

- Architecture Review: to confirm and enhance the various diagrams describing the data and activity architectures that surround the proposed system.

- Function Selection: to identify the business functions that the proposed system will support, and to identify benefits associated with supporting each business function.

- Implementation Issues: to discuss problems related to the development of the system software and the conversion from existing systems to the new system. These general topics will be expanded as necessary, and additional topics added when appropriate.

2.3.2 PID2 - Project Scope Document

The project scope document identifies the business activities, business data, business locations and organisation units that comprise the scope of the project. The project boundary represents the dividing line between which components are included and which are excluded from the project scope.

The deliverable may contain all or part of the following elements:

Graphical illustration and narrative definitions of data entities and entity relationships.

Company functional model conveying the business activities contained in the business area

2.3.3 PID3 - Preliminary Project Plan

To provide a rough estimate of the time frame in which all the ADM stages might be accomplished for an information system development project. It will indicate the elapsed time for completion of the project.

2.3.3.1 PID3.1 - Preliminary Project Timelines For The Next Stage

The preliminary project timeline may contain all or part of the following elements:

- Tailored set of Application Development Methodology deliverables and tasks.
- Resource assumptions.

  Estimated staff months.
  Estimated development cost.
  Estimated incremental hardware/software costs.

- Identified factors that will abnormally affect the progress of the project.
- Preliminary project schedule.
- Information users of the product of the information system.
- Identification of the creators, maintainers and users of the data referenced within the scope of the project.
- Identification of project participants.

2.3.3.2 PID3.2 - End User's Assignment Of Measurable Resources To The Project

2.3.3.3 PID3.3 - Project Participant's Commitment

2.3.3.4 PID3.4 - Preliminary Project Timelines For The Design, Construction and Implementation Stages

Each preliminary project timeline may contain all or part of the following elements:

- Tailored set of Application Development Methodology deliverables and tasks.
- Identified factors that will abnormally affect the progress of the project.
- Resource assumptions.

  Estimated staff months by stage.
  Estimated development cost by stage.
  Estimated incremental hardware/software costs.
  Estimated total development cost.

- Preliminary project schedule.

2.3.3.5 PID3.5 - Dependencies Between Project Tasks Or Between This Project And Other Projects

2.3.4 PID4 - Next Stage Project Plans

The project plan for the Analysis Stage defines the resource requirements and timelines necessary for that stage.

2.3.4.1 PID4.1 - Refined Project Scope, Projected Resource Requirement's And Planned Funding Level for the Next Stage

2.3.4.2 PID4.2 - Notification To Identified Control Function Groups

2.3.4.3 PID4.3 - Refined Project Plan For Next Stage

Refined resource requirements estimate for Next Stage.

Refined funding level planned for Next Stage.

Project tasks.

Dependencies between tasks.

Project time line.

Project milestones.

2.3.4.4 PID4.4 - Formal Project Plan Document

This document represents all assumptions, issues and decisions that affect the project plan for the next stage.

Refined resource requirements estimate for Next Stage.

Refined funding level planned for Next Stage.

Project tasks.

Dependencies between tasks.

Project time line.

Project milestones.

2.3.4.5 PID4.5 - Analysis Standards

These standards include defining consistent modelling techniques, modelling notation and naming conventions for all types of objects (entities, attributes, relationships, processes, events, etc.).

2.3.5 PID6 - Needs Analysis Report

2.3.6 PID7 - Decision As To Whether To Proceed With Project As Defined

As a result of this decision, the next stage of the project will be initiated or the project will not proceed without further consideration by the user.

3. EM - Enterprise Modelling Phase

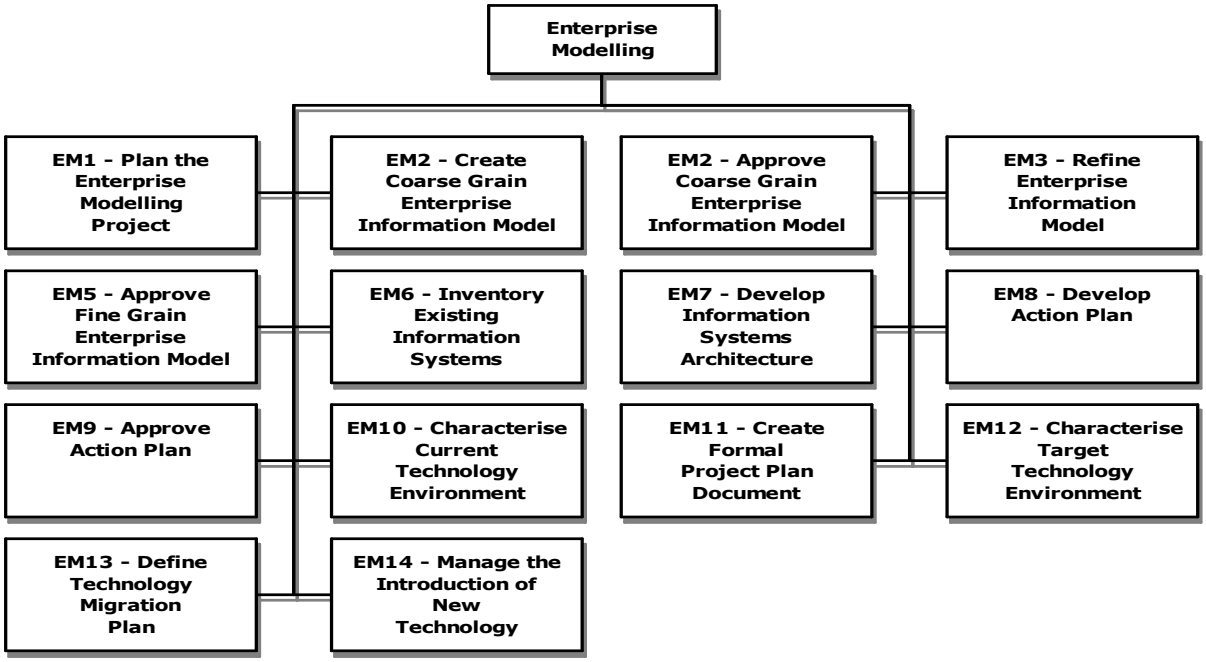
3.1 Introduction

This phase of the ADM contains the tasks that perform the enterprise's strategic planning aspect of the information systems life cycle. The objective of this phase is to identify those business areas (broad, major information system projects) that are significant to the enterprise's strategic information needs. The company's information system architecture and data architecture are defined and the business areas are prioritised. This stage also addresses the funding of strategic information systems projects.

The input for the phase is the company's business functions, information requirements and perceived need for increased technology support.

The output from the phase is the one or more defined (scoped) business areas; each with a defined set of business activities and data. Another type of stage output can be budgetary approval for a slate of one or more information systems projects. A third type of output is the formulation of a business person's request for the creation of an information system to produce needed business information.

Schematically, the structure of the steps in the enterprise modelling phase is:



3.2 Steps in Enterprise Modelling Phase

3.2.1 EM1 - Plan the Enterprise Modelling Project

Identify the deliverables, tasks and resources to model the organisation as an enterprise. Define the time line to accomplish these tasks using those resources and create a project plan. Establish a steering committee to oversee the enterprise modelling project.

This step creates the following deliverables:

- **Enterprise Modelling Project Plan** - see section 3.3.1 EMD1 - Enterprise Modelling Project Plan on page 52

3.2.2 EM2 - Create Coarse Grain Enterprise Information Model

Build a model of the enterprise's business functions, subject areas of data, organisational structure, business locations, goals and objectives and enterprise concerns. Associate the business functions with the subject areas, business locations, organisation units and enterprise goals and concerns.

This step creates the following deliverables:

- **Inventory of Enterprise Directions and Concerns** - see section 3.3.2 EMD2 - Inventory of Enterprise Directions and Concerns on page 52
- **Inventory of Enterprise Business Locations and Organisation Units** - see section 3.3.3 EMD3 - Inventory of Enterprise Business Locations and Organisation Units on page 52
- **Inventory of Business Functions** - see section 3.3.4 EMD4 - Inventory of Business Functions on page 52
- **Inventory of Subject Areas** - see section 3.3.5 EMD5 - Inventory of Subject Areas on page 52
- **Association of Business Functions with Subject Areas, Enterprise Directions, Concerns, Business Locations and Organisation Units** - see section 3.3.6 EMD6 - Association of Business Functions with Subject Areas, Enterprise Directions, Concerns, Business Locations and Organisation Units on page 52
- **Association of Subject Areas with Enterprise Directions, Concerns, Business Locations and Organisation Units** - see section 3.3.6 EMD6 - Association of Business Functions with Subject Areas, Enterprise Directions, Concerns, Business Locations and Organisation Units on page 52
- **Scope of Fine Grain Information Model** - see section 3.3.8 EMD8 - Scope of Fine Grain Information Model on page 53

3.2.3 EM3 - Approve Coarse Grain Enterprise Information Model

The steering committee approves the coarse grain enterprise information model. The scope of the fine grain model is also defined.

This step updates the following deliverables:

- **Inventory of Enterprise Directions and Concerns** - see section 3.3.2 EMD2 - Inventory of Enterprise Directions and Concerns on page 52
- **Inventory of Enterprise Business Locations and Organisation Units** - see section 3.3.3 EMD3 - Inventory of Enterprise Business Locations and Organisation Units on page 52
- **Inventory of Business Functions** - see section 3.3.4 EMD4 - Inventory of Business Functions on page 52
- **Inventory of Subject Areas** - see section 3.3.5 EMD5 - Inventory of Subject Areas on page 52
- **Association of Business Functions with Subject Areas, Enterprise Directions, Concerns, Business Locations and Organisation Units** - see section 3.3.6 EMD6 - Association of Business Functions with Subject Areas, Enterprise Directions, Concerns, Business Locations and Organisation Units on page 52
- **Association of Subject Areas with Enterprise Directions, Concerns, Business Locations and Organisation Units** - see section 3.3.6 EMD6 - Association of Business Functions with Subject Areas, Enterprise Directions, Concerns, Business Locations and Organisation Units on page 52
- **Scope of Fine Grain Information Model** - see section 3.3.8 EMD8 - Scope of Fine Grain Information Model on page 53

3.2.4 EM4 - Refine Enterprise Information Model

The coarse grain enterprise information model is driven to a finer grain model of the company. Associate the business processes with the entity types, business locations, organisation units, business goals and concerns.

This step creates the following deliverables:

- **Inventory of Business Processes** - see section 3.3.9 EMD9 - Inventory of Business Processes on page 53
- **Inventory of Entity Types** - see section 3.3.10 EMD10 - Inventory of Entity Types on page 53

- **Association of Business Processes with Enterprise Business Locations, Organisation Units, Entity Types, Enterprise Directions and Concerns** - see section 3.3.11 EMD11 - Association of Business Processes with Enterprise Business Locations, Organisation Units, Entity Types, Enterprise Directions and Concerns on page 53

- **Association of Entity Types with Enterprise Directions, Concerns, Business Locations and Organisation Units** - see section 3.3.12 EMD12 - Association of Entity Types with Enterprise Directions, Concerns, Business Locations and Organisation Units on page 53

3.2.5 EM5 - Approve Fine Grain Enterprise Information Model

The fine grain enterprise information model is approved by the steering committee.

This step updates the following deliverables:

- **Inventory of Business Processes** - see section 3.3.9 EMD9 - Inventory of Business Processes on page 53

- **Inventory of Entity Types** - see section 3.3.10 EMD10 - Inventory of Entity Types on page 53

- **Association of Business Processes with Enterprise Business Locations, Organisation Units, Entity Types, Enterprise Directions and Concerns** - see section 3.3.11 EMD11 - Association of Business Processes with Enterprise Business Locations, Organisation Units, Entity Types, Enterprise Directions and Concerns on page 53

- **Association of Entity Types with Enterprise Directions, Concerns, Business Locations and Organisation Units** - see section 3.3.12 EMD12 - Association of Entity Types with Enterprise Directions, Concerns, Business Locations and Organisation Units on page 53

3.2.6 EM6 - Inventory Existing Information Systems

Identify and assess the current application systems and the data files and databases they utilise. Associate those current systems and data to with the finest grain version of the logical enterprise information model built earlier.

This step creates the following deliverables:

- **Inventory of Mechanisms and Data Collections** - see section 3.3.13 EMD13 - Inventory of Mechanisms and Data Collections on page 53

- **Assessments of Mechanisms and Data Collections** - see section 3.3.14 EMD14 - Assessments of Mechanisms and Data Collections on page 54

- **Associations of Existing Information Systems to Enterprise Activities, Data, Business Locations and Organisation Units** - see section 3.3.15 EMD15 - Associations of Existing Information Systems to Enterprise Activities, Data, Business Locations and Organisation Units on page 54

3.2.7 EM7 - Develop Information Systems Architecture

Utilise the associations of modelling objects created earlier to identify the major information system projects (business areas) comprising the enterprise.

This step updates the following deliverables:

- **Data Architecture** - see section 3.3.17 EMD17 - Data Architecture on page 54

This step creates the following deliverables:

- **Prioritised Business Areas** - see section 3.3.18 EMD18 - Prioritised Business Areas on page 54

3.2.8 EM8 - Develop Action Plan

Establish the prioritisation criteria to use in developing the enterprise action plan. Apply the prioritisation criteria to the business areas previously identified. Formalise the prioritised business areas into an action plan.

This step creates the following deliverables:

- **Strategic Enterprise Information Plan** - see section 3.3.19 EMD19 - Strategic Enterprise Information Plan on page 54

3.2.9 EM9 - Approve Action Plan

The steering committee approves the action plan.

This step updates the following deliverables:

- **Strategic Enterprise Information Plan** - see section 3.3.19 EMD19 - Strategic Enterprise Information Plan on page 54

3.2.10 EM10 - Identify Information Resource Management Principles

A technology planning group is formed and they define the information resource management principles that represent the company's philosophy.

This step creates the following deliverables:

- **Information Resource Management Principles** - see section 3.3.20 EMD20 - Information Resource Management Principles on page 54

3.2.11 EM11 - Characterise Current Technology Environment

The technology environment currently employed by the company is characterised.

This step creates the following deliverables:

- **Characterisation of Current Technology Environment** - see section 3.3.21 EMD21 - Characterisation of Current Technology Environment on page 55

3.2.12 EM12 - Characterise Target Technology Environment

The technology environment envisioned by the company at a specified future date is characterised. The IRM principles defined earlier will help guide this task.

This step creates the following deliverables:

- **Characterisation of Target Technology Environment** - see section 3.3.22 EMD22 - Characterisation of Target Technology Environment on page 55

3.2.13 EM13 - Define Technology Migration Plan

Identify the steps that the company would take to move from the current technology environment to the environment characterised by the targeted technology architecture.

This step updates the following deliverables:

- **Characterisation of Target Technology Environment** - see section 3.3.22 EMD22 - Characterisation of Target Technology Environment on page 55

This step creates the following deliverables:

- **Technology Migration Plan** - see section 3.3.23 EMD23 - Technology Migration Plan on page 55

3.2.14 EM14 - Manage the Introduction of New Technology

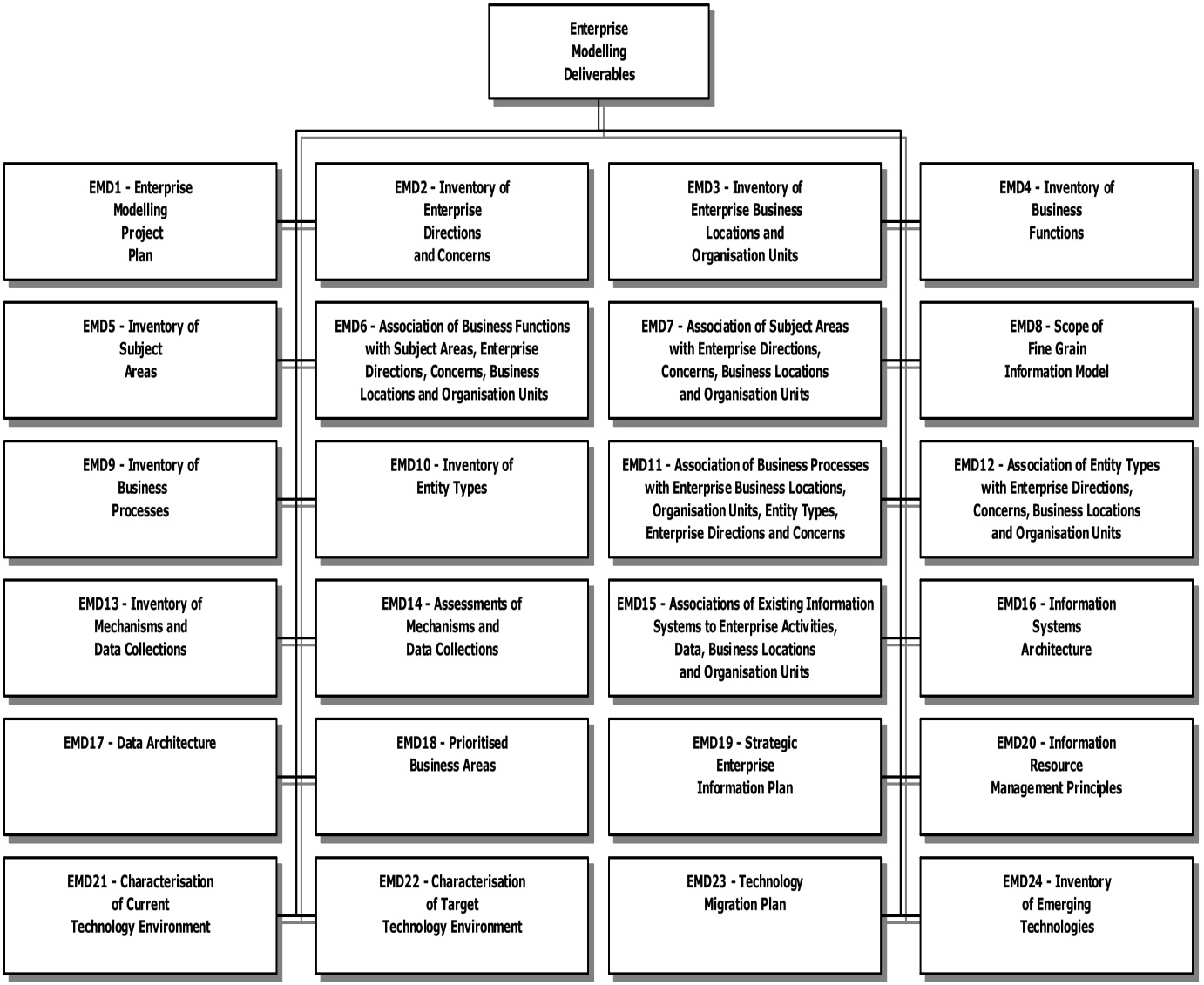
Evaluate new technology to determine its appropriateness for the company.

This step creates the following deliverables:

- **Inventory of Emerging Technologies** - see section 3.3.24 EMD24 - Inventory of Emerging Technologies on page 55

3.3 Enterprise Modelling Phase Deliverables

Schematically, the structure of the deliverables in the enterprise modelling phase is:


Enterprise Modelling Deliverables
EMD1 - Enterprise Modelling Project Plan
EMD2 - Inventory of Enterprise Directions and Concerns
EMD3 - Inventory of Enterprise Business Locations and Organisation Units
EMD4 - Inventory of Business Functions
EMD5 - Inventory of Subject Areas
EMD6 - Association of Business Functions with Subject Areas, Enterprise Directions, Concerns, Business Locations and Organisation Units
EMD7 - Association of Subject Areas with Enterprise Directions, Concerns, Business Locations and Organisation Units
EMD8 - Scope of Fine Grain Information Model
EMD9 - Inventory of Business Processes
EMD10 - Inventory of Entity Types
EMD11 - Association of Business Processes with Enterprise Business Locations, Organisation Units, Entity Types, Enterprise Directions and Concerns
EMD12 - Association of Entity Types with Enterprise Directions, Concerns, Business Locations and Organisation Units
EMD13 - Inventory of Mechanisms and Data Collections
EMD14 - Assessments of Mechanisms and Data Collections
EMD15 - Associations of Existing Information Systems to Enterprise Activities, Data, Business Locations and Organisation Units
EMD16 - Information Systems Architecture
EMD17 - Data Architecture
EMD18 - Prioritised Business Areas
EMD19 - Strategic Enterprise Information Plan
EMD20 - Information Resource Management Principles
EMD21 - Characterisation of Current Technology Environment
EMD22 - Characterisation of Target Technology Environment
EMD23 - Technology Migration Plan
EMD24 - Inventory of Emerging Technologies

3.3.1 EMD1 - Enterprise Modelling Project Plan

A plan for tasks, resources and time to produce the desired deliverables of the enterprise modelling stage.

3.3.2 EMD2 - Inventory of Enterprise Directions and Concerns

A model of the organisation's mission, goals, objectives, critical success factors and concerns.

3.3.3 EMD3 - Inventory of Enterprise Business Locations and Organisation Units

A model of the geographic locations where the organisation does business; an organisation chart of the organisation; and an association of the business locations and organisation units that identifies where each organisation unit is located geographically.

3.3.4 EMD4 - Inventory of Business Functions

A model of the business functions performed by the organisation.

3.3.5 EMD5 - Inventory of Subject Areas

A model of the data subjects considered important to the organisation.

3.3.6 EMD6 - Association of Business Functions with Subject Areas, Enterprise Directions, Concerns, Business Locations and Organisation Units

Association of business function involves subject area, business function supports enterprise directions, business function addresses enterprise concerns, business function is performed at business location, and business function is performed by organisation unit.

3.3.7 EMD7 - Association of Subject Areas with Enterprise Directions, Concerns, Business Locations and Organisation Units

Association of the availability of data subject supports achieving enterprise direction, the lack of access to data subject inhibits addressing enterprise concern, data subject is needed at business location, and data subject is used by organisation unit.

3.3.8 EMD8 - Scope of Fine Grain Information Model

The definition of the boundaries of the fine grain model.

3.3.9 EMD9 - Inventory of Business Processes

A model of the business processes performed by the organisation.

3.3.10 EMD10 - Inventory of Entity Types

A model of the entity types needed to support the organisation's business processes.

3.3.11 EMD11 - Association of Business Processes with Enterprise Business Locations, Organisation Units, Entity Types, Enterprise Directions and Concerns

Association of business process performed at enterprise business location, business process performed by organisation unit, business process involves entity type, business process supports enterprise direction, and business process addresses enterprise concern.

3.3.12 EMD12 - Association of Entity Types with Enterprise Directions, Concerns, Business Locations and Organisation Units

Association of entity type availability supports achievement of enterprise direction, lack of entity type availability inhibits enterprise concern, entity type is available at business location, and entity type is used by organisation unit.

3.3.13 EMD13 - Inventory of Mechanisms and Data Collections

Inventory of existing information systems (mechanism) and inventory of existing data files (data collections).

3.3.14 EMD14 - Assessments of Mechanisms and Data Collections

An assessment of the usability, maintainability and convertibility of the existing information systems and data files.

3.3.15 EMD15 - Associations of Existing Information Systems to Enterprise Activities, Data, Business Locations and Organisation Units

Associations of existing information system implements enterprise activity, existing data file implements entity type, existing information system is executed at business location, and existing information system is operated by and supplies information to organisation unit.

3.3.16 EMD16 - Information Systems Architecture

Association of business activities creating, updating, retrieving or deleting enterprise data. Identification of business areas containing business activities and enterprise data.

3.3.17 EMD17 - Data Architecture

Identification and structure of enterprise data and identification of business activities that populate, use and maintain it.

3.3.18 EMD18 - Prioritised Business Areas

Business areas are prioritised by the degree they support or address the needs and goals of the enterprise as a whole.

3.3.19 EMD19 - Strategic Enterprise Information Plan

Estimates of resources and time required to analyse prioritised business areas.

3.3.20 EMD20 - Information Resource Management Principles

A statement of the principles and philosophy underlying information technology management's beliefs, goals and objectives.

3.3.21 EMD21 - Characterisation of Current Technology Environment

The forms of current technology employed by the organisation are characterised in the three classes of technology architecture. Those three classes of technology are:

1. **Hardware Architecture** - Identification of the types of hardware and the nature of their use in a particular computing environment.

2. **Communications Architecture** - Identification of the types of communications facilities and the use to which they will be put in a particular computing environment.

3. **Software Architecture** - Identification of the types of computing software utilised to plan, develop, operate and maintain the computing environment. Application software is specifically excluded.

3.3.22 EMD22 - Characterisation of Target Technology Environment

The forms of technology targeted by the organisation are characterised in the three classes of technology architecture so. Those three classes of technology are:

1. **Hardware Architecture** - Identification of the types of hardware and the nature of their use in a particular computing environment.

2. **Communications Architecture** - Identification of the types of communications facilities and the use to which they will be put in a particular computing environment.

3. **Software Architecture** - Identification of the types of computing software utilised to plan, develop, operate and maintain the computing environment. Application software is specifically excluded.

3.3.23 EMD23 - Technology Migration Plan

The plan defines the steps that the organisation would take to move from the current technology environment to the targeted technology architecture.

3.3.24 EMD24 - Inventory of Emerging Technologies

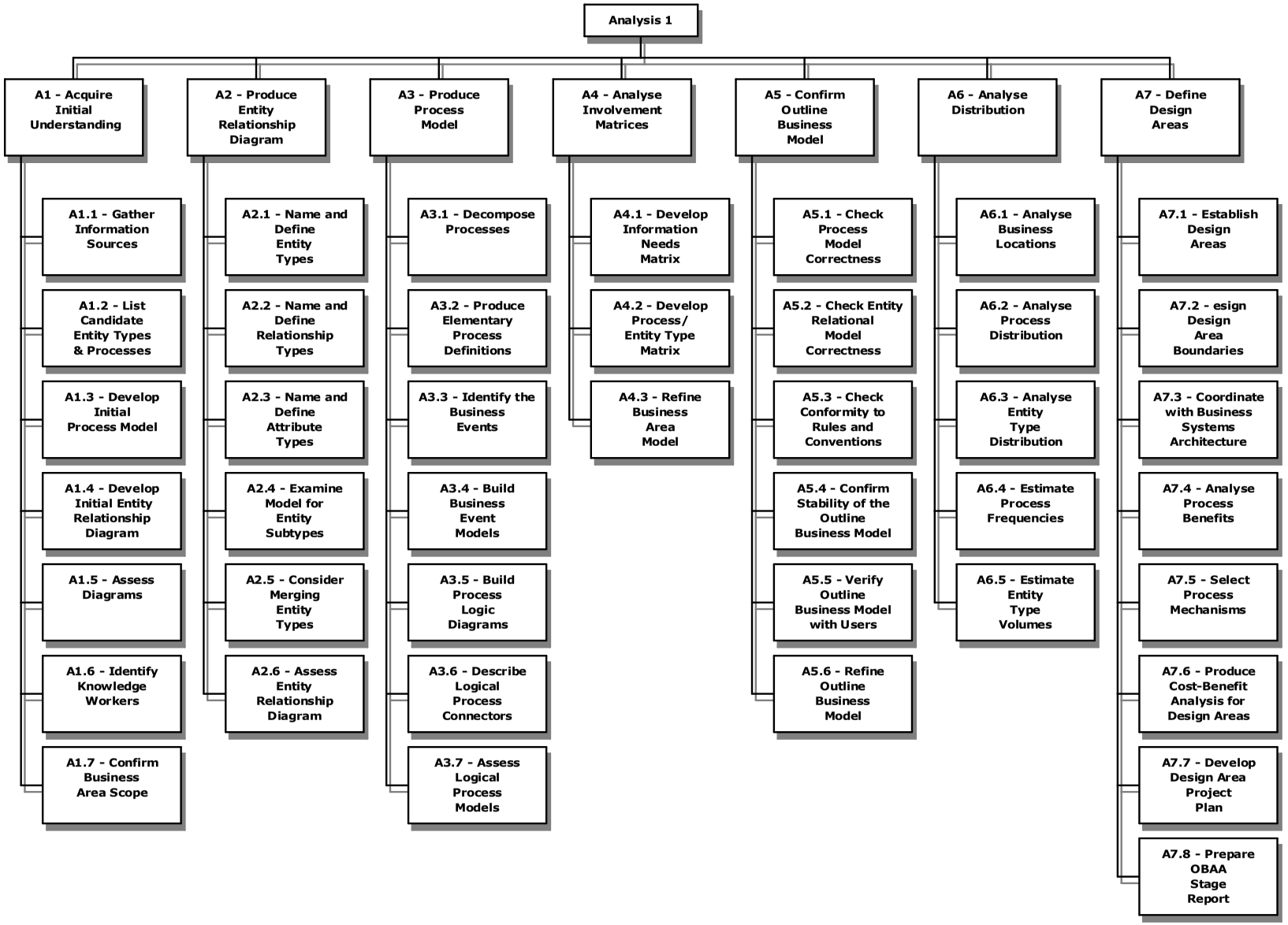
An inventory of leading edge technologies that the organisation will evaluate for enterprise use

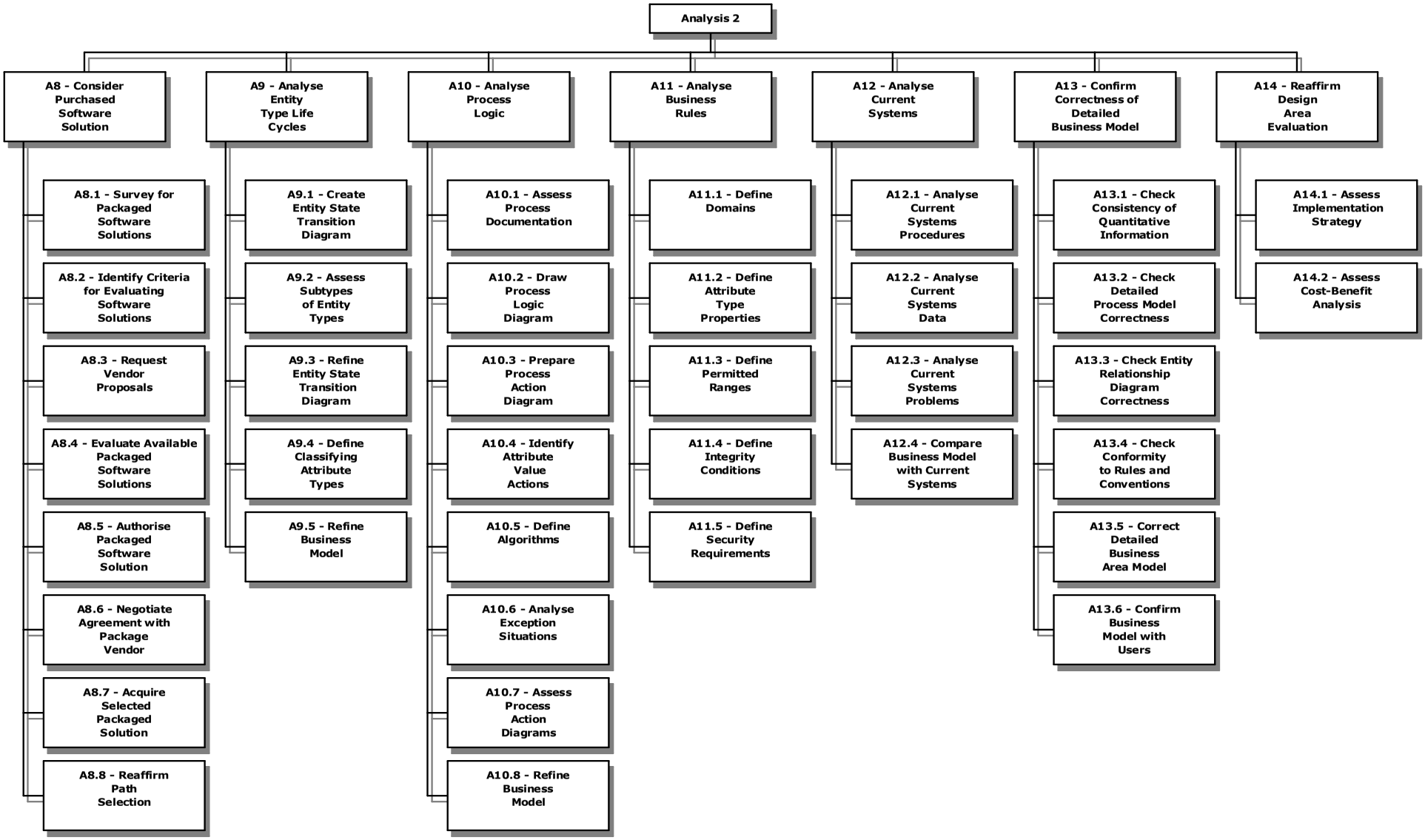
4. Analysis Phase

4.1 Introduction

This is where the determination and analysis of the business requirements are performed. This includes both a high level and broad analysis of business activity(s) and events as well as an analysis of the data those business activities require or produce. The scope of the analysis is either a specified business area or a well-defined information system project. The unit of work performed in this stage is known as a Business Area Analysis project.

Schematically, the structure of the steps in the analysis phase is:


Analysis 1
A1 - Acquire Initial Understanding
A2 - Produce Entity Relationship Diagram
A3 - Produce Process Model
A4 - Analyse Involvement Matrices
A5 - Confirm Outline Business Model
A6 - Analyse Distribution
A7 - Define Design Areas
A1.1 - Gather Information Sources
A1.2 - List Candidate Entity Types & Processes
A1.3 - Develop Initial Process Model
A1.4 - Develop Initial Entity Relationship Diagram
A1.5 - Assess Diagrams
A1.6 - Identify Knowledge Workers
A1.7 - Confirm Business Area Scope
A2.1 - Name and Define Entity Types
A2.2 - Name and Define Relationship Types
A2.3 - Name and Define Attribute Types
A2.4 - Examine Model for Entity Subtypes
A2.5 - Consider Merging Entity Types
A2.6 - Assess Entity Relationship Diagram
A3.1 - Decompose Processes
A3.2 - Produce Elementary Process Definitions
A3.3 - Identify the Business Events
A3.4 - Build Business Event Models
A3.5 - Build Process Logic Diagrams
A3.6 - Describe Logical Process Connectors
A3.7 - Assess Logical Process Models
A4.1 - Develop Information Needs Matrix
A4.2 - Develop Process/ Entity Type Matrix
A4.3 - Refine Business Area Model
A5.1 - Check Process Model Correctness
A5.2 - Check Entity Relational Model Correctness
A5.3 - Check Conformity to Rules and Conventions
A5.4 - Confirm Stability of the Outline Business Model
A5.5 - Verify Outline Business Model with Users
A5.6 - Refine Outline Business Model
A6.1 - Analyse Business Locations
A6.2 - Analyse Process Distribution
A6.3 - Analyse Entity Type Distribution
A6.4 - Estimate Process Frequencies
A6.5 - Estimate Entity Type Volumes
A7.1 - Establish Design Areas
A7.2 - esign Design Area Boundaries
A7.3 - Coordinate with Business Systems Architecture
A7.4 - Analyse Process Benefits
A7.5 - Select Process Mechanisms
A7.6 - Produce Cost-Benefit Analysis for Design Areas
A7.7 - Develop Design Area Project Plan
A7.8 - Prepare OBAA Stage Report


Analysis 2
A8 - Consider Purchased Software Solution
A8.1 - Survey for Packaged Software Solutions
A8.2 - Identify Criteria for Evaluating Software Solutions
A8.3 - Request Vendor Proposals
A8.4 - Evaluate Available Packaged Software Solutions
A8.5 - Authorise Packaged Software Solution
A8.6 - Negotiate Agreement with Package Vendor
A8.7 - Acquire Selected Packaged Solution
A8.8 - Reaffirm Path Selection
A9 - Analyse Entity Type Life Cycles
A9.1 - Create Entity State Transition Diagram
A9.2 - Assess Subtypes of Entity Types
A9.3 - Refine Entity State Transition Diagram
A9.4 - Define Classifying Attribute Types
A9.5 - Refine Business Model
A10 - Analyse Process Logic
A10.1 - Assess Process Documentation
A10.2 - Draw Process Logic Diagram
A10.3 - Prepare Process Action Diagram
A10.4 - Identify Attribute Value Actions
A10.5 - Define Algorithms
A10.6 - Analyse Exception Situations
A10.7 - Assess Process Action Diagrams
A10.8 - Refine Business Model
A11 - Analyse Business Rules
A11.1 - Define Domains
A11.2 - Define Attribute Type Properties
A11.3 - Define Permitted Ranges
A11.4 - Define Integrity Conditions
A11.5 - Define Security Requirements
A12 - Analyse Current Systems
A12.1 - Analyse Current Systems Procedures
A12.2 - Analyse Current Systems Data
A12.3 - Analyse Current Systems Problems
A12.4 - Compare Business Model with Current Systems
A13 - Confirm Correctness of Detailed Business Model
A13.1 - Check Consistency of Quantitative Information
A13.2 - Check Detailed Process Model Correctness
A13.3 - Check Entity Relationship Diagram Correctness
A13.4 - Check Conformity to Rules and Conventions
A13.5 - Correct Detailed Business Area Model
A13.6 - Confirm Business Model with Users
A14 - Reaffirm Design Area Evaluation
A14.1 - Assess Implementation Strategy
A14.2 - Assess Cost-Benefit Analysis

4.2 Steps in Analysis Stage

4.2.1 A1 - Acquire Initial Understanding

4.2.1.1 A1.1 - Gather Information Sources

An initial perspective on the business area has to be obtained to prepare the analysis of the business. The project team builds an initial model of the business area; additional information will be gathered from users later in the project. Strategic business information is obtained from the products of the Enterprise Modelling stage, if available. These may take the form of interview worksheets or material stored in the repository. Strategic business information will include a decomposition of the functions within the business area and an overview of the entity types that the functions involve.

Strategic business information should also include the following:

- Business objectives
- Business strategies
- Goals
- Critical success factors
- Performance measures
- Information needs

Business documents such as the annual report, business plans, organisation charts, job descriptions and operating manuals will provide an insight into the nature of the business area; these documents should be obtained and reviewed, then placed in the project's information library.

Another major source of information is the project team members' own knowledge of the business area, which will be drawn upon in subsequent subtasks.

4.2.1.2 A1.2 - List Candidate Entity Types and Processes

List candidate entity types

Seek to include entity types that are suggested by the project team from their knowledge of the business and that are not apparent in the existing documentation.

List candidate business processes

The names of candidate business processes are recorded in another list. These processes should be assigned comprehensive, descriptive names because no

definitions will be written for the processes. Just list the processes as you discover them, and, if it is appropriate, record synonyms for them.

4.2.1.3 A1.3 - Develop Initial Process Model

Decomposition Diagrams of Business Functions

Prepare one or more decomposition diagram(s), each of which is headed by one of the business functions defined in the project scope. The business function may decompose into several lower-order functions or it may decompose into processes. It is possible that lower-order functions identified during EM are now recognised as processes, so they would be classified, redefined and renamed accordingly.

Methods of Decomposition

The decomposition is achieved by a mixture of routes. For each function, consider what operational, control, planning and strategic processes will be needed to achieve the business objectives by the current strategies. Also consider what processes must exist in order to create each candidate entity type. At the same time, candidate processes identified in the earlier subtask List Candidate Entity Types and Processes need to be incorporated into the developing decomposition.

Level of Decomposition

It is essential to subdivide each business function into a set of processes - two levels deep. Decompose the processes as far as the team's understanding extends, without going below the elementary process level. The result can be documented as an indented list or by a decomposition diagram. This decomposition diagram will be used to confirm the project's scope.

This step creates the following deliverables:

- **Draft Process Decomposition Diagram** - see section 4.3.3.2 AD3.2 - Draft Process Decomposition Diagram on page 125
- **Process Definition** - see section 4.3.3.1 AD3.1 - Process Definition on page 125

4.2.1.4 A1.4 - Develop Initial Entity Relationship Diagram

Review the list of entity types found so far, check whether there has been any duplication or overlap, and adjust the list accordingly.

Name the entity types

For each entity type, write a definition and choose the best name. State at least one identifier for the entity type (this may be a composite identifier involving attribute types and relationship types). If an entity type is a subtype of another entity type in the list, record the name of the parent entity type against the subtype.

Construct an entity relationship diagram

Construct a diagram showing the entity types and relationship types found so far. By drawing the diagram, you may think of more entity types and relationship types. Do not wander outside the perceived scope or include redundant relationship types.

Name the relationship types

Always provide a name for the relationship types on the diagram. Relationship types have two names, one for each relationship membership.

Determine the cardinality and optionality of relationship memberships

Consider the cardinality of each relationship membership and indicate the cardinality of many or one on the diagram. Consider whether it is mandatory for the entity type to participate in the relationship membership; if it is not, indicate optionality on the diagram.

This step creates the following deliverables:

- **Entity Type Definition** - see section 4.3.2.1 AD2.1 - Entity Type Definition on page 123
- **Initial Entity Relationship Diagram**- see section 4.3.2.2 AD2.2 - Initial Entity Relationship Diagram on page 123

4.2.1.5 A1.5 - Entity Relationship Diagram

Entity relationship modelling enables us to describe data and its inherent structure. The entity relationship model is represented as a diagram, known as the entity relationship diagram. This diagram is used to show our understanding of data and is used in later stages of the Application Development Methodology as a basis for data structure design.

The topic of entity relationship modelling is described in more detail in section 14 Appendix 2 - Entity Relationship Modelling on page 346.

4.2.1.6 A1.6 - Assess Diagrams

The entire team reviews the diagrams and definitions and records improvements.

Cross-check the initial entity relationship diagram against the initial process decomposition diagrams - the nouns in the process names may reveal further entity types, subtypes, or synonyms.

Cross-check the process diagrams against the entity relationship diagram. This may suggest further, essential processes that can be added to the process diagrams. For each entity type, consider:

- How it is conceived
- How it comes into being
- How it changes
- How it is disposed of
- How it is monitored

4.2.1.7 A1.7 - Identify Knowledge Workers

Build a matrix of process performed by organisation units. Use the lowest level set of processes identified. Associate with the organisation units to identify who to interview during analysis stage.

This step creates the following deliverables:

- **Process/Organisation Unit Matrix** - see section 4.3.4.10 AD4.10 - Process/Organisation Unit Matrix on page 129

4.2.1.8 A1.8 - Confirm Business Area Scope

Project sponsor and project managers examine the matrix of process performed by organisation units to confirm which processes and organisation units are to be included within the scope of the analysis project.

This step updates the following deliverables:

- **Initial Entity Relationship Diagram** - see section 4.3.2.2 AD2.2 - Initial Entity Relationship Diagram on page 123

- **Draft Process Decomposition Diagram** - see section 4.3.3.2 AD3.2 - Draft Process Decomposition Diagram on page 125

4.2.2 A2 - Produce Entity Relationship Diagram

4.2.2.1 A2.1 - Name and Define Entity Types

Entity types are discovered during EM and during the Gather Information task. Many more potential entity types are discovered from interviews or user workshops. In this task, the entity relationship diagram is extended by naming all

of the new entity types and providing each with a formal definition. The names and definitions are prepared according to the standard ADM conventions.

In carrying out this task, you must check all entity types to confirm that the newest ones are separate from the ones identified earlier, and to ensure that all are properly named and defined.

You must also ensure that information needs expressed during EM or discovered while gathering information from users can be met by the entity types.

This step updates the following deliverables:

- **Entity Type Definition** - see section 4.3.2.1 AD2.1 - Entity Type Definition on page 123

- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123

4.2.2.2 A2.2 - Name and Define Relationship Types

Determine the relationship types between the entity types that you have discovered. Give each relationship membership a name that describes the reason why the entity types are related. Complete the remaining documentation for a relationship type by specifying the optionality, cardinality and permanence of each relationship membership.

This step updates the following deliverables:

- Entity Relationship Diagram

4.2.2.3 A2.3 - Name and Define Attribute Types

Some attribute types have been found in the Gather Information task:

- When classifying business objects
- When stating entity type identifiers

Other attribute types have been found in the Produce Process Model task:

- When discussing processes with users
- When identifying and describing information views

Ensure that the attribute type is named properly and is assigned to an entity type. Write a definition for the attribute type, unless its meaning is clear from its name.

New entity types may be identified when you attempt to assign the attribute type to an entity type. Examine the resulting attribute types of each entity type and remove synonyms. Replace any foreign attribute types with relationship types.

This step updates the following deliverables:

- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123

4.2.2.4 A2.4 - Examine Model for Entity Subtypes

Examine the attribute types, relationship types, and definition of each entity type and decide whether it would be useful to define any entity subtypes. An entity subtype must always have at least one attribute type and/or relationship type not shared by other entities of the same type. Wherever a subtype is proposed, it must be assigned to a partitioning; the partitioning must have a classifying predicate. If these have not been defined, define them now.

Record the subtype and the subtype's classifying values, and transfer the attribute types and relationship types from the entity type to the subtype. Note that this may cause relationship memberships to switch from optional to mandatory and may change the exclusivity of a relationship type.

The entity relationship diagram may already include subtypes. Review the justification for these subtypes and define their partitioning and classifying attribute values.

This step updates the following deliverables:

- **Entity Type Definition** - see section 4.3.2.1 AD2.1 - Entity Type Definition on page 123
- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123

4.2.2.5 A2.5 - Consider Merging Entity Types

Examine the entity relationship diagram to see whether any existing entity types could be merged into a single entity type. This can be a valuable way of rationalising the model in cases where two (or more) entity types have several corresponding relationship types and attribute types. The original entity types can then be defined as subtypes, if they are left with any special attribute types or relationship types of their own.

This step updates the following deliverables:

- **Entity Type Definition** - see section 4.3.2.1 AD2.1 - Entity Type Definition on page 123

- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123

4.2.2.6 A2.6 - Assess Entity Relationship Diagram

The entity relationship diagram is changed during the five previous subtasks, so you should check the structure of the diagram for consistency (e.g., to confirm that no redundant (transitive) relationships have been set up).

This step updates the following deliverables:

- **Entity Type Definition** - see section 4.3.2.1 AD2.1 - Entity Type Definition on page 123
- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123

4.2.3 A3 - Produce Process Model

This step and its sub-steps creates one or more models of the business activities included within the scope of the analysis project

4.2.3.1 A3.1 - Decompose Processes

As a result of discussions with the users, refine the initial process decomposition diagrams, make corrections and extend the diagrams to the elementary process level.

Formally apply the criteria for deciding whether a process is elementary to each process. Remember that an elementary process may not be decomposed into further elementary processes and that events may not trigger sub-elementary processes.

This step updates the following deliverables:

- **Process Decomposition Diagram** - see section 4.3.3.3 AD3.3 - Process Decomposition Diagram on page 125

4.2.3.2 A3.2 - Produce Elementary Process Definitions

Discover the essential, logical business activities within the defined project scope. Record the definition and purpose of each elementary process.

Confirm business activities with business end users.

After using the elementary processes in the business event models, document the number of times the process is expected to be performed in a given time period. Obtain this number directly from the user (it may be a "best guess"). During the Analyse Distribution task you will refine the number from statistics provided by computerised systems. Also record the periodicity and the time period (day, month, year) to which the periodicity relates.

This step creates the following deliverables:

- **Process Definition** - see section 4.3.3.1 AD3.1 - Process Definition on page 125

4.2.3.3 A3.3 - Identify the Business Events

Identify and define the business events included in the scope of the project scope.

Identify the business response(s) to be produced to "satisfy" each business event. The initiator of the event deserves to be responded to in a fashion that brings either satisfaction or closure to them.

This step creates the following deliverables:

- **Business Event List** - see section 4.3.3.5 AD3.5 - Business Event List on page 126

4.2.3.4 A3.4 - Build Business Event Models

Identify the elementary processes that are executed to produce the response to satisfy each business event. Each business event, its response and its associated business processes are called a theme.

Confirm business events, business responses and associated business processes with users.

This step creates the following deliverables:

- **Business Event Model** - see section 4.3.3.6 AD3.6 - Business Event Model on page 126

4.2.3.5 A3.5 - Build Process Logic Diagrams

There are two types of process logic diagrams - process dependency diagrams (PDD) and logical data flow diagrams (LDFD). They serve similar purposes. To perform this task, select either diagram type, but do not attempt both, as it would be largely a redundant exercise. The scope (context) of either type of diagram would be a theme (the set of processes that produce the response to a

business event. Therefore, there would be created a series of either PDDs or LDFDs containing all elementary processes that are within the scope of the area of analysis.

Draw process dependency diagram (PDD)

Develop a set of process dependency diagrams that show, between them, all elementary processes.

Start with a theme (event model) containing the processes that produce the response to a business event. This selected theme is the context for the PDD. Draw one dependency diagram for each theme. Make sure that every elementary process within that theme is included in the diagram.

Correlate the Decomposition Diagrams with the Dependency Diagrams

You will probably find errors in the decomposition diagrams while you are drawing the dependency diagrams. Correct the decomposition diagrams so that they remain consistent with the evolving dependency diagrams.

Verify the Consistency of Dependency Diagrams

Verify the consistency of these dependency diagrams by ensuring that every elementary process is contained in at least one PDD.

There will result a dependency diagram for each business event previously identified by the user community.

The dependency diagrams will be validated by the users who either perform or who are accountable for the performance of the processes represented in each diagram.

This step creates the following deliverables:

- **Process Dependency Diagram** - see section 4.3.3.4 AD3.4 - Process Dependency Diagram on page 126

Build logical data flow diagram (LDFD)

Develop a data flow diagram for each theme.

The scope or context of the data flow diagram is the set of elementary processes comprising a theme. The external agents reflect the business event that triggered the theme, the response to the business event and any other sources of information or recipients of information or resources of the processes included in the theme. Place the external agents on the same diagram as the elementary processes.

This step creates the following deliverables:

- **Logical Data Flow Diagram** - see section 4.3.3.7 AD3.7 - Logical Data Flow Diagram on page 126

4.2.3.6 A3.6 - Describe Logical Process Connectors

Both process dependency diagrams and logical data flow diagrams reflect their interaction with similar symbols. For process dependency diagrams, the connectors are referred to as information views. For logical data flow diagrams, the connecting symbols are referred to as data flows and data stores.

Developing process dependency diagrams

Information views are described and named to correspond with the dependencies existing among the elementary processes in the dependency diagrams.

Now formally document each information view. Assign each information view a unique, consistent name, and describe its contents in terms of entity types, attribute types or other information views.

Views of globally available data, not included in the information views that are imported for the elementary process, are also documented (although corresponding dependencies cannot appear on a dependency diagram). These views represent the "expected effects" on the pool of information (data model) the elementary process will have.

This step updates the following deliverables:

- **Process Dependency Diagram** - see section 4.3.3.4 AD3.4 - Process Dependency Diagram on page 126

Developing logical data flow diagrams

Add data flows to reflect the dependency of elementary processes upon other elementary processes for data or resources.

Add data stores where appropriate to reflect the "storage" of data in a different time frame than when it is to be utilised by an elementary process.

This step updates the following deliverables:

- **Logical Data Flow Diagram** - see section 4.3.3.7 AD3.7 - Logical Data Flow Diagram on page 126

4.2.3.7 A3.7 - Assess Logical Process Models

Evaluate the reasonableness, consistency and conformity to modelling standards of the process decomposition diagram, event models, data flow diagrams and process dependency diagrams.

This step updates the following deliverables:

- **Process Decomposition Diagram** - see section 4.3.3.3 AD3.3 - Process Decomposition Diagram on page 125
- **Process Definition** - see section 4.3.3.1 AD3.1 - Process Definition on page 125
- **Business Event** List- see section 4.3.3.5 AD3.5 - Business Event List on page 126
- **Business Event Model** - see section 4.3.3.6 AD3.6 - Business Event Model on page 126
- **Logical Data Flow Diagram** - see section 4.3.3.7 AD3.7 - Logical Data Flow Diagram on page 126
- **Process Dependency Diagram** - see section 4.3.3.4 AD3.4 - Process Dependency Diagram on page 126

4.2.4 A4 - Analyse Involvement Matrices

4.2.4.1 A4.1 - Develop Information Needs Matrix

Prepare the matrix. Include every information need associated with each function in the scope of the Business Area Analysis project. Include every information need identified during information gathering within the project. Experience shows that clearly defined information needs are really information views.

For each entity type, indicate the information needs to which it contributes. A matrix is the most straightforward technique to accomplish this.

For each information need, check that the entity types indicated are sufficient. If necessary, confirm this with the appropriate user and by reference to the lists of attribute types.

This step creates the following deliverables:

- **Information Needs Matrix** - see section 4.3.4.1 AD4.1 - Information Needs Matrix on page 127

4.2.4.2 A4.2 - Develop Process/Entity Type Matrix

Prepare a Process/Entity Type Matrix

Prepare the process/entity type matrix. You construct the matrix initially from the process descriptions and information view definition produced earlier on.

Check Each Entity Type in the Process/Entity Type Matrix

Sometimes processes are missing from the matrix because they are outside the scope of the analysis. If the create, update, read or delete entity action occurs in another business area, there is no requirement to supply a missing process.

For each entity type, check the following:

- There is at least one process that creates entities of this type. Ask yourself if all occurrences of the entity type can be created (e.g., it may be that the process only creates occurrences of a particular subtype).
- There is at least one process that moves entities of this type to the termination states.
- There is at least one process that updates entities of this type. It is conceivable that entities of some types will never need to be modified (e.g., entities with very short lives). Remember that the resulting business system must contain procedures that will facilitate the recording of any change to an attribute value.
- There is at least one process that reads the entities of this type. If not, why are you interested in information about this entity type?

Note All Missing Actions

Note all missing actions (or correct the matrix if the action is missing because of an error made by the author of the matrix).

This step creates the following deliverables:

- **Data Stewardship Assignment** - see section 4.3.2.5 AD2.5 - Data Stewardship Assignment on page 124

- **Process/Entity Type Matrix** - see section 4.3.4.2 AD4.2 - Process/Entity Type Matrix on page 127

4.2.4.3 A4.3 - Refine Business Area Model

Improve the business area model by including the missing elementary processes and actions.

There is a danger that inconsistencies will creep into the model during this subtask. Much of the work consists of correcting process dependency diagrams. The various models that have been built must be kept in synchronisation.

This step creates the following deliverables:

- **Outline Business Model** - see section 4.3.5.1 AD5.1 - Outline Business Model on page 129

4.2.5 A5 - Confirm Outline Business Model

4.2.5.1 A5.1 - Check Process Model Correctness

The following checks can be performed on the process model. Perform the checks only if requested to by quality assurance, if there is low confidence in the process model, or if automated tools are available.

Check consistency of process dependency diagrams

First check the process dependency diagrams that show processes at the elementary level.

- Does each elementary process appear on a process dependency diagram?
- Does every process have at least one input and at least one output information view?
- Does every process have a logical position in the sequence of processes?
- Does every process have a precondition and a unique result?

Search for duplicated processes

Find processes that are the same but that appear more than once in the dependency diagram, possibly under different names. This can happen in a large project when analysts work on different parts of the business area. If found, the two processes must be given the same name so that future modifications to one are also applied to the other. Unless you recognise that certain processes are essentially the same, you will waste effort in the design stages by designing two procedures (or more) for the same process.

You may find candidate duplicate processes by scanning the process/entity type matrix for processes that have the same set of actions upon the same entity types. Processes with identical sets of actions are not necessarily duplicates, so check the definitions of the processes, their preconditions and results to determine whether they are truly the same. If they are, choose a common name and make appropriate changes to the diagrams.

You can also search for processes that have the same (or similar) input and output information views.

This step creates the following deliverables:

- **Outline Business Model** - see section 4.3.5.1 AD5.1 - Outline Business Model on page 129

4.2.5.2 A5.2 - Check Entity Relational Model Correctness

This step is only undertaken if there is a lack of confidence in the entity relationship diagram or if it is demanded by quality assurance. It can normally be done concurrently with searching for redundant attribute types.

Check that each entity type is normalised

Experienced analysts will ensure, during analysis, that attribute types are correctly placed with entity types. However, in a project involving large numbers of analysts, inexperienced or careless analysts, it is worthwhile to use normalisation to check whether the attribute types have been assigned to the correct entity type. Normalisation is also worthwhile when there have been many modifications during the analysis of interactions or checking for completeness tasks.

Determine identifiers

Take an entity type and its attribute types and determine which attribute types are the identifier of the entity type. As a temporary measure, foreign identifiers are included as attribute types of the entity type.

Decompose the entity type into 4NF relations

A 4NF relation is a set of attribute types with the following characteristics:

- Some attribute types are designated as the "key"
- There are no repeating attribute values
- The non-key attribute types fully depend upon the key; they depend on the key, the whole key, and nothing but the key
- The non-key attribute types do not depend upon some other attribute type taking on a particular value in order for them to exist

By examining each attribute type of the entity type in turn, you will be able to group the attribute types directly into 4NF relations.

Apply the four normalisation steps to the attribute types

If it proves difficult to reach 4NF directly, apply the four normalisation steps to the attribute types by subdividing them.

Remember that each subdivided set of attribute types must have an identifier (consisting of one or more attribute types). Do not be surprised if some of the steps yield no subdivisions. The less subdivision, the better your entity relationship diagram is.

1. Subdivide the collection of attribute types to remove repeating groups and multi-valued attribute types. The attribute types are now in First Normal Form (1NF).
2. Subdivide again to remove attribute types that are only dependent upon a part of the identifier (i.e., not all the attribute types of a composite identifier). This is 2NF.

3. Subdivide again to remove attribute types that depend not upon the identifier, but upon some other attribute type. This is 3NF.
4. Subdivide again to remove attribute types that only exist when some other attribute type (or group of attribute types) takes on a certain value (or range).

This is 4NF and is the equivalent of decomposing into subtypes.

Examine the 4NF relations

Examine the resulting 4NF relations and decide whether to introduce new entity types, new subtypes, or new relationship types and whether to move any attribute types to other entity types.

Search for redundant attribute types

This step can be carried out at the same time as normalisation.

Review the attribute types of each entity type to ensure that there is no redundancy.

Two forms of redundancy:

- **Synonym** - when analysts have documented the same attribute type twice under two different names
- **Derivation** - when an attribute's values can be derived from the value(s) of some other attribute type(s), it should be classified as "derived," and an algorithm should be documented

Read all the attribute type definitions of one entity type to search for synonyms. When you find synonyms, decide which one to remove and whether any renaming is needed.

Check that the attribute type source (i.e., basic, derived, or designed) is correct. You may find that some derived attribute types are redundant.

Search for overlapping entity types

Review each entity type definition and check that no entity could belong to more than one entity type. If it can, there is a case for merging the two entity types or for specifying a relationship type between them.

At the same time, review the subtypes. Are they all necessary - i.e., does each subtype have some attribute types and/or relationship types of its own?

Search for redundant relationship types

Look for "circles" of relationship types on the entity relationship diagram; perhaps one of the relationship types will be redundant; i.e., can one of the relationship types be inferred from the others? The only way to answer this question is to think about the meaning of each relationship type and to consider the cardinality and optionality of the relationship types.

This step creates the following deliverables:

- **Outline Business Model** - see section 4.3.5.1 AD5.1 - Outline Business Model on page 129

4.2.5.3 A5.3 - Check Conformity to Rules and Conventions

Check that the deliverables of Outline Business Area Analysis conform to project standards and to the requirements of the enterprise.

Any diagrams and matrices drawn by hand must follow the project conventions or that are modified by the quality assurance standards function.

Check that the contents of the deliverables obey all the integrity rules determined thus far. This includes the following rules:

- All mandatory properties must exist for an object.
- All mandatory associations must exist for an object.
- Properties and associations must obey any integrity conditions that have been defined.
- Object names must conform to naming rules.

4.2.5.4 A5.4 - Confirm Stability of the Outline Business Model

Identify likely business changes

Some changes are based on common sense, and others are known from Enterprise Modelling (EM) or from interviews conducted during the initial and detailed analyses.

You may sometimes want to conduct further studies just for stability analysis. It may be that the enterprise has a policy unit that can be consulted. It will be valuable to interview the chief executive about changes expected in the next five to ten years. You can also look into other business areas.

Prepare a list of changes to the nature or conduct of the business that are likely to be introduced in the future. Here are the main categories to consider and some examples:

- New business activities
- Organisational changes
- Takeovers, mergers, and collaboration
- New legislation
- New procedures

Analyse impact of change

In the lists that follow, the changes have been stated broadly in the order of their degree of impact. Those at the top of the list have more impact on the entity relationship diagram and are thus the most costly to introduce to an operational system.

For each business change, consider the consequent changes to the following business models:

Entity relationship diagram

Based on the likelihood of the change and the degree of impact, you will have to decide how to modify the model to minimise the impact when any change occurs.

Making a previously mandatory relationship type optional or changing a "one" to a "many" is sometimes appropriate for many business changes. Take care that relationship types that were previously considered redundant are not now needed.

For example, if a business change is 80 percent likely within the next five years and involves changes to entity types or relationship types, it should be incorporated into the entity relationship diagram.

The following changes are listed according to their degree of impact upon the entity relationship diagram:

- Splitting one entity type into two entity types
- Merging two entity types
- Adding a new relationship type
- Removing a relationship type
- Removing an entity type
- Adding a new entity type
- Changing optionality and cardinality of a relationship type
- Moving an attribute type from one entity type to another
- Adding a new attribute type to an entity type
- Removing an attribute type
- Changing the optionality of an attribute type

Process Models

Examples of changes to the process models are:

- Adding a new function
- Adding a new elementary process
- Removing a process
- Changing the logic of a process
- Changing a dependency
- Changing an information view

Distribution Model

Examples of change to the distribution model are changes in:

- Location
- Process frequency
- Entity type volume
- Expected growth in process frequency and entity type volume
- Performance requirements

Build resilience into business model

Modify the business model enhance its resilience to changes in the business. Based on the likelihood of the change and the degree of impact, you will have to decide how to modify the model to minimise the impact when any change occurs.

Making a previously mandatory relationship type optional or changing a "one" to a "many" is sometimes appropriate for many business changes. Take care that relationship types that were previously considered redundant are not now needed.

For example, if a business change is 80 percent likely within the next five years and involves changes to entity types or relationship types, it should be incorporated into the entity relationship diagram.

Review completeness and correctness

After modifying the business area model, decide which subtasks of the Compare Business Model with Current Systems task, Analyse Involvement Matrices task, and Confirm Correctness task need to be repeated.

For example, if you have identified a new entity type, the process/entity type matrix will need to be altered and re-examined.

Prepare stability analysis report

Prepare a report that lists all likely business changes considered. This report can be prepared during the subtasks Identify Likely Business Changes, Analyse Impact of Change, and Build Resilience into Business Model.

An ability to cope with some business change is a useful input in the Define Design Areas task, and Evaluate Design Areas task.

For each business change, explain:

- Its impact on the business area model
- The likelihood of the change
- Changes made to the business area models to minimise the impact of the change

This step creates the following deliverables:

- **Stability Analysis Report** - see section 4.3.5.3 AD5.3 - Stability Analysis Report on page 129

This step updates the following deliverables:

- **Outline Business Model** - see section 4.3.5.1 AD5.1 - Outline Business Model on page 129

4.2.5.5 A5.5 - Verify Outline Business Model with Users

The following deliverables should be the subject of structured walkthroughs or facilitated sessions with users:

- Entity relationship diagram and entity type definitions
- Process decomposition diagram, process dependency diagrams, data flow diagram, business event models and elementary process definitions
- Stability analysis report

The more detailed diagrams should not be discussed during the workshop but are to be changed, where necessary, in the next task.

Error correction is not normally treated as a separate subtask. Correct errors as they are found in each subtask.

Review the nature and extent of any corrections to decide whether it is necessary to repeat any subtasks of this or any previous task, so that you can be confident that you have a complete and correct business model.

The quality assurance group will also inspect the deliverables and will initiate any further work to be done either by themselves or the project team.

This step updates the following deliverables:

- **Outline Business Model** - see section 4.3.5.1 AD5.1 - Outline Business Model on page 129

4.2.5.6 A5.6 - Refine Outline Business Model

A structured walkthrough will probably reveal some errors. You will have to decide whether they are significant enough to be corrected in the model. Do not imagine that the model will ever be perfect; many decisions are subjective. If modifications are made to the model, you must look through all the confirmation analysis steps (i.e., correctness checking and stability analysis) and decide which steps need to be repeated.

This step updates the following deliverables:

- **Outline Business Model** - see section 4.3.5.1 AD5.1 - Outline Business Model on page 129

4.2.6 A6 - Analyse Distribution

4.2.6.1 A6.1 - Analyse Business Locations

List the locations where processes in the business area are performed. Use the locations identified in the Business Systems Architecture as a starting point. Include locations external to the business, if processes are performed or planned to be performed there.

Identify the processes performed at locations and, if relevant, group similar locations to form location types.

Record the number of locations known to exist for each location type and obtain user estimates (business planning assumptions) of the expected growth of locations of each type.

This step creates the following deliverables:

- **Distribution of Data Usage** - see section 4.3.6.2 AD6.2 - Distribution of Data Usage on page 130

4.2.6.2 A6.2 - Analyse Process Distribution

Determine which elementary processes are performed at which location or location type and create the process distribution matrix. Do this either by determining where the equivalent procedure is performed or by obtaining user opinion on where it should be performed (if it is currently not done). Omit unused locations or location types from subsequent matrices.

This step updates the following deliverables:

- **Distribution of Business Processes** - see section 4.3.6.2 AD6.2 - Distribution of Data Usage on page 130

4.2.6.3 A6.3 - Analyse Entity Type Distribution

This subtask is performed if distributed data support for the business area is part of the technical architecture, and if refinement of the data store distribution is a critical input to the design area definition tasks.

For each entity type and location or location type, determine the locations of processes that create, read, update, and delete entities of the entity type; then create the entity type distribution matrix. Use the process/entity type involvement matrix and the process distribution matrix as starting points.

This step updates the following deliverables:

- **Distribution of Data Usage** - see section 4.3.6.2 AD6.2 - Distribution of Data Usage on page 130

4.2.6.4 A6.4 - Estimate Process Frequencies

Record or estimate the frequency of execution of each elementary process at each location or location type, using the process frequency matrix. Record the basis of any calculations and any assumptions made.

This step creates the following deliverables:

- **Process Frequency Matrix** - see section 4.3.6.3 AD6.3 - Process Frequency Matrix on page 130

4.2.6.5 A6.5 - Estimate Entity Type Volumes

Examine the process frequency matrix and the entity type distribution matrix. Use the entity creation and deletion frequency to record or estimate the volumes of entity types expected to be included in data stores at each location (or location type). Record the estimates in the entity type volume matrix, and record the basis of any calculations and any assumptions made.

This step creates the following deliverables:

- **Entity Type Volume Matrix** - see section 4.3.6.1 AD6.1 - Entity Type Volume Matrix on page 129

4.2.7 A7 - Define Design Areas

4.2.7.1 A7.1 - Establish Design Areas

Design areas are established by grouping processes and entity types to form coherent, cohesive units suitable to be the scope for a system and to be the subject of subsequent detailed analysis, design and implementation.
You can perform the initial grouping with cluster analysis. You then refine these groupings using other business area diagrams to clarify the exact boundaries.
Clustering objects on a matrix is carried out in the following manner:

Either perform steps 1 to 6:

1. Cluster the processes by using a mathematical cluster analysis algorithm and construct a matrix of similarity measures.
2. Cluster the entity types by using a mathematical cluster analysis algorithm and construct a matrix of similarity measures.
3. Apply cut-off(s) to the process and the entity type similarity matrices.
4. Identify the groups of objects after the cut-off.

5. Review the groups; if you are not satisfied, repeat from step 3 with a different cut-off.
6. Construct a process/entity type matrix with the order dictated by the clustering.

Or (instead of steps 1 to 6) apply manual clustering to the updated process/entity type matrix by moving rows and columns about in order to form groups of processes and entity types where no create/update is outside a group. A characteristic of the groups is that the processes should make use of the same broad set of entity types.

Then perform steps 7 to 9:

7. Manipulate the rows and columns of the clustered matrix so that the groups of processes and entity types form a diagonal across the matrix from the top left corner to the bottom right. The groups must be kept intact throughout this process, and the inclusion of create and update actions in the grouped diagonal should be maximised - ideally, there will be no ungrouped create or update actions.
8. Delineate the groups on the manipulated matrix.
9. Review the result; if you are not satisfied, review the clustered processes and entity types and the weightings, and repeat from the beginning.

This step creates the following deliverables:

- **Clustered Process/Entity Type Matrix** - see section 4.3.4.3 AD4.3 - Clustered Process/Entity Type Matrix on page 127

- **Design Area Definition** - see section 4.3.7.1 AD7.1 - Design Area Definition on page 130

4.2.7.2 A7.2 - Refine Design Area Boundaries

In this review, you check that the groupings of processes and entity types do not conflict with any of the logical structuring constraints in the process and entity relationship diagrams.

Examine process groupings

Good groupings will display certain characteristics; they will show:

- A low level of DEPENDENCE ON OTHER GROUPINGS
- A HIGH LEVEL of interdependence among the processes within the grouping

The characteristics of good groupings are best explored by examining the groupings in relation to decomposition and dependency diagrams. The decomposition diagram gives a first clue as to how closely the processes are related. The dependency diagram highlights how many dependencies cross boundaries between one grouping and another (the process/entity type matrix also does this, but less graphically). Inspect these diagrams to confirm whether

the design area process groupings are good and make good business sense (i.e., whether they are pragmatic).

Examine process distribution

Compare the elementary process groupings with the process distribution matrix to identify the location types where processes in each grouping are performed. The ideal grouping contains processes that are performed at a minimum number of location types. Review processes that have been considered for inclusion in different groupings. If a process is performed at a different location type from other processes in its group, consider how the pattern of distribution would be improved by moving the process to another grouping.

Distribution is not the principal criterion for grouping processes, but it may be possible to reduce the volume of communication between distributed data stores or computing facilities, or to group processes that support locations of the same type.

Examine data groupings

Examine the data groupings in relation to the entity relationship diagram. This can be done at two levels - first against the full diagram, then against a summarised diagram (one with approximately the same number of subject areas as clustered groupings).

The ideal boundary between groupings is one that cuts across fully optional relationship types only. Wherever there is a fully mandatory relationship type, both entity types must be placed in the same grouping.

Wherever a grouping differs from a subject area, a question is asked about the boundary - they need not correspond, but the reason why they do not should be understood before the design area boundary is fixed.

Usage statistics for relationship types should also be examined. A boundary that passes through a heavily used relationship type will be expensive to support.

Adjust the design areas

Adjust the design areas to accommodate any unacceptable constraints and repeat the earlier steps until a set of design areas that have acceptable data and process constraints between them is found (i.e., where the process dependencies are all acceptably distributed and there are acceptable data constraints). The inclusion or exclusion of the business locations and organisation units comprising the design areas may influence the boundaries of the design areas. Additionally, the lack of availability of enabling technology may influence the boundaries of the design areas.

This step updates the following deliverables:

- **Clustered Process/Entity Type Matrix** - see section 4.3.4.3 AD4.3 - Clustered Process/Entity Type Matrix on page 127

- **Design Area Definition** - see section 4.3.7.1 AD7.1 - Design Area Definition on page 130

4.2.7.3 A7.3 - Coordinate with Business Systems Architecture

Each proposed design area should correspond to a proposed business system in the enterprise model. Discrepancies between the proposed design areas and business systems need to be negotiated with staff members responsible for the enterprise model until an acceptable compromise is reached.

This step updates the following deliverables:

- **Clustered Process/Entity Type Matrix** - see section 4.3.4.3 AD4.3 - Clustered Process/Entity Type Matrix on page 127

- **Design Area Definition** - see section 4.3.7.1 AD7.1 - Design Area Definition on page 130

4.2.7.4 A7.4 - Analyse Process Benefits

The principal criterion used in determining whether a process should be offered computerised support is the benefit that might be expected to accrue from doing so. Benefit analysis may be carried out on each elementary process. However, in most instances it is preferable to carry out the benefit analysis on a group of highly dependent elementary processes. The outcome is a benefit value for each process or process group. Although it is preferable to place a monetary value on each benefit, it is often more practical at this stage to award each process or process group a benefit value score. A method of doing this is given in the cost-benefit selection technique.

Contributing factors to the process' score:

- Contribution to meeting business objectives
- Direct cost savings from computerising the current manual procedure supporting this process
- Direct cost savings from replacing the current computerised procedure
- Contribution to the solution of current system problems

The task can be used to aid establishing criteria for evaluating application software packages.

Activity Ranking

Rank the business activities (each elementary processor group of elementary processes), starting from the one with the greatest benefit and continuing to the activity with the least benefit.

In general, discard activities that display negative or dubious benefits from computerisation (i.e., that are not included in any of the design areas defined in the following steps), and consider the remainder selected.

Process Interdependence

The strong interdependence of processes will mean that in many cases one process cannot be computerised without computerising other processes needed to provide input information views. This factor should be taken into account; selection is not just a matter of drawing a line across the list of ranked activities. Any data dependency discrepancies not accounted for in this subtask will, however, be corrected by the analysis performed in the next subtask.

There will also be many instances where a process does not merit computer support but is essential to the design area as one of its supporting manual activities. Include it.

Benefit Analysis Using Entity Types

If the reason for conducting the Business Area Analysis project is to establish foundation databases, rather than conventional business systems, benefit analysis and selection may be conducted with entity types rather than processes.

This step creates the following deliverables:

- **Cost-Benefit Analysis** - see section 4.3.7.2 AD7.2 - Cost-Benefit Analysis on page 130

4.2.7.5 A7.5 - Select Process Mechanisms

The main criteria guiding this choice are the complexity, formality, frequency, intercommunication and shared use of data.

The feasible mechanisms for each elementary process should be considered and the preferred one chosen. At this stage, choose between a central or local processor and an alternate set of mechanisms:

- Manual
- Automated
- Not sure at this stage

4.2.7.6 A7.6 - Produce Cost-Benefit Analysis of Each Design Area

A cost-benefit analysis is performed for each design area (each proposed business system).

Cost Estimate

Estimate operational costs on a year-by-year basis for the next (n) years. Estimate the non-recoverable development costs as:

- A portion of the Enterprise Modelling costs
- A portion of the Outline Business Area Analysis costs

Depending on the chosen development path only as far as relevant, estimate the remaining development costs as:

- Detailed Business Area Analysis cost
- Business System Design cost
- Technical Design cost
- Construction cost
- Implementation costs (including data collection, conversion of files, bridging programs and user training)

Benefits Estimate

Using the cost-benefit analysis approach, described in the cost-benefit selection technique, place a monetary value on each of the benefits defined earlier for the business systems. It is better to give the benefit a roughly estimated monetary value than no value at all. As soon as it is written down, people can challenge it and a consensus will develop. Inevitably, some benefits cannot be quantified, so they will need to remain as qualitative rather than quantitative benefits in the final analysis.

Results of the Analysis

The results of this analysis may lead to a reappraisal of the processes selected and hence of the design areas and implementation sequence. The results will be reported to information systems and user management along with the outline subsequent projects schedule.

All calculations and methods used to derive each cost and each benefit should be retained. They will be helpful if during the task Review Design Area Evaluation adjustments to proposed design areas or to the implementation sequence have to be considered.

These working papers are also a useful reference document when you review costs and benefits at the end of subsequent development stages.

This step updates the following deliverables:

- **Cost-Benefit Analysis** - see section 4.3.7.2 AD7.2 - Cost-Benefit Analysis on page 130

4.2.7.7 A7.7 - Develop Design Area Project Plan

Prepare a schedule that shows:

- Each design project arising from the Outline Business Area Analysis project
- The process, data, business location and organisation unit composition of each design area
- Anticipated development path for each project
- Projected start and end date of each project

This step updates the following deliverables:

- **Subsequent Projects Schedule** - see section 4.3.7.3 AD7.3 - Subsequent Projects Schedule on page 131

4.2.7.8 A7.8 - Prepare OBAA Stage Report

The OBAA report is largely a compilation and summarisation of project deliverables and statistics. Introductory remarks, interpretations, and conclusions are added to relate the results to each other in a coherent manner.
A presentation of final conclusions, plans, and recommendations for action is prepared and made to the project's steering committee. Before the final presentation, all items must have been discussed and clarified with the committee members so that approval of subsequent actions can be obtained when the final presentation is made.

Once the final report is presented and the subsequent actions are approved, the OBAA project is complete.

This step creates the following deliverables:

- **BAA Report** - see section 4.3.9 AD9 - BAA Report on page 134

4.2.8 A8 - Consider Purchased Software Solution

4.2.8.1 A8.1 - Survey for Packaged Software Solutions

Investigate the market availability for packaged software solutions to the business need identified thus far in the analysis stage.

4.2.8.2 A8.2 - Identify Criteria for Evaluating Software Solutions

There will need to be two sets of criteria - one used for final packaged software selection and a coarser grained set used to select vendors and products to qualify for the "short list."

Identify the criteria to be used in evaluating and selecting a packaged software solution.

Determine criteria for evaluating products.

- Functional criteria

Determine the functional capabilities of the packaged software solution including which logical business processes it is intended to perform. The operator usability of the software solution should also be included in the criteria.

- Data compatibility criteria

Identify the content of batch input, screens and reports and identify the content and structure of databases that the product should accept, produce or maintain.

- Technology environment criteria

Determine the technology environment in which the packaged software solution must operate. Consider operating systems, hardware, network, middleware and database management systems. Consider existing standards for screens, forms and reports. The existence of a technology architecture or technology strategy will reduce dramatically the effort required for this task.

- Product references

How many product references should be verified? From what industries or technology environments should the references be drawn? What questions should be asked of the references?

Determine criteria for evaluating vendors.

- Financial viability

Define criteria for judging the vendor's projected growth and/or survivability. Determine the length of time that the vendor would need to remain viable to provide maintenance, customer support and product enhancements.

- Customer service

Define the nature and level of customer support expected of the chosen vendor for this system. Will this system require on-site help? What are their fees for on-site assistance? Or will telephone support be sufficient? What is their track record of providing support with other products?

Additionally, the degree and quality of vendor support required to both install and to operate the software solution should be included in the criteria.
The criteria should also address the experience they've had providing similar software solutions in the past.

- Maintenance

For how long will software maintenance be required? What is the vendor's track record of maintenance for this product? For other products?

- Vendor references

How many vendor references should be verified? From what industries or technology environments should the references be drawn? What questions should be asked of the references?

Cost

The criteria should include the direct fees to be paid for the software solution. It should also include the on-going cost for system operation, product maintenance and vendor support.

Identify method for selecting vendors to the "short list."

Vendors whose offerings make the "short list" will be afforded a thorough evaluation. Define the minimal criteria required for a vendor and their product to make the "short list." Consider whether the "short list" should be limited in size to ensure there are has adequate resources available for the thorough evaluation required.

4.2.8.3 A8.3 - Request Vendor Proposals

Identify vendors potentially capable of supplying the desired software solution. Prepare and submit to these vendors a request for their proposal to supply the desired software solution. The standard process for soliciting vendor proposals is to be followed.

4.2.8.4 A8.4 - Evaluate Available Packaged Software Solutions

Apply criteria established earlier to each of the proposed packaged software solutions and evaluate the findings.

Assess each packaged software solution against the pre-defined product criteria.

Does the packaged software solution possess enough of the business process functionality desired? Will the databases established and maintained by the packaged software solution be compatible with existing databases? Can data in the proposed databases be obtained from existing databases? To what degree will the proposed packaged software solution integrate with existing data and systems?

Will the packaged software solution operate in the current as well as planned technology environment? Consider requiring a demonstration of the product in the existing technology environment.

Assess the vendors against the pre-defined vendor criteria.

What is the vendor's track record of product enhancements, product maintenance and product support? Will this vendor be financially viable for the period that the user is dependent on them?

Check the references of both the products and the vendors.

Consider on-site visits as well as telephone checking of references.

Conduct a two-tier evaluation

Initially assess the vendor proposals using a coarse grain version of the evaluation criteria. The object is to eliminate from further consideration those vendor offerings that will definitely not satisfy the agreed evaluation criteria. This initial evaluation should be done quickly by assessing the proposals against only a few critical criteria. This first assessment will identify a "short list" of vendors. Apply the full set of criteria against all the proposals making the "short list." After a thorough assessment, identify the packaged software solution most suitable to satisfying the criteria previously established.

In the event that no packaged software solution is suitable, consider changing methodology paths or folding the project.

Advise vendors of the decision.

4.2.8.5 A8.5 - Authorise Packaged Software Solution

A company representative commits the company to proceed with the acquisition of the selected packaged software solution.

4.2.8.6 A8.6 - Negotiate Agreement with Packaged Software Solution Vendor

Negotiate business aspects of contract with the vendor of the selected packaged software solution. Consider a pilot operation of the packaged software solution in the company environment for a pre-determined period of time to be assured of the interoperability of the product in the company’s technology environment. Involve the company’s contracts office to ensure the company’s legal position is protected.

4.2.8.7 A8.7 - Acquire Selected Packaged Software Solution

Acquire the selected packaged software solution under the terms and conditions previously negotiated.

4.2.8.8 A8.8 - Reaffirm Path Selection

Reassess and reaffirm that the Methodology Path previously chosen remains appropriate for the project.

Overview of Detailed Business Area Analysis

Purpose of Detailed Business Area Analysis

The purpose of a DBAA project is to provide detailed models as a solid basis for system design. The methodology helps find the right answers to the right questions. Applying the methodology is never an end in itself.

The analysis must ultimately lead to decisions to do certain things, such as to start other projects. For the successful implementation of the decisions, it is important that the decisions be supported by the organisation as a whole. A DBAA project provides a framework for communication about things that are relevant to the decisions. During this communication process, conflicts in opinions and interests can be elucidated and resolved, and the overall commitment to the decisions that are made will thus be improved.

Objectives of Detailed Business Area Analysis

During the course of Detailed Business Area Analysis, the scope of the project is constantly refined. Detail is added only where there is a clear requirement for it from users. The level of detail should be enough to evaluate the definition of the design area, covering the scope of a business system. A subsequent design project covers such a design area.

The objectives of the Detailed Business Area Analysis Stage are to:

- Develop a detailed understanding of the design area
- Identify specific information needs and priorities for the business areas as a basis for system design
- Collect and analyse relevant current systems information assuring that the present functionality and problems are resolved in the design area models
- Evaluate the identification and prioritisation of the design area as to its support to the needs of the business area

General Approach to Detailed Business Area Analysis

Overview of Outline Business Area Analysis

An Outline Business Area Analysis (OBAA) project involves a more detailed analysis of entity types (data) and business processes (activities) than an enterprise modelling project. From an OBAA project, users can see precisely

what is relevant to them and can agree on plans for developing supporting systems.

The result of the OBAA is a full understanding of the business area, with identified information needs supported by relevant quantitative data as a basis for system design and distribution.

Finally, identification and prioritisation of systems to support the needs of the business area, delivers defined design areas, the scope for Detailed Business Area Analysis (DBAA) projects.

Results of Detailed Business Area Analysis

The result of the business modelling is a set of models that describe all that is relevant. The set can then be considered a "statement of requirements" or a "design area description." The major elements of this design area description are:

- A description of the scope of the area
- The detailed business area model, which itself consists of (1) an entity relationship diagram, including relevant business rules, (2) a process decomposition and a process dependency model, (3) the interactions between processes and entity types, and (4) process logic and process action diagrams for elementary processes and business algorithms
- An analysis of the current systems procedures and data stores supporting the design area, including a comparison report stating coverage and differences with the models
- The evaluated definition of the design area

Use in Later Phases of Development

Analysis at the level of a design area creates a base of information about the business on which to build one or more systems designed to serve the requirements of a collection of strongly related business activities. Because the design area is defined within the architecture of the business area, it is ensured that the resulting systems will be capable of being well integrated.

The DBAA project results in the evaluated definition and prioritisation of the design areas. The next stage would be the design of the system.

Planning for the Next Stage

For one design area, experience typically shows that one business system is identified for work in the next stage. Business systems are first identified in enterprise modelling stage and may be refined and re-scoped after completing the detailed business area model in the light of the more detail available at that time.

Role of Detailed Business Area Analysis

EM produces a general information architecture for the enterprise. OBAA leads to a detailing of the architecture for a particular area so that users can see precisely what is relevant to them and can agree on plans for developing systems to support the architecture. Then, DBAA fully describes in a rigorous way the detailed contents of a design area, being the conceptual basis for a business system.

Analysis lets people show just what their part of the enterprise looks like and what should happen within it. It highlights similarities and differences with other parts of the enterprise. It then becomes possible to see where difficulties in communication could arise and what the options are for resolving them.
An important feature is that the principal results from analysis can be represented diagrammatically and can be easily read and checked. The diagrams are used as a formal definition of the design area - i.e., they are the users' description of the way the business should be. The diagrams are valuable because they provide unambiguous instruction to the designers of information systems. The designers can translate the instructions directly into a specification for computer support.

Characteristics of Detailed Business Area Analysis

In addition to adhering to the general philosophy of the Application Development Methodology (ADM), DBAA has certain characteristics of its own:

- DBAA takes place within the framework of an EM and an OBAA and is driven by corporate objectives. It aims at producing a clear view, in business-oriented terms, of selected parts of the business, so that they can be given effective information system support.
- DBAA is geared to producing information system support for a whole design area, not to ad hoc solutions to short-term problems.
- DBAA focuses on the user's perception of the enterprise.
- DBAA provides a sound basis for estimating the broad costs and benefits of providing computer support through the implementation of high-quality information systems.
- DBAA requires greater initial effort and investment than traditional development approaches, but it leads to greater productivity and more effective, higher-quality systems.
- DBAA deliverables follow the concepts and definitions of the ADM and are used again in subsequent stages of the development process.

Critical Success Factors of Detailed Business Area Analysis

A number of critical factors require special attention during a DBAA project. To a large extent, these factors determine the success or failure of the project.

Top-Management Commitment

Though not requiring the same level of involvement by top management as EM, a DBAA project does require top-management commitment to providing the required resources. This can be particularly important for ensuring that

knowledgeable and appropriately skilled users are made available to participate in the project.

Furthermore, from the start of the project it should be possible to nominate an "owner" for the design area and its resulting system, giving an opportunity for very specific sponsoring of the project.

End-User Involvement

It is of prime importance to gain the commitment of senior managers whose world is to be analysed. During EM and OBAA, these managers' opinions helped define the design area and its priority for system support. If the project is to succeed, senior management must be seen to be backing it. It must be clear right from the start that the DBAA must lead to management decisions for subsequent actions.

It is critical to the success of this approach that the direction and priorities of the effort be established and controlled by users. A real commitment is now required in terms of access, involvement by user staff, participation in user workshops and access to all information relevant to the design area, including details of business objectives, plans, and procedures.

For this reason, resources for a DBAA project must always be drawn in part from the user community directly interested in and knowledgeable about the business requirements of the area in question. The most effective way to involve users is the use of user workshops in which the business is analysed and requirements are defined. These intensive group sessions are normally much more productive than interviewing. Reaching consensus is part of the workshop.

Correct Team Composition

Attention must be paid to ensuring that the project team members and user workshop participants have the required experience and capabilities. End users should provide business experience and specific knowledge of the design area under study. Analysts should be experts with techniques and should understand information systems. The overall direction should be controlled and coordinated through development coordination.

Right Duration

Projects involving analysis can become subject to "paralysis by analysis." Such projects are never successfully concluded, being constantly overtaken by changes in the business requirements. In general, this happens because analysts become too immersed in details that are irrelevant to the definition of the design area model. It is, however, important that the right level of detail be reached, because the next stage of development should be able to make a direct translation into a system design. The amount of time allocated to a DBAA project should be realistic, reflecting the number and complexity of business processes involved. Producing a detail design area model should generally not exceed two months. Using CASE tools for support and applying rapid development

techniques, like user workshops and timebox management, will reduce the lead-time.

4.2.9 A9 - Analyse Entity Type Life Cycles

4.2.9.1 A9.1 - Create Entity State Transition Diagram

Determine entity states

Consider how the entity type is first made known to the enterprise, and when it ceases to be of interest to the enterprise. Then concentrate on finding intermediate states. You will think of many states by asking what happens to an entity of this type and from having a general knowledge of the business. For example, the entity type EMPLOYEE might suggest the states Applied, Hired, Fired and Retired. The entity type ORDER might suggest the states Received, Accepted, Delivered, Paid and Archived.

To find more states, consider each elementary process. Use the following guidelines as clues for identifying states. Determine the:

- Creation states of the entity type
- Intermediate states of the entity type
- Termination states of the entity type

Set up the initial entity state transition diagram. Start construction by representing all states on the diagram. Creation state lines are placed at the top of the diagram, and termination state lines are placed at the bottom.

Identify entity state/process involvements:

Determine entity states and elementary processes

Decide which state changes are possible, and then determine the elementary processes that affect each state change and add them to the diagram. As a result, you may find elementary processes missing from the process model, process definitions that are inaccurate, and entity types that are not well defined. On the other hand, it may be the choice of entity states that needs adjustment. Entity states that do not appear to place any constraints upon processes are not useful as states.

Add processes to the diagram

Decide which processes involve the entity type but do not cause state changes. It may be impractical to fit all such processes into the diagram, so restrict the diagram to the significant processes. Add them to the diagram. Processes that involve the entity in almost every state do not add value to the diagram.
Add events to the diagram

If any process is triggered by an event, show the event on the diagram.

Guidelines for identifying entity state/process involvements:

- An entity may be in only one state at a time.
- The same process cannot result in alternative states unless an execution condition exists within the process and is defined on the diagram.
- The diagram will probably not be perfect the first time.

This step creates the following deliverables:

- **Entity State Diagram** - see section 4.3.4.7 AD4.7 - Entity State Diagram on page 128

- **Entity State Matrix** - see section 4.3.4.8 AD4.8 - Entity State Matrix on page 128

4.2.9.2 A9.2 - Assess Subtypes of Entity Types

If the entity type being analysed has subtypes, decide whether it would be more meaningful to draw separate life cycles for each subtype of a single partitioning. If you decided to analyse each subtype separately, decide now whether to merge all life cycles if they are identical or very similar.

This step updates the following deliverables:

- **Entity State Diagram** - see section 4.3.4.7 AD4.7 - Entity State Diagram on page 128

- **Entity State Matrix** - see section 4.3.4.8 AD4.8 - Entity State Matrix on page 128

4.2.9.3 A9.3 - Refine Entity State Transition Diagram

Refine the entity state diagram until it is satisfactory; check that each entity state is named correctly.

Repeat the subtasks Identify State/Process Involvements , Review Subtypes of Entity Type, and Refine Entity State Diagram until the diagram is satisfactory.

Review all elementary processes

The analysis of the life of an entity type is not complete until all elementary processes have been scanned to decide whether they involve the entity type. This can be done most effectively by selecting all processes that reference the entity type or its attribute types in their information views. After selecting the processes, ask the following questions:

- Does the process cause a change of state? (It must appear in the diagram.)
- Is the process restricted to one or a few states? (It should be represented in the diagram, as long as the diagram does not become too cluttered.)
- Is the process applicable to all or nearly all states? (It is not represented in the diagram but must appear in the state change matrix.)

This step updates the following deliverables:

- **Entity State Diagram** - see section 4.3.4.7 AD4.7 - Entity State Diagram on page 128

- **Entity State Matrix** - see section 4.3.4.8 AD4.8 - Entity State Matrix on page 128

4.2.9.4 A9.4 - Define Classifying Attribute Types

An attribute type, set of attribute types, or predicates, whose values can be used to determine which state the entity is in, must be identified. The easiest case is when the entity state can be determined from the value of a single attribute type. For example, BANK ACCOUNT may have an attribute type called Status, which indicates whether the account is Open, Closed or Suspended.

Entity states are sometimes determined from a combination of several attribute types: the values of the attribute types Start Date, End Date, and Dismissal Reason need to be inspected to determine whether an employee is in the Hired, Fired, or Retired State. Sometimes the state may depend upon the existence or non-existence of a relationship type, or upon a foreign attribute type. In all such cases, a new, derived attribute type called Status (prefixed by the entity type name) should be defined. An algorithm must also be defined that specifies how the Status attribute value is derived.

In other cases, there are no attribute types or relationship types that can be used to determine the states, so a new, non-derivable attribute type named Status must be defined. Each entity type in a life cycle should have an attribute type called Status.

This step updates the following deliverables:

- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123

4.2.9.5 A9.5 - Refine Business Model

Additions and improvements to the entity relationship diagram and process models will need to be made as a result of the life cycle analysis.

This step updates the following deliverables:

- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123
- **Process Definition** - see section 4.3.3.1 AD3.1 - Process Definition on page 125
- **Process Decomposition Diagram** - see section 4.3.3.3 AD3.3 - Process Decomposition Diagram on page 125
- **Process Dependency Diagram** - see section 4.3.3.4 AD3.4 - Process Dependency Diagram on page 126
- **Business Event Model** - see section 4.3.3.6 AD3.6 - Business Event Model on page 126
- **Logical Data Flow Diagram** - see section 4.3.3.7 AD3.7 - Logical Data Flow Diagram on page 126
- **Process/Entity Type Matrix** - see section 4.3.4.2 AD4.2 - Process/Entity Type Matrix on page 127
- **Clustered Process/Entity Type Matrix** - see section 4.3.4.3 AD4.3 - Clustered Process/Entity Type Matrix on page 127
- **Entity State Diagram** - see section 4.3.4.7 AD4.7 - Entity State Diagram on page 128

4.2.10 A10 - Analyse Process Logic

4.2.10.1 A10.1 - Assess Process Documentation

Examine existing process documentation

- Process name
- Process definition
- Import and export information views
- Preceding and succeeding processes
- Preceding event

Incomplete documentation

Is this documentation intelligible and consistent? Do you understand the purpose of the process? Do you know enough about it to perform the process logic analysis? If not:

- Contact appropriate personnel to clear up queries
- Revise the documentation

External import and export information views

Make sure that the external import and export information views have been included in the process dependency diagram - i.e., those originating from or being passed to external objects, as compared with those originating with or being taken up by other processes within the business area.

The external imports and exports must be shown on the process dependency diagrams, and their information view contents must be fully defined.

Prepare informal action list

Decide upon the actions needed

Look at an entity relationship diagram while considering the questions. Make note of the actions as you think of them. Remember, an action involves only entities of one entity type or pairings of one relationship type. Use the lowest-level entity subtype appropriate to the action.

Include the described checks in your analysis, but do not, at this stage, investigate which actions to perform when errors arise. Concentrate on identifying all the actions that occur when everything goes right.

Additional things to consider

Consider the following:

- Entities of which types are to be created, updated or deleted by the process?
- Pairings of which relationship types are to be associated or disassociated by the process?
- How are the entities and pairings involved in these actions selected: directly or by pairing?
- Existence checks for entity update, deletion or disassociation
- Non-existence checks for entity creation or association
- Whether integrity conditions are not violated by any of the actions (probably more integrity conditions will be identified during process action diagramming)

Number the actions

Number the actions to indicate the logical sequence in which they may occur (e.g., you cannot update an entity until you have selected it). If an entity has a foreign identifier, you cannot select it directly but must select some other entity first. The action number will show a feasible sequence of actions, although it is not necessarily the only possible sequence.

4.2.10.2 A10.2 - Draw Process Logic Diagram

The process logic diagram is drawn on top of (a subset of) the entity relationship diagram. Use the correct notation, include all the actions shown in your informal action list, and draw the diagram in the sequence indicated in your informal action list. Indicate on the diagram:
:

- Entity actions
- Pairing actions (previously called relationship actions)
- Selection conditions
- The action sequence

This step creates the following deliverables:

- **Process Logic Diagram**- see section 4.3.4.4 AD4.4 - Process Logic Diagram on page 127

4.2.10.3 A10.3 - Prepare Process Action Diagram

The logic of a process is now expressed in the form of a process action diagram. This is built by including all actions shown on the process logic diagram in the specified sequence. In addition, control statements, which control the execution of conditional groups of actions or repeating groups of actions, are inserted in the action diagram. Furthermore, selection conditions for entities of particular types are expressed formally in the action diagram.

This step creates the following deliverables:

- **Process Action Diagram** - see section 4.3.4.9 AD4.9 - Process Action Diagram on page 128

4.2.10.4 A10.4 - Identify Attribute Value Actions

This subtask is optional, depending on the defined required level of detail. Decompose each entity type action into attribute value actions that are added to the process action diagram. The attribute value actions include the following:

- CREATE entity decomposes into one or more SET attribute value actions.
- UPDATE entity decomposes into one or more SET or REMOVE actions.
- DELETE entity does not decompose.
- READ entity does not decompose.

This step creates the following deliverables:

- **Process/Attribute Type Matrix** - see section 4.3.4.5 AD4.5 - Process/Attribute Type Matrix on page 128

4.2.10.5 A10.5 - Define Algorithms

Business algorithms that are relevant to the business area must be defined during DBAA (e.g., the calculation of interest on bank accounts or of the order quantity in an ordering system, or of an automobile insurance policy premium). In principle it is necessary to define an algorithm for every derived attribute type (given the fact that on the analysis level the derived attribute types have a business value and are not of a design nature).

Define each algorithm used within the process, unless it has been defined previously, in which case the existing algorithm should be reviewed. Do this as an action block, using process action diagram syntax.

Algorithm names are formed by adding "Algorithm" to the name of the attribute type derived (e.g., Order Total Amount is calculated using Order Total Amount Algorithm). This is a general guideline, not a rule.

This step creates the following deliverables:

- **Business Algorithm** - see section 4.3.2.4 AD2.4 - Business Algorithm on page 124

4.2.10.6 A10.6 - Analyse Exception Situations

This subtask is optional, depending on the defined required level of detail. It involves identifying exception situations and deciding on the subsequent activities. Exception conditions and their subsequent activities are documented in the process action diagrams.

Principal Exception Situations

- Expected objects not found - An entity, pairing or attribute value may not be found.
- Unexpected object found - An unexpected entity, pairing or attribute value may be found
- Integrity conditions broken
- Execution conditions not satisfied

Subsequent Activities

Rather than specifying the full logic for dealing with errors, you can describe the business requirement in a NOTE.

In data processing systems, errors in the format of data (e.g., field must be numeric or field must be "O" or "1"). This type of data processing error is not considered in process logic analysis.

Examples of subsequent business activities are:

- Ignoring the exception situation
- Terminating the process and presenting the reason for termination

- Terminating the process and executing some other process (and possibly eventually returning to this process)
- Returning to an earlier part of the process, to find out whether the error recurs
- Special actions within the same process

This step updates the following deliverables:

- **Process Action Diagram** - see section 4.3.4.9 AD4.9 - Process Action Diagram on page 128

4.2.10.7 A10.7 - Assess Process Action Diagrams

This step is performed if authorised by the project manager, or if required by quality assurance. It is performed by quality assurance personnel or by a formal inspection meeting. Appropriate end users may be asked to attend the inspection.

The assessor checks that:

- All the deliverables have been produced
- All entity types, relationship types and attribute types appearing within the process logic diagram and process action diagrams are incorporated in the entity relationship diagram
- Process logic conforms with the process definition and other aspects of the process models
- Process logic models, including exception actions, are sensible and practical models of the business world (i.e., they are not just theoretically sound)
- Selection conditions, execution conditions, integrity conditions, and algorithms are properly documented
- The process really is elementary

This step updates the following deliverables:

- **Process Action Diagram** - see section 4.3.4.9 AD4.9 - Process Action Diagram on page 128

4.2.10.8 A10.8 - Refine Business Model

Process logic analysis may uncover omissions and inaccuracies in the business area model. Any of the objects documented during entity analysis or process analysis may be altered, added, or removed.

This step updates the following deliverables:

- **Entity Type Definition** - see section 4.3.2.1 AD2.1 - Entity Type Definition on page 123

- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123
- **Process Definition** - see section 4.3.3.1 AD3.1 - Process Definition on page 125
- **Process Decomposition Diagram** - see section 4.3.3.3 AD3.3 - Process Decomposition Diagram on page 125
- **Process Dependency Diagram** - see section 4.3.3.4 AD3.4 - Process Dependency Diagram on page 126
- **Process Action Diagram** - see section 4.3.4.9 AD4.9 - Process Action Diagram on page 128
- **Business Event Model** - see section 4.3.3.6 AD3.6 - Business Event Model on page 126
- **Process/Entity Type Matrix** - see section 4.3.4.2 AD4.2 - Process/Entity Type Matrix on page 127
- **Process Logic Diagram** - see section 4.3.4.4 AD4.4 - Process Logic Diagram on page 127
- **Process/Attribute Type Matrix** - see section 4.3.4.5 AD4.5 - Process/Attribute Type Matrix on page 128
- **Process/Relationship Type Matrix** - see section 4.3.4.6 AD4.6 - Process/Relationship Type Matrix on page 128
- **Entity State Diagram** - see section 4.3.4.7 AD4.7 - Entity State Diagram on page 128
- **Entity State Matrix** - see section 4.3.4.8 AD4.8 - Entity State Matrix on page 128

This step creates the following deliverables:

- **Detailed Business Model** - see section 4.3.5.2 AD5.2 - Detailed Business Model on page 129

4.2.11 A11 - Analyse Business Rules

4.2.11.1 A11.1 - Define Domains

An initial set of domains should be defined. Ideally, the domains should be defined by the data administrator, and the same set should be used in all BAA

projects. New domains would be added only if an attribute type arises that is not covered by any of the existing domains.

If a domain is restricted to particular values (e.g., YES, NO), this is recorded as permitted ranges.

If there is a conversion algorithm between domains in the same hierarchy (e.g., from barrels to litres), this should be recorded.

This step updates the following deliverables:

- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123

4.2.11.2 A11.2 - Define Attribute Type Properties

The remaining properties of each attribute type are recorded. For derived attribute types, record the derivation algorithm.

This step updates the following deliverables:

- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123

4.2.11.3 A11.3 - Define Permitted Ranges

Note that an attribute type must always be assigned to a domain.

Defining Permitted Ranges for Attribute Values

When it is known that an attribute value is restricted to a particular range or ranges of values over and above that specified by its domain (e.g., Pupil Age = 4 through 19; Age domain has values 0 through 999) this information is recorded as a permitted range related to the attribute type. When it is known that an attribute type is restricted to specific values, each value is recorded (as a permitted range).

Defining Sub-domains

If the same permitted ranges constrain more than one attribute type, consider defining a sub-domain.

Defining Codes as Range Values

Where well-established codes are in use within the business (and the codes must represent single attribute types), the codes and their meanings should be recorded as permitted range values.

This step updates the following deliverables:

- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123

4.2.11.4 A11.4 - Define Integrity Conditions

Inspect each relationship type

Inspect each relationship type and determine whether any integrity conditions need to be defined. The following types of integrity condition should be sought:

- Conditions governing exclusive relationship types; these should have been indicated on the entity relationship diagram
- Conditions governing combinations of relationship type exclusivity and inclusivity
- Conditions governing when an optional relationship pairing should (or should not) exist

Inspect each optional partitioning

For each optional partitioning, an integrity condition is defined.

Inspect the attribute types of each entity type

Inspect the attribute types of each entity type and search for any integrity conditions governing the attribute values. It is not necessary to find every single inter-attribute type integrity condition at this stage. Just document what you are aware of so far.

This step updates the following deliverables:

- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123

This step create the following deliverables:

- **Data Security Requirements** - see section 4.3.2.6 AD2.6 - Data Security Requirements on page 124

4.2.11.5 A11.5 - Define Security Requirements

The security requirements of the data and processes associated with the business area are defined.

4.2.12 A12 - Analyse Current Systems

4.2.12.1 A12.1 - Analyse Current Systems Procedures

Identify current systems procedures

This subtask identifies current procedures associated with each of the systems within the scope of this Detailed Business Area Analysis project. Existing system documentation is examined, and the procedures are identified.

Collect information for each procedure

- Inputs and outputs (data flows)
- Sources and destinations of these data flows

Construct an indented procedure list

For each business system studied, construct an indented procedure list (or hierarchy diagram) that shows the system, subsystems, and procedures. When there are a number of different sources of documentation covering the same or overlapping procedures, the analyst should draw a rough procedure list for each source. The interfaces and overlaps between the different lists should then be identifiable. In particular, pay attention to interfaces between manual and computer procedures. Once these interfaces are resolved, the lists can be merged and unique procedures can be identified.

Use current system documentation

Inspection of current system documentation may often need to be augmented by fact-finding interviews or observation. This applies when current system documentation is not available or is of poor quality, out of date, or otherwise unreliable. It also applies when problems and difficulties with the current procedures are known to exist, have not been documented recently, and need to be restated. Problems identified at this time should be added to the problems and issues list; they will be analysed later in the Analyse Current Systems Problems task.

Produce low-level data flow diagrams

Develop data flow diagrams. Draw one diagram per system or subsystem, and include all the procedures shown on your procedure list. Take care to ensure that the interfacing flows between different diagrams are named consistently.

Produce higher-level data flow diagrams

After procedures have been recorded in system or subsystem data flow diagrams, the diagrams should be synthesised into a system dependency diagram for the business area. This task flushes out any inconsistencies in data flow naming and allows lower-level diagrams to be amended accordingly.
The result is a diagram comparable to a subset of the Business Systems Architecture, and it provides a coherent view of system support to the business area.

This step create the following deliverables:

- **Physical Data Flow Diagram** - see section 4.3.8.1 AD8.1 - Physical Data Flow Diagram on page 132
- **Problem Report** - see section 4.3.8.2 AD8.2 - Problem Report on page 133
- **Procedure List** - see section 4.3.8.3 AD8.3 - Procedure List on page 133

4.2.12.2 A12.2 - Analyse Current Systems Data

Gather user view sources

First select the layouts to be analysed. It is not necessary to analyse every layout. Try to select a sample that covers the full range of data used by the system; the master files or database must be analysed, plus all heavily used screen layouts and reports. Only include layout forms and lightly used layouts if they appear to contain further basic implied attribute types, or if they are considered to be documents of major importance by end users.

Each layout is examined, and its fields and keys are highlighted for contrast. Objects such as page numbers, processing dates, and line sequence number should be omitted unless they are vital to the purpose of the system (e.g., if they identify another field).

Develop initial data analysis diagrams

Draw first versions of each data analysis diagram by working downward from the top left of each layout. At this stage, do not attempt to check that all the rules and guidelines have been obeyed, or to optimise the shape of the diagram.

Refine data analysis diagrams

Once you have drawn the initial diagrams, produce second versions that obey the rules and follow the guidelines, including the "hierarchical downwards" convention. Check that the names of the same field on different charts are consistent, and note any anomalies on the problems and issues list.

Perform canonical synthesis

Canonical synthesis is best performed with automated tools. When you perform the synthesis manually, you develop a key-oriented list by analysing each data analysis diagram in turn.

If difficulties arise during the synthesis (e.g., because of possible synonyms), record them on the problems and issues list and refer them to appropriate users.

Analysis of First Data Analysis Diagram

For each key:

1. Create an entry in the key-oriented list that corresponds to the key and enter an appropriate name for the implied entity type.
2. Create an entry in the index.
3. Create an entry for each non-key field that is dependent on the key in this view.

For each multiple association:

1. If joining two keys, enter an association membership against both the key groups - 1:M for the key at the multiple end of the association, and 1:1 for the other.
2. If a non-key field points to a key, it belongs to the key group as a secondary key.
3. If a key has a multiple association with a non-key field, add the non-key field as a temporary identifier. It will probably be a key in subsequent views. If the non-key field has no other fields dependent on it, it will be changed in the key-oriented list to become a multi-valued field corresponding to the key field.

Analysis of Subsequent Data Analysis Diagrams

For each key:

1. Check if it is already represented as a key in the index. If not, proceed as above; then check that it does not already appear in some list as a non-identifying field. If it does, follow the guidelines for building key-oriented lists.
2. Add any new non-key fields that are dependent on this key in this diagram to the existing list against the key group identified by the key. If any of the fields are already in the index as keys, follow the guidelines for building key-oriented lists.
3. If the key is already included in a composite key, add a 1:M association membership from the key field to the composite key, and a 1:1 association membership from the composite key to the new key.

For each multiple association:

1. If joining two keys, add the association between the key groups if it does not already exist. If it results in a M:M (many to many) association, follow the guidelines for building key-oriented lists.
2. For other cases, proceed as in 1 above, if the details are not already recorded.

Review implied entity types

Perform the following checks:
- Non-identifying fields should be 1:1 with keys.
- No fields listed as keys should be left as non-identifying.

Checks to Perform

No fields should be repeated in more than one key group.

No fields, especially non-key fields, should be repeated in more than one key group; if they are, follow the guidelines for building key-oriented lists.

There should be no transitive associations.

This involves following the chains of associations. Follow the guidelines for building key-oriented lists.

Add key groups with no non-identifying fields to other key groups.

If any key group has no non-identifying fields but has associations with other key groups, add it to these; otherwise, remove it. If a field is added to more than one key group, follow the guidelines for building key-oriented lists.

When the checks are complete

Once these checks are complete, each key group entry in the key-oriented list can be taken to be an implied entity type. Its key and non-key fields become implied attribute types, and the association memberships between key group entries become implied relationship types.

Construct implied entity relationship diagram

Draw up the implied entity relationship diagram from the key-oriented list, using the entity relationship diagram conventions. Optionality of relationship types can be added if they are evident at the time the diagram is drawn up.

This step create the following deliverables:

- **Problem Report** - see section 4.3.8.2 AD8.2 - Problem Report on page 133
- **Data Analysis Diagram** - see section 4.3.8.4 AD8.4 - Data Analysis Diagram on page 133
- **Implied Entity Relationship Diagram** - see section 4.3.8.5 AD8.5 - Implied Entity Relationship Diagram on page 133
- **Key-Oriented List** - see section 4.3.8.6 AD8.6 - Key-Oriented List on page 133

4.2.12.3 A12.3 - Analyse Current Systems Problems

Perform problem analysis

Break the problem down into its root causes

Very often, the problem as perceived by the user and as reported to the analyst is merely a symptom of one or more fundamental problems. It may be necessary

to break these perceived problems down through several levels of decomposition to arrive at the root cause or causes. This involves researching the area of the problem by:

- Reviewing available documentation and statistics
- Conducting short fact-finding interviews with associated management and staff

Determine the impact of the problem

How is the performance of dependent procedures affected? Estimate the cost to the organisation of any adverse effect. Check your assumptions with the management of the affected area.

Use cause/effect diagrams

Use cause/effect diagrams to help you find the root causes and eventual effects.

Prepare current systems problem analysis report

To prepare the current systems problem analysis report follow the steps below:

Classify the problem

Classify the problem as one of the following:

- **Information Problem**: This should relate to the usefulness of the information provided by existing systems.
- **System Problem**: This should relate to the speed, accuracy, completeness or timeliness of work performed by current systems.
- **Technical Problem**: This should relate to inadequate technical facilities that serve the system.
- **Organisational Problem**: This should be an organisation weakness within the enterprise that directly affects the business area being studied.

Organise the problem analysis report

Organise the problem analysis report according to the classifications given to the problems.

Prioritise the problems by order of significance of the estimated cost to the organisation. However, lower-estimated cost items should be given high priority if a worsening situation is causing urgency.

Record identified solutions.

Review the problem analysis

Review the problem analysis with business managers responsible for overseeing the BAA project. This is a good time to elicit management's views on alternative

solutions to the problems. Take care to record acceptable solutions in detail, together with the name of the manager proposing the solution.

This step create the following deliverables:

- **Problem Report** - see section 4.3.8.2 AD8.2 - Problem Report on page 133
- **Problem Analysis Report** - see section 4.3.8.7 AD8.7 - Problem Analysis Report on page 134

4.2.12.4 A12.4 - Compare Business Model with Current Systems

Compare ER model with implied ER model

Note any objects (entity types, relationship types, or attribute types) missing from the business area model.

- For each implied entity type in the current systems model, find a corresponding entity type in the business area model.
- For each implied relationship type in the current systems model, find a corresponding relationship type in the business area model.
- For each implied attribute type in the current systems model, find a corresponding attribute type in the business area model.

Find corresponding objects

If there is no corresponding object in the business area model, decide whether this represents a missing object.

- The current systems model may contain objects that exist only because of the design of the current system.
- Different names may have been used in the two models for the same object, or the same name may have been used for two different objects.
- The current systems model may artificially merge two different objects into one object.
- Comparing the two requires thinking about the meaning of objects - not just about their names.

Perform the comparison the opposite way around

Find out whether the current systems lack any objects included in the business area model. Note any weaknesses in the current systems that are apparent from the comparison.

Compare PDDs with DFDs of current systems

For each data flow in the current systems data flow diagram, find the corresponding process dependency. Also compare data views with information views.

The two models are not exactly comparable, but dependency should tie in with data flow. This comparison is laborious because it involves tracing all similar paths between the two sets of diagrams.

Note any missing elementary processes, dependencies, or attribute types; note any weaknesses in the current systems apparent from the comparison.

Compare processes with current systems procedures

For each procedure in the current systems procedure list, find a corresponding process in the business area model.

There is not always a 1:1 correspondence, because existing systems procedures may have been built to suit the available hardware and software, or because they may have been designed by mirroring previous clerical procedures rather than by being based on the underlying business processes. Some procedures exist only for housekeeping and error-correction and will therefore not need any corresponding process.

This comparison can also be performed the opposite way around, to find processes that have no corresponding procedure. Decide whether this is a weakness in the current system or is merely caused by the scope of the current system analysis.

Refine business model

Review the missing objects noted during the preceding subtasks and update the list, removing inconsistencies and duplications. Then update the business area model.

This is not a trivial task, because the addition of a new object may propagate other additions to the model, and there is a danger that inconsistencies will creep in.

For example, when a new elementary process is added, process logic analysis will need to be performed for that process. This will involve defining actions, algorithms, selection conditions, and execution conditions for the process. The entity state diagrams and matrices will need to be updated. A new event may have been identified. Process dependencies will need to be revised, and information views defined. The process hierarchy diagram will also require revision.

Prepare comparison check report

The differences between models, together with weaknesses and problems in the current systems noted during the preceding tasks, are listed in the comparison check report.

The problems of the current systems, and their causes and effects, have been recorded in the problem analysis report as business problems. Any problems identified in analysing the current systems that have a solution identifiable through the business area models should also be noted in the problem analysis report.

This step create the following deliverables:

- **Comparison Check Report** - see section 4.3.8.8 AD8.8 - Comparison Check Report on page 134

4.2.13 A13 - Confirm Correctness of Detailed Business Model

4.2.13.1 A13.1 - Check Consistency of Quantitative Information

This step uses volume and frequency estimates to check the correctness of structure and to cross-check the estimates themselves. This step is sometimes called cardinality checking. Following is a list of some of the checks that can be performed.

Subtype Volumes

The number of entities of one type must equal the total number of entities in all the subtypes of any single partitioning of that entity type.

Relationship Type Cardinality

When two entity types are associated, the entity type volume figures should be consistent with the cardinality of the relationship type associating them.

Process Dependency Cardinality

This check corresponds to the previous one but uses dependencies and processes instead.

Action Frequency

Create and delete action frequencies should be checked against entity volume figures and growth estimates.
For example, say the volume estimate for CUSTOMER is 100 and the action frequencies are 9 customers created a month, and 1 customer deleted a year. This is not consistent because you appear to be gaining 107 customers per year. It would be reasonably consistent however, if the analysts had predicted 100% growth per year.

This step updates the following deliverables:

- **Detailed Business Model** - see section 4.3.5.2 AD5.2 - Detailed Business Model on page 129

4.2.13.2 A13.2 - Check Detailed Process Model Correctness

In this subtask several checks on the process model are described. Perform the checks only if requested to by quality assurance, if there is low confidence in the process model, or if automated tools are available.

The following checks can be performed:

Check consistency of elementary process dependency diagrams

First check the process dependency diagrams that show processes at the elementary level.

- Does each elementary process appear on a process dependency diagram?
- Does every process have at least one input and at least one output information view?
- Does every information view have a source? This must be a process or - an external object
- Does every information view have a destination? This must be another process or an external object

Check hierarchy of process dependency diagrams

If you have produced a hierarchy of process dependency diagrams, check that the net input and output of views at level (n) correspond to the net input and output of the combination for level (n+1) diagrams.

Search for duplicated processes

Find processes that are the same but that appear more than once in the hierarchy (or the dependency diagram), possibly under different names. This can happen in a large project when analysts work on different parts of the business area. The two processes must be given the same name so that future modifications to one are also applied to the other.

Unless you recognise that certain processes are essentially the same, you will waste effort in the design stages by designing two procedures (or more) for the same process.

Scan the process/entity type matrix

Candidate duplicate processes can be found by scanning the process/entity type matrix for processes that have the same set of actions upon the same entity types. Processes with identical sets of actions are not necessarily duplicates, so check the process logic analysis and the definitions of the processes to determine

whether they are the same. If they are, choose a common name and make appropriate changes to the diagrams.

You can also search for processes that have the same (or similar) input and output information views.

When two processes are similar but not identical, the chances are that some of their subprocesses are the same.

This step updates the following deliverables:

- **Detailed Business Model** - see section 4.3.5.2 AD5.2 - Detailed Business Model on page 129

4.2.13.3 A13.3 - Check Entity Relationship Diagram Correctness

In this subtask several checks on the entity relationship diagram are described. Perform the checks only if requested to by quality assurance, if there is low confidence in the process model, or if automated tools are available.
The following checks can be performed:

Check that each Entity Type is Normalised

Experienced analysts will ensure, during analysis, that attribute types are correctly placed with entity types. However, in a project involving large numbers of analysts or inexperienced analysts, it is worthwhile to use normalisation to check whether the attribute types have been assigned to the correct entity type. Normalisation is also worthwhile when there have been many modifications during the analysis of interactions or checking for completeness tasks.

This step is therefore only undertaken if there is a lack of confidence in the entity relationship diagram or if it is demanded by quality assurance. It can normally be done concurrently with searching for redundant attribute types.

Determine identifiers

Take an entity type and its attribute types and determine which attribute types are the identifier of the entity type. As a temporary measure, foreign identifiers are included as attribute types of the entity type.

Decompose the entity type into 4NF relations

A 4NF relation is a set of attribute types with the following characteristics:

- Some attribute types are designated as the "key."
- There are no repeating attribute values.
- The non-key attribute types fully depend upon the key; they depend on the key, the whole key, and nothing but the key.

- The non-key attribute types do not depend upon some other attribute type taking on a particular value in order for them to exist.
- By examining each attribute type of the entity type in turn, you will be able to group the attribute types directly into 4NF relations.

Apply the four normalisation steps to the attribute types

If it proves difficult to reach 4NF directly, apply the four normalisation steps to the attribute types.

Remember that each subdivided set of attribute types must have an identifier (consisting of one or more attribute types). Do not be surprised if some of the steps yield no subdivisions. The less subdivision, the better your entity relationship diagram is.

1. Subdivide the collection of attribute types to remove repeating groups and multi-valued attribute types. The attribute types are now in First Normal Form (1NF).
2. Subdivide again to remove attribute types that are only dependent upon a part of the identifier (i.e., not all the attribute types of a composite identifier). This is 2NF.
3. Subdivide again to remove attribute types that depend not upon the identifier, but upon some other attribute type. This is 3NF.
4. Subdivide again to remove attribute types that only exist when some other attribute type (or group of attribute types) takes on a certain value (or range).

This is 4NF and is the equivalent of decomposing into subtypes.

Examine the 4NF relations

Examine the resulting 4NF relations and decide whether to introduce new entity types, new subtypes, or new relationship types and whether to move any attribute types to other entity types.

Search for redundant attribute types

Review the attribute types of each entity type to ensure that there is no redundancy.

Read all the attribute type definitions of one entity type to search for synonyms. When you find synonyms, decide which one to remove and whether any renaming is needed.

Check that the attribute type source (i.e., basic, derived, or designed) is correct. You may find that some derived attribute types are redundant (e.g., Age can be derived from Birth Date; why is Age needed if Birth Date is available?). Ensure that an algorithm exists for each derived attribute type (if the project manager has decided that all algorithms are to be defined during the BAA stage).
This step can be carried out at the same time as normalisation.

Two forms of redundancy

- Synonym: Analysts have documented the same attribute type twice under two different names
- Derivation: When an attribute's values can be derived from the value(s) of some other attribute type(s), it should be classified as "derived," and an algorithm should be documented

Search for overlapping entity types

Review each entity type definition and check that no entity could belong to more than one entity type. If it can, there is a case for merging the two entity types or for specifying a relationship type.

At the same time, review the subtypes. Are they all necessary (i.e., does each subtype have some attribute types and/or relationship types of its own?)?

Search for redundant relationship types

Look for "circles" of relationships on the entity relationship diagram; perhaps one of the relationships will be redundant (i.e., can one of the relationships be derived from the others?). The only way to answer this question is to think about the meaning of each relationship and to consider the cardinality and optionality of the relationships. There is no formal technique for doing this.

Check process/relationship type matrix

If automated facilities do not exist to produce the process/attribute type matrices, this laborious subtask should be omitted.

Prepare a process/relationship type matrix

The process/relationship type matrix is constructed from information in the process action or process logic diagrams.

Check each relationship type in the process/relationship type matrix

For each relationship type, check the following:

- There is at least one process that creates pairings for this relationship type (associate action).
- There is at least one process that terminates pairings for this relationship type (disassociate action).
- To find missing disassociate actions, try looking at the processes that delete the associated entities. When an entity is deleted, all of its pairings must be disassociated at the same time. A relationship type associating two indefinite termination entity types may not need a disassociate action.
- There is at least one process that selects entity types by using pairings of this relationship type.
- It is possible that actions are missing from the matrix because they are outside the scope of the analysis. If associate, disassociate, and select-via-

pairing actions occur in other business areas, there is no requirement to supply the missing action.

Note all missing actions

Note all missing actions (or correct the prepared matrix if the action is missing because of an error made by the author of the matrix).

Check process/attribute type matrices

If automated facilities do not exist to produce the process/attribute type matrices, this laborious subtask should be omitted.

Prepare a process/attribute type matrix

The process/attribute type matrix can be prepared from the contents of the process action diagrams or from the information views identified during process dependency analysis.

Check each attribute type in the process/attribute type matrix

It is permissible for actions to be missing from the matrix because they are outside the scope of the analysis. If set and remove actions and references occur in other business areas, there is no requirement to supply them. For each attribute type, check the following:

- There is a process that sets (creates or recreates (modifies)) the attribute values.
- There is a process that removes (deletes or nullifies) optional attribute values.
- Optional attribute values that may change to "no-value" (e.g., Employee Home Telephone Number) require a remove action.
- Mandatory attribute types must not have a remove action. Deleting an entity removes all its attribute values, and, by convention, it is not necessary to document the removal actions that accompany entity deletion.
- There is a process that references the attribute values.
- Each attribute type should be referenced within a selection condition, execution condition, or algorithm, or because it is reported or presented by a process. (Note that referencing is not defined as a formal action but happens in the course of other actions.)

Note all missing actions

Note all missing actions (or correct the prepared matrix if the action is missing because of an error made by the author of the matrix).

This step updates the following deliverables:

- **Detailed Business Model** - see section 4.3.5.2 AD5.2 - Detailed Business Model on page 129

4.2.13.4 A13.4 - Check Conformity to Rules and Conventions

This task should be performed by the quality assurance function.

The deliverables of BAA are checked to ensure that they conform to project standards and to the rules and conventions described in this hypermedia product. Any diagrams and matrices drawn by hand must follow the conventions that are laid out in this hypermedia product or that are modified by the information management standards function.

The contents of the documentation are checked to ensure that all the integrity rules are obeyed. This includes the following:

- All mandatory properties must exist for an object.
- All mandatory associations must exist for an object.
- Properties and associations must obey any integrity conditions that have been defined.
- Object names must conform to naming rules.

This step updates the following deliverables:

- **Detailed Business Model** - see section 4.3.5.2 AD5.2 - Detailed Business Model on page 129

4.2.13.5 A13.5 - Correct Detailed Business Area Model

Errors found in the other subtasks of this task are corrected. This is not normally treated as a separate subtask. Errors are corrected as they are found in each subtask.

The nature and extent of any corrections are reviewed to decide whether it is necessary to repeat any subtasks of this or any previous task, so that you can be confident that you have a complete and correct business model.
The quality assurance group will also inspect the deliverables and will initiate any further work to be done either by themselves or the project team.

This step updates the following deliverables:

- **Detailed Business Model** - see section 4.3.5.2 AD5.2 - Detailed Business Model on page 129

4.2.13.6 A13.6 - Confirm Business Model with Users

The following deliverables should be the subject of structured walkthroughs with users:

- Entity relationship diagram and entity type definitions
- Information needs entity type matrix

- Process hierarchy diagram and elementary process definitions
- Comparison check report
- Stability analysis report
- The more detailed diagrams should not be discussed during the workshop but are to be changed when necessary in the next subtask

A structured walkthrough will probably reveal some errors. You will have to decide whether they are significant enough to be incorporated in the model. Do not imagine that the model will ever be perfect - many decisions are subjective. If modifications are made to the model, you must look through all the confirmation analysis steps (comparison checking, involvement matrix checking, correctness checking and stability analysis) and decide which steps need to be repeated.

This step updates the following deliverables:

- **Detailed Business Model** - see section 4.3.5.2 AD5.2 - Detailed Business Model on page 129

4.2.14 A14 - Reaffirm Design Area Evaluation

4.2.14.1 A14.1 - Assess Implementation Strategy

The global implementation aspects for the design area were defined during Outline Business Area Analysis. Now having analysed the design areas to full detail, the earlier statements should be reassessed.

This step updates the following deliverables:

- **Implementation Strategy** - see section 4.3.7.4 AD7.4 - Implementation Strategy on page 132

4.2.14.2 A14.2 - Assess Cost-Benefit Analysis

Cost Estimate

The earlier defined estimated operational costs on a year-by-year basis for the next(n) years are to be corrected and enhanced.

Estimate the non-recoverable development costs

The non-recoverable development costs are estimated as:

- A portion of the Enterprise Modelling costs
- A portion of the Outline Business Area Analysis costs
- The Detail Business Area Analysis costs

Estimate the remaining development costs

The remaining development costs are estimated as (depending on the chosen development path only as far as relevant):

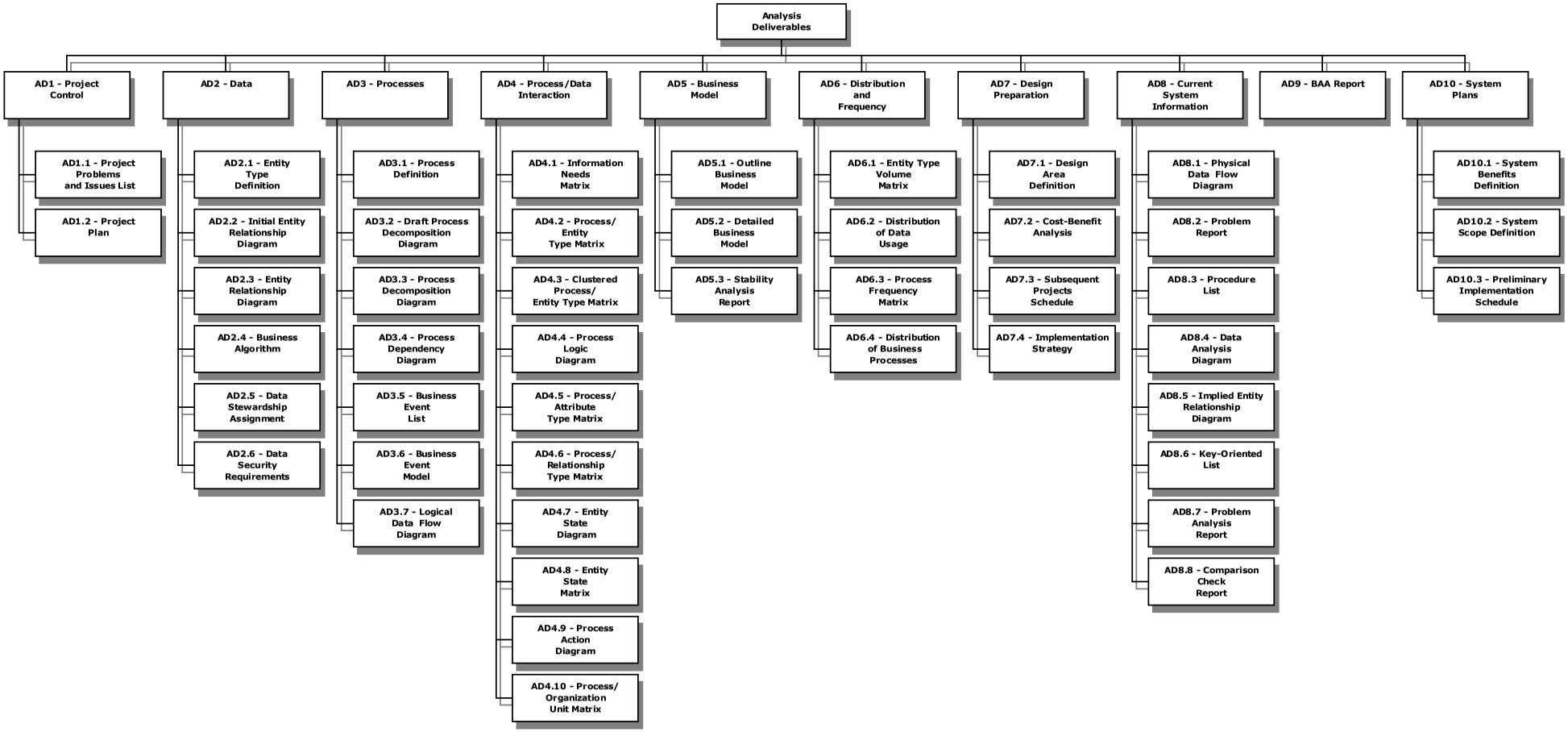
- Business System Design cost
- Technical Design cost
- Construction cost
- Implementation costs (including data collection, conversion of files, bridging programs, user training)

Benefits Estimate

Using the cost-benefit selection technique, place a monetary value on each of the benefits defined earlier for the business system. It is better to give the benefit a roughly estimated monetary value than no value at all. As soon as it is written down, people can challenge it and a consensus will develop. Inevitably, some benefits cannot be quantified, so they will need to remain as qualitative rather than quantitative benefits in the final

4.3 Analysis Phase Deliverables

Schematically, the structure of the deliverables in the analysis phase is:


Analysis Deliverables
AD1 - Project Control
AD1.1 - Project Problems and Issues List
AD1.2 - Project Plan
AD2 - Data
AD2.1 - Entity Type Definition
AD2.2 - Initial Entity Relationship Diagram
AD2.3 - Entity Relationship Diagram
AD2.4 - Business Algorithm
AD2.5 - Data Stewardship Assignment
AD2.6 - Data Security Requirements
AD3 - Processes
AD3.1 - Process Definition
AD3.2 - Draft Process Decomposition Diagram
AD3.3 - Process Decomposition Diagram
AD3.4 - Process Dependency Diagram
AD3.5 - Business Event List
AD3.6 - Business Event Model
AD3.7 - Logical Data Flow Diagram
AD4 - Process/Data Interaction
AD4.1 - Information Needs Matrix
AD4.2 - Process/ Entity Type Matrix
AD4.3 - Clustered Process/ Entity Type Matrix
AD4.4 - Process Logic Diagram
AD4.5 - Process/ Attribute Type Matrix
AD4.6 - Process/ Relationship Type Matrix
AD4.7 - Entity State Diagram
AD4.8 - Entity State Matrix
AD4.9 - Process Action Diagram
AD4.10 - Process/ Organization Unit Matrix
AD5 - Business Model
AD5.1 - Outline Business Model
AD5.2 - Detailed Business Model
AD5.3 - Stability Analysis Report
AD6 - Distribution and Frequency
AD6.1 - Entity Type Volume Matrix
AD6.2 - Distribution of Data Usage
AD6.3 - Process Frequency Matrix
AD6.4 - Distribution of Business Processes
AD7 - Design Preparation
AD7.1 - Design Area Definition
AD7.2 - Cost-Benefit Analysis
AD7.3 - Subsequent Projects Schedule
AD7.4 - Implementation Strategy
AD8 - Current System Information
AD8.1 - Physical Data Flow Diagram
AD8.2 - Problem Report
AD8.3 - Procedure List
AD8.4 - Data Analysis Diagram
AD8.5 - Implied Entity Relationship Diagram
AD8.6 - Key-Oriented List
AD8.7 - Problem Analysis Report
AD8.8 - Comparison Check Report
AD9 - BAA Report
AD10 - System Plans
AD10.1 - System Benefits Definition
AD10.2 - System Scope Definition
AD10.3 - Preliminary Implementation Schedule

4.3.1 AD1 - Project Control

4.3.1.1 AD1.1 - Project Problems and Issues List

A work list of open issues and identified problems, which, depending on their content, must be solved within the next task, before the next milestone, or before the project can be completed.

4.3.1.2 AD1.2 - Project Plan

The overall plan for the project management components identified, created, monitored and controlled during the project's life cycle. The project plan is, in practice, a prerequisite for every ADM task.

The project plan should contain the following elements:

Definition Elements

- Project Approach
- Project Calendar
- Project Constraints
- Project Contract
- Project Development Environment
- Project Objectives
- Project Organisation
- Project Scope
- Project Standards
- Project Work Breakdown Structure

Resourcing Elements

- Project Costs
- Project Milestones
- Project Resource Claims
- Project Resource Release
- Project Resource Type Estimation
- Project Schedule
- Project Start Date
- Operations Elements
- Project Completion Estimation Report
- Project Corrective Actions List
- Project Deliverables Status Report
- Project Earned Value Report
- Project Metrics
- Project Problems and Issues List
- Project Resource Usage Report
- Project Risk Memo

- Project Time Report

Miscellaneous Elements

- Project Profile
- Project Promotional Material

4.3.2 AD2 - Data

4.3.2.1 AD2.1 - Entity Type Definition

The formal definition and description of an entity type and its properties. For complex situations, an entity life cycle subtype hierarchy may also be included.

Properties That Can Be Recorded

- Entity type name
- Identifier(s)
- Definition
- Synonym(s) (if any)
- Attribute types
- Parent entity type name (if this is a subtype)

4.3.2.2 AD2.2 - Initial Entity Relationship Diagram

The diagram for a business area, conforming to all the conventions for this type of diagram, and showing only the detail that is immediately available.

The initial ERD can include:

- Entity type names
- Relationship names
- Relationship cardinality
- Relationship optionality
- Entity subtypes and partitionings
- Exclusive relationships

4.3.2.3 AD2.3 - Entity Relationship Diagram

The Entity Relationship Diagram defines entity and relationship concepts, their documentation and various guidelines and rules concerning the development of an entity relationship model.

Entity relationship modelling enables us to describe data and its inherent structure. The entity relationship model is represented as a diagram, known as the entity relationship diagram. This diagram is used to show our understanding

of data and is used in later stages of the Application Development Methodology as a basis for data structure design.

4.3.2.4 AD2.4 - Business Algorithm

A set of methods or instructions, including detailed specification of calculations, computations, and/or deductions, for the derivation or modification of values of an attribute type.

Typically, there are many methods for validating domains, converting values from one subset to another, and deriving attributes. These methods are called business algorithms. Attribute values may be calculated or deduced within a process, so a business algorithm that explains how the attribute values are derived must be specified. You are, however, concerned only with what needs to be done to derive, validate, or convert an attribute, not with how it is done. If there is a standard, unchanging algorithm already defined, this should be used; otherwise, the algorithm must be defined. Business algorithms can be documented in a variety of ways. For example, algorithms may be documented as a process action diagram or as a decision table. They are defined either for a domain or as a derivation of an attribute.

4.3.2.5 AD2.5 - Data Stewardship Assignment

Data stewards are accountable (answerable) for the quality of the user's business data. They are usually assigned responsibility for the validity and accuracy of data of one or more entity types.

The deliverable may contain all or part of the following elements:

- Identify data stewardship responsibility for the data created, updated or deleted within the scope of this project.
- Data stewards are selected.

4.3.2.6 AD2.6 - Data Security Requirements

To define the business requirements for constraining access to selected business data as well as for ensuring the privacy of selected business data.

Data Privacy Requirements.

Controlled Data Access Needs.

- Defines which business roles have the right to create or change the values of enterprise data.
- Defines which business roles have the right to retrieve the values of enterprise data.

4.3.3 AD3 - Processes

4.3.3.1 AD3.1 - Process Definition

The formal definition and description of the purpose and logical content of a business process. Properties that can be recorded:

- Process name
- Synonym(s) (if any)
- Parent process or function

4.3.3.2 AD3.2 - Draft Process Decomposition Diagram

There will be one decomposition diagram for each business function listed in the scope, so the topmost activity is always a business function. This may be decomposed into further business functions or processes (not both at the same level). The lowest-level business activities must be processes. At this time, they need not be elementary processes, and they should not be more detailed than elementary processes.

4.3.3.3 AD3.3 - Process Decomposition Diagram

A representation of the breakdown of processes into progressively increasing detail.

The decomposition of business activities is represented with a process hierarchy diagram. The structure is a hierarchy. During Business Area Analysis, functions should break down into processes, and the processes should break down into progressively more detailed processes. The process decomposition diagram is created by adding substance and depth to the draft process decomposition diagram.

Processes with More Than One Parent

Occasionally, in a decomposition, you find some processes that are part of (i.e., are used in) more than one higher-level process. You are therefore not producing a strict hierarchy, but a network where some processes can have more than one parent. It is useful to identify this in order to produce a realistic picture of a business area. It also reduces the redundancy in your model, because the process needs to be defined and decomposed only once.
Duplicate the process wherever it appears in the decomposition diagram (i.e., always draw a pure hierarchy) and denote the process as duplicated to avoid confusion. If a duplicated process is decomposed, the decomposition is drawn only once.

4.3.3.4 AD3.4 - Process Dependency Diagram

A diagram that shows why, for each process, an execution may depend upon the prior execution of other processes. The Process Dependency Diagram depicts the dependencies existing between the elementary business processes (as defined in the Process Decomposition Diagram). At minimum a dependency diagram is prepared for each process that lies immediately above elementary process level. Diagrams can be made of higher levels for overview purposes.

The diagram shows:

- Elementary process
- Information views
- Dependencies
- External objects

4.3.3.5 AD3.5 - Business Event List

A list of the business events - both external and temporal - that initiate elementary business processes in the business area. The list also includes a description of the response(s) required to satisfy each business event.

4.3.3.6 AD3.6 - Business Event Model

An association of elementary business processes and the business event that triggers them. This model is also called a theme. The elementary business processes associated with each business event are those required to produce the response to the event.

4.3.3.7 AD3.7 - Logical Data Flow Diagram

A diagram that shows the data object types output by one activity and later used by another activity.

A logical data flow diagram is a representation of the flow of data into, out of, and between procedures, subsystems or systems. The diagram is similar to the process dependency diagram.

Process dependency diagrams show the dependencies that exist for business processes and do not indicate how the dependencies may be manifested. Data flow diagrams show the flow of actual data between designed activities and logical data stores, thus depicting the manner in which two business activities interact.

4.3.4 AD4 - Process/Data Interaction

4.3.4.1 AD4.1 - Information Needs Matrix

A matrix that shows, for each information need associated with a business area, which of the entity types of the business area are used in satisfying the need.

4.3.4.2 AD4.2 - Process/Entity Type Matrix

The Process/Entity Matrix represents the nature of the action of the process upon an entity type, (relationship) pairing or attribute type. For each process, a subset of this matrix is the Expected Effects Matrix.

4.3.4.3 AD4.3 - Clustered Process/Entity Type Matrix

When clustered (using a clustering algorithm), the process/ entity type matrix shows groups of closely related processes and entity types that may be systems or subsystems.

To create the matrix, affinity analysis is performed that leads to an intermediate deliverable, an Affinity Matrix.

4.3.4.4 AD4.4 - Process Logic Diagram

A diagram that shows the inherent logic of a process that is created by annotating a subset of the entity relationship diagram.

The diagram depicts:

- The entity types and relationships used
- Actions upon entities
- A possible sequence of execution

A process logic diagram depicts the actions of an elementary process upon entities of specified entity types and pairings of specified relationships, giving a possible sequence for their execution in that process. The diagram appears as a navigation of the process through the entity relationship diagram. Process logic diagrams are drawn to ensure that the entity relationship diagram can support the information needs of elementary processes and to provide a skeleton for developing more detailed logic.

The diagram expresses, at the "type" level, what can happen at the "occurrence" level. It therefore consolidates for one process the various actions possible on any execution of that process.

4.3.4.5 AD4.5 - Process/Attribute Type Matrix

A matrix that shows, for attribute types, the processes in which they are involved and the operative actions (set value, retrieve value and remove value).

4.3.4.6 AD4.6 - Process/Relationship Type Matrix

A matrix that shows, for relationship types, the processes in which they are involved and the operative actions.

4.3.4.7 AD4.7 - Entity State Diagram

A representation of an entity type life cycle. It depicts all the possible states in a life cycle of entities of one type, and the processes that cause changes in their states. Processes that use the entity type but do not change it may also be included in the diagram if space permits.

Entity States and Process Dependencies (BAA)

Processes are often dependent on one another, because a precondition for one is established by the other. The condition will be an entity or entities in a particular state. Proper knowledge of entity states is therefore essential to the construction of dependency diagrams. Conversely, process dependency diagrams provide invaluable information to the builder of entity state diagrams.

4.3.4.8 AD4.8 - Entity State Matrix

A tabular representation of an entity type life cycle. It shows which processes are valid for each state applicable to the entities of one type. All processes that act upon the entity type are included: both the processes that cause state changes and those that do not.

4.3.4.9 AD4.9 - Process Action Diagram

A representation of the logic of a process in terms of the actions carried out on each entity analysis object involved and the conditions constraining these actions. The degree of detail specified in the diagram can be varied to suit the nature of the problem, the objectives of the project and the availability of automated tools.

Process Action Diagrams in Process to Procedure Mapping (BSD)

A process action diagram describes, in its entirety, the inherent logic of an elementary process defined in Business Area Analysis, including attribute actions and exception processing. During process to procedure mapping, the association

is established between a process and procedures. During procedure action diagramming, that association is examined to determine how the logic can be used within the procedure. The process action diagram conventions are similar to those for a procedure action diagram.

4.3.4.10 AD4.10 - Process/Organisation Unit Matrix

An association identifying which business processes are performed by which organisation units. This matrix identifies who to contact for knowledge of the business processes.

4.3.5 AD5 - Business Model

4.3.5.1 AD5.1 - Outline Business Model

The business model for a business area resulting from the outline business area analysis phase of an analysis project. It contains all elements of a business model including process models, business event models, entity-relationship diagrams, and matrices built using the objects in these models. However, the outline business model does not include all business rules.

4.3.5.2 AD5.2 - Detailed Business Model

A portion of an outline business area model (for a design area) to which the business rules have been added.

The refined detailed business model for a design area that has been checked against the current systems and data stores.

The complete detailed business model that has been checked for correctness. This business model represents all the individual models built and maintained in the analysis stage.

4.3.5.3 AD5.3 - Stability Analysis Report

A document that identifies potential future business changes, their expected impact on the business area model and necessary modifications to the model.

4.3.6 AD6 - Distribution and Frequency

4.3.6.1 AD6.1 - Entity Type Volume Matrix

A tabular representation of the presence and volume of storage of each entity type at each location. The intermediate format is the Entity Type Distribution Matrix.

4.3.6.2 AD6.2 - Distribution of Data Usage

A matrix of entity types and business locations is created. The association is "entity type is used at business location." For those interactions that are valid, the cell contains an "X."

4.3.6.3 AD6.3 - Process Frequency Matrix

A tabular representation of the presence and frequency of executions of each process at each location. The intermediate format is a Process Distribution Matrix.

4.3.6.4 AD6.4 - Distribution of Business Processes

An association of business processes and business locations. The association is "business process is performed at business location." In each cell representing a valid interaction, an "X" is placed.

4.3.7 AD7 - Design Preparation

4.3.7.1 AD7.1 - Design Area Definition

The description of the contents of a design area, consisting of a global description and detailed information about the processes and entity types included in the scope.

For all of the processes and entity types included in the design area, include the:

- Process Definition
- Process Mechanism
- Process Dependency Diagram
- Entity Type Description
- Entity Relationship Diagram
- Process/Entity Type Matrix

4.3.7.2 AD7.2 - Cost-Benefit Analysis

A parameter-based estimate of the total cost to develop and cutover the proposed area, based on the present definition of the areas scope, against the

quantitative estimate of benefits that could be achieved by the area. The analysis can be done for several scopes, depending on the task involved. Typical scopes are per business area, design area and/or procedure.

Costs accrue from the implementation of an activity or entity type. As with benefits, costs are either tangible or intangible. Tangible costs are directly caused by the implementation of an activity or entity type and can be attached to a direct monetary value.

Other tangible costs include things like training and implementation. Intangible costs are often more important than tangible costs. They include things like disruption to the enterprise and consequential loss of customer service, and non-availability of staff. Accurate costing of systems development from analysis information is generally not possible. Even so, estimates are essential. Most organisations have rules of thumb accumulated by experience, but development costs can change significantly for many reasons, including:

- The programmer or designer employed
- The software in use
- Database size
- Database complexity
- Transaction volumes
- Transaction complexity
- Experience with the technology to be used

Components of the Cost-Benefit Analysis

- Cost-Benefit Summary Report
- Benefit Analysis Matrix
- Benefit Analysis Table

4.3.7.3 AD7.3 - Subsequent Projects Schedule

The suggested schedule of subsequent projects to follow the present development stage.

The schedule shows:

- Each project arising from the project
- Development path for each project
- Suggested start and end dates for each project

Classification of Projects

- System development projects
- Technical projects
- Organisational projects
- Implementation projects

4.3.7.4 AD7.4 - Implementation Strategy

The strategy describes the general approach that will be adopted during the implementation of systems, based on a desire to minimise dependencies or interfacing requirements, cost and organisational change. The document identifies the key considerations on which the strategy is based, and defines the types of implementation (big-bang, shadow, fade-out, etc.) that will be employed. The strategy will in a later stage be the input for the Implementation Plan.

For each implementation strategy, the following elements are considered:

General requirements

The way business systems and data stores are dealt with in the strategy e.g., the level of automation, the use of standard packages, and the sharing of systems and data stores throughout the enterprise.

The use of technology

The level of sophistication, the speed of implementation of technical components, and the overall business impact (the competitive edge) of the technical solution. This part of the strategy relates to the technical architecture options of the task Define Technical Architecture.

The organisational impact

The required changes in the IM organisation, the special staffing and educational needs for this strategy and the impact on the overall organisation's tasks, structures, and required policies.

Transition from existing systems

A global view is given on the order and speed of the new systems analysis and development.

4.3.8 AD8 - Current System Information

4.3.8.1 AD8.1 - Physical Data Flow Diagram

A diagram that shows the data object types output by one activity and later used by another activity.

A Physical data flow diagram is a representation of the flow of data into, out of, and between procedures, subsystems or systems. The diagram is similar to the process dependency diagram.

Process dependency diagrams show the dependencies that exist for business processes and do not indicate how the dependencies may be manifested. Data flow diagrams show the flow of actual data between designed business activities and data stores, thus depicting how two activities interact.

In BAA Current Systems Procedure Analysis, a diagram should be constructed for the procedure level, possibly with other diagrams abstracted to show subsystems and systems.

4.3.8.2 AD8.2 - Problem Report

A description of a problem (i.e., an error or a perceived shortcoming) relating to a current business system.

4.3.8.3 AD8.3 - Procedure List

A hierarchical decomposition of current procedures as described in existing documentation and observed in practice. The decomposition is normally represented as an indented list, but a decomposition diagram can also be used. Separate procedure lists are produced for each business system analysed. The procedure list is analogous to the process decomposition diagram in the business area model.

4.3.8.4 AD8.4 - Data Analysis Diagram

A representation of the fields (data facts) and their associations that make up a user view.

4.3.8.5 AD8.5 - Implied Entity Relationship Diagram

An entity relationship diagram constructed by the synthesis of user views from current systems.

Supporting Deliverables

- User View Diagram
- Implied Entity Type List

4.3.8.6 AD8.6 - Key-Oriented List

A tabular representation of the key groups (data objects which are uniquely identifiable) and associations between key groups found in the analysis of user views from current systems. Keys are the physical equivalent of identifiers of entity types.

All keys found from the user views are listed and described. Each entry type in the key-oriented list contains the key (or concatenation of keys), associated non-key fields, associations with other key groups, and a name to be used for the possible implied entity type that may result from the entry

4.3.8.7 AD8.7 - Problem Analysis Report

A report that details the estimated causes, costs, impact, urgency and possible solution to each identified business problem. It contains a Cause/Effect Diagram.

4.3.8.8 AD8.8 - Comparison Check Report

A summary of the differences between the content and structure of current systems and the business model of the design area under consideration.

4.3.9 AD9 - BAA Report

The final product of a Business Area Analysis stage. The report may be split into an Outline BAA Report and a Detailed BAA Report for each design area.

The BAA Report contains the final deliverables of the Analysis stage, in any format and on any medium - but usually in the information library or code library - which are to be archived and/or serve as starting point for subsequent stages/projects. The final deliverables for the BAA stage should include:

- Clustered Process/Entity Type Matrix
- Comparison Check Report
- Data Stewardship Assignment
- Data Security Requirements
- Distribution of Data Usage
- Entity State Diagram
- Entity Type Volume Matrix

The final analysis stage deliverables are:

	Created/Updated in:	
Deliverable	**OBAA**	**DBAA**
Detailed Business Model		X
Outline Business Model	X	
BAA Report	X	X
Business Algorithm		X
Business Event Model	X	X
Cost-Benefit Analysis		X
Data Analysis Diagram		X
Design Area Definition	X	X
Distribution of Business Processes	X	

Entity Relationship Diagram	X	X
Entity State Matrix		X
Entity Type Definition		X

The interim deliverables for the analysis stage - which are usually not retained - would include:

Deliverable	**Created/Updated in:**	
	OBAA	**DBAA**
Business Event List	X	
Detailed Business Model		X
Detailed Business Model		X
Outline Business Model	X	
Draft Process Decomposition Diagram	X	
Information Needs Matrix	X	
Initial Entity Relationship Diagram	X	
Detailed Business Model		X
Outline Business Model	X	

4.3.10 AD10 - System Plans

4.3.10.1 AD10.1 - System Benefits Definition

The System Benefits Definition describes the quantitative and qualitative benefits expected to be derived from the implementation and operation of the proposed system.

Expression of Benefits

If possible, quantitative benefits of the system are expressed in direct cost savings. The assumptions underlying the calculation of these saving are clearly presented.

Qualitative benefits are usually expressed as general statements. If possible, the non-tangible benefits should be related to some quantitative measurement of potential benefit.

Scope of the Document

The scope of this document will vary greatly between projects, depending on the requirement to justify the development of the proposed system.

4.3.10.2 AD10.2 – System Scope Definition

This is a document that defines the scope of the project.

4.3.10.3 AD10.3 - Preliminary Implementation Schedule

The Preliminary Implementation Schedule identifies when major implementation milestones are likely to be completed. These milestones include completion of the design, commencement of construction and cutover, and completion of cutover activities. Dates are expressed in terms of elapsed time from an assumed start date.

5. Design Phase

5.1 Introduction

The business and technology designs of the information system and its supporting data structures are performed. The primary deliverable is referred to as a system design. An additional deliverable of this stage is the first-cut information system implementation and training plans as well as the continued materialisation of the information system documentation. The deliverables of this stage are collectively known as the information system design. The unit of work that produces these deliverables is known as the design project.

There are two groups of design steps:

1. Business System Design Phase Tasks

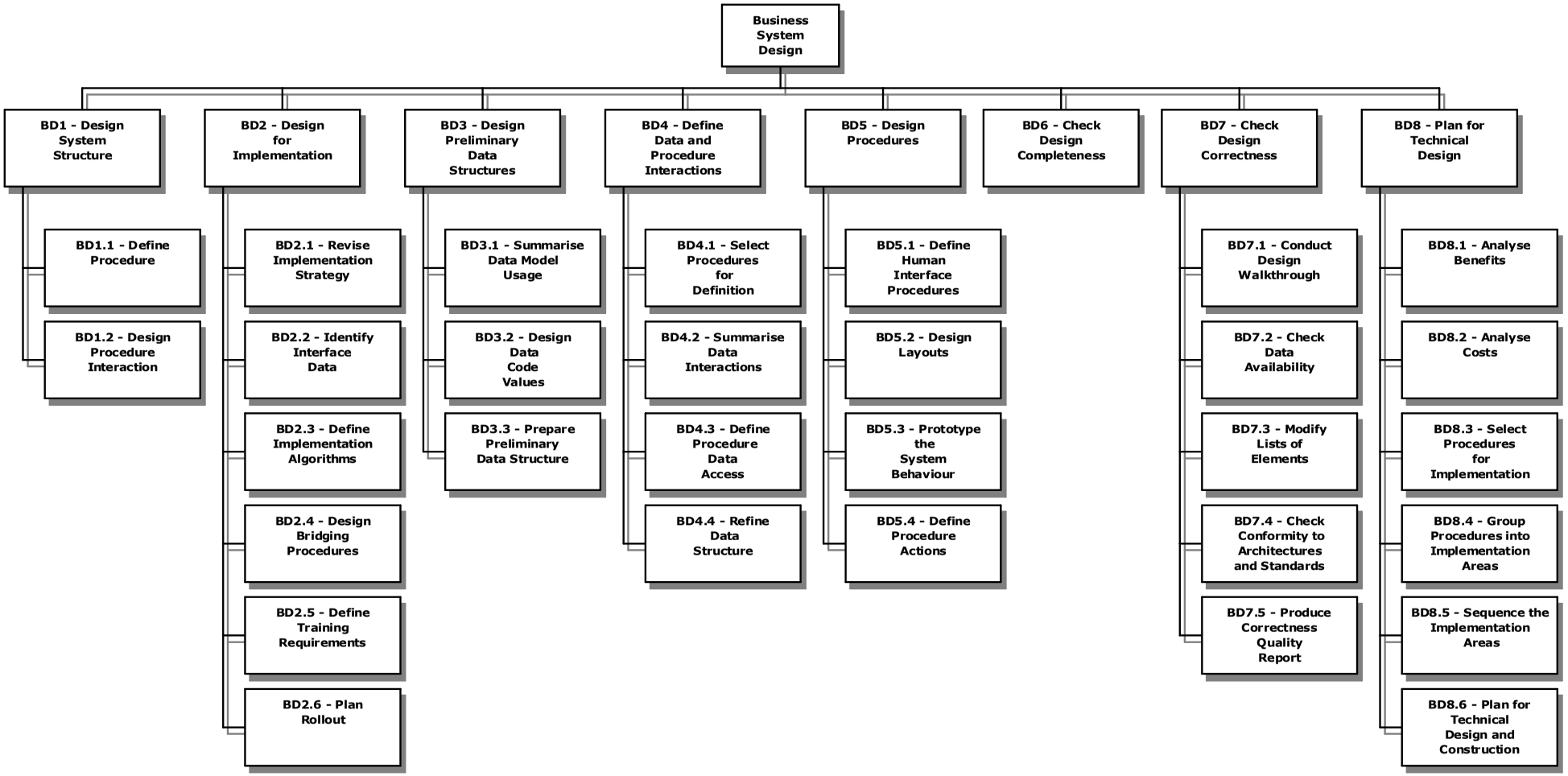
- Design system structure
- Design for implementation
- Design preliminary data structures
- Define data and procedure interactions
- Design procedures
- Check design completeness
- Check design correctness
- Plan for technical design

2. Technical Design Phase Tasks

- Define technical context and requirements
- Design the data structures
- Develop program specifications
- Prepare for testing
- Prepare for implementation
- Assess performance
- Design for system testing
- Plan construction and implementation stages

5.2 Business System Design

Schematically, the structure of the steps in the Business System Design phase is:


Business System Design
BD1 - Design System Structure
BD2 - Design for Implementation
BD3 - Design Preliminary Data Structures
BD4 - Define Data and Procedure Interactions
BD5 - Design Procedures
BD6 - Check Design Completeness
BD7 - Check Design Correctness
BD8 - Plan for Technical Design
BD1.1 - Define Procedure
BD1.2 - Design Procedure Interaction
BD2.1 - Revise Implementation Strategy
BD2.2 - Identify Interface Data
BD2.3 - Define Implementation Algorithms
BD2.4 - Design Bridging Procedures
BD2.5 - Define Training Requirements
BD2.6 - Plan Rollout
BD3.1 - Summarise Data Model Usage
BD3.2 - Design Data Code Values
BD3.3 - Prepare Preliminary Data Structure
BD4.1 - Select Procedures for Definition
BD4.2 - Summarise Data Interactions
BD4.3 - Define Procedure Data Access
BD4.4 - Refine Data Structure
BD5.1 - Define Human Interface Procedures
BD5.2 - Design Layouts
BD5.3 - Prototype the System Behaviour
BD5.4 - Define Procedure Actions
BD7.1 - Conduct Design Walkthrough
BD7.2 - Check Data Availability
BD7.3 - Modify Lists of Elements
BD7.4 - Check Conformity to Architectures and Standards
BD7.5 - Produce Correctness Quality Report
BD8.1 - Analyse Benefits
BD8.2 - Analyse Costs
BD8.3 - Select Procedures for Implementation
BD8.4 - Group Procedures into Implementation Areas
BD8.5 - Sequence the Implementation Areas
BD8.6 - Plan for Technical Design and Construction

5.2.1 BD1 - Design System Structure

5.2.1.1 BD1.1 - Define Procedure

Review documentation on processes:

Review the documentation generated as a result of BAA (Analyse Process Logic task and Produce Process Model task) to confirm understanding of the process. The process description, process logic diagram and process action diagram describe what the process is. The process decomposition diagram and process dependency diagram identify dependencies and relationships that may influence the type of mapping to be used.

Identify technique to be used:

Some Possible Techniques

For each process, select the most appropriate technique or techniques to implement it. The following partial list illustrates some of the possible techniques:

Computer supported

- On line: single exchange, conversational
- Batch: transaction driven, monolithic
- Generalised facility: end user query language, end user report writer, spreadsheet

Manual

- Entirely manual
- Machine supported (non-computer)

Determining the Best Technique

To determine the best technique, you will need information on:

- Time constraints for the process
- Volumes
- Type of user
- Design objectives of the project
- Technical context (e.g., hardware and software available)

Procedures That Use More Than One Technique

A procedure may use more than one technique; each different technique would be treated as a separate procedure step. For example, the Check Booking procedure may consist of an on-line step to receive the booking, and a background batch step to print a confirmation slip.

Attempt one-to-one mapping:

To start, try to map each process to a single procedure. Later steps will consider whether an alternative mapping is more appropriate. One-to-one mapping is most desirable because:

- Each elementary process is a complete activity from the user's point of view; the procedure will therefore be meaningful to the user and will leave the database in a consistent state.

- A change in an elementary process, due to a business change, will require a change only to the single procedure.

Consider alternative procedures:

Decide whether it is appropriate to provide more than one procedure to implement a process (i.e., to allow for more than one method of executing the same process). Consideration of fallback procedures (i.e., when the system is not available) may also lead to the definition of alternative procedures, but this step is undertaken after the primary procedures have all been defined.

Alternative procedures may be required if the same process is performed:

- In different processing environments (e.g., telephone sale versus sale by mail order)
- By users with different levels of experience or familiarity with the system
- In different hardware or system software environments

Consider serial procedures:

Decide whether a process should be broken down into two or more procedures that are performed one after the other. These are called serial procedures. Both alternative and serial procedures are cases of a one-to-many mapping from a process to procedures.

Consider compound procedures:

Decide whether several processes are best performed by one procedure, also called a compound procedure. Compound procedures are sometimes a good solution when:

- Two or more processes are always performed in series (e.g., two processes could be performed as one procedure in a batch job)
- Several similar processes involve the same entity type (e.g., the processes Increase Employee Salary and Change Employee Work Location are included in a single procedure, Maintain Employee Details)
- Procedures are generalised (e.g., a generalised procedure is written to maintain various tables of codes and meanings)

Define procedures:

Define each procedure by recording the following details:

- Procedure definition including: Procedure name, technique used, frequency/timing constraints, brief description of the procedure
- Elementary processes supported

This step creates the following deliverables:

- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

5.2.1.2 BD1.2 - Design Procedure Interaction

Prepare procedure dependency diagrams:

Prepare a procedure dependency diagram by mapping the elements of the process dependency diagram. When process to procedure mapping is 1:1 for every process, this exercise is trivial. More thought is required, however, whenever the process to procedure mapping is not 1:1.

The procedure dependency diagram is simply a copy of the process dependency diagram, where each:

- Process becomes a procedure
- Information view becomes a data view
- External process becomes an external object
- Process dependency becomes a data flow

This step creates the following deliverables:

- **Procedure Dependency Diagram** - see section 5.4.1.2 DD1.2 - Procedure Dependency Diagram on page 209

Define the data views:

Define each data view as a collection of other data views and/or fields, and document this in the information library.

Identify the data views while you construct the procedure dependency diagram.

Determine the data stores:

Before constructing the data flow diagram, define the data stores. The general approach involves defining a data store for each prospective physical file. By defining a data store, you are not committing yourself to using one physical file for that data store. The actual physical files will be determined during the TD phase of design stage.

Each data store is defined in the information library. It is assigned a name, given a brief description, and associated with the record types it contains. In some cases (e.g., transfer files and clerical files) it will also be necessary to define the record types.

Examples of Data Stores

A data store could be:

- An entire database
- A part of a database
- A master file
- A transfer file for interfacing procedures or systems
- A clerical file

This step creates the following deliverables:

- **Record Types List** - see section 5.4.5.7 DD5.7 - Record Types List on page 216

- **Data Store Description** - see section 5.4.1.6 DD1.6 - Data Store Description on page 210

Construct data flow diagrams:

Convert the procedure dependency diagrams to data flow diagrams using the Data Flow Diagramming technique.

This step creates the following deliverables:

- **Data Flow Diagram** - see section 5.4.1.5 DD1.5 - Data Flow Diagram on page 210

Synthesise to higher-level data flow diagrams:

The data flow diagram that has been drawn at the procedure level is now summarised at the subsystem and system levels.

This step creates the following deliverables:

- **Data Flow Diagram** - see section 5.4.1.5 DD1.5 - Data Flow Diagram on page 210

Check diagrams:

The diagrams will be checked formally as part of the Confirmation task, but, at this point, it is sensible to check the consistency of what has been drawn, and the completeness of the documentation.

Design the system controls:

System controls, which are requirements arising out of the system design, are now defined. The controls required by the business auditors that are independent of the design should have already been defined as processes during BAA. These controls should be reviewed again at the end of the "Procedure Design" task.

This step creates the following deliverables:

- **Procedure Definition (for System Controls)** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

Design fallback procedures:

For each procedure, the consequences to the business of a failure of the computer system must be analysed. You must decide whether the procedure can remain unavailable for the maximum expected outage, or whether alternative procedures must be available. When deciding on the fallback procedures, consult the output of the procedure design task and the data flow diagram.

5.2.2 BD2 - Design for Implementation

5.2.2.1 BD2.1 - Revise Implementation Strategy

An implementation strategy was chosen in the EM stage and revised in the last task of the BAA stage (i.e., Evaluate Design Areas). It dealt primarily with transition between design areas. A review is now undertaken to revise the strategy to take into account decisions made in the BSD phase of design stage and to deal with transition between implementation areas.

This step updates the following deliverables:

- **Implementation Strategy** - see section 4.3.7.4 AD7.4 - Implementation Strategy on page 132

5.2.2.2 BD2.2 - Identify Interface Data

- Identify data required as input from existing systems:

Every field in the design area must be put into one category to reflect its expected status at the time of implementation. If implementation is being phased, fields that are implemented in some locations but not in others must be considered.

The output from this subtask is a list of the fields in the interfaces. Every field in the design area is in one of the following categories:

- Input directly by human resources from data external to existing systems
- Derived from data in formerly existing systems

- Already present on the databases from a recently implemented system

- Identify data required as input to existing systems:

Every item established (i.e., created, updated or deleted) in the area must be considered for its impact on existing systems and must be described on the interface field list.

This step creates the following deliverables:

- **Interface Data** - see section 5.4.3 DD3 - Interface Definition on page 212

5.2.2.3 BD2.3 - Define Implementation Algorithms

For each field in the interface field input, an attempt must be made to define an implementation algorithm from which the field can be derived.

5.2.2.4 BD2.4 - Design Bridging Procedures

Every field in the design area required in bridging will have been identified previously.

Ascertain the frequency of bridging from one system to another. This, together with the data, will indicate the types of procedure required. For each bridging procedure, produce the full range of design documentation required, including all the implementation algorithms. All procedures designed in this stage must be developed with standard ADM techniques, must be included within the scope of the project and the specification, and must be ready for input to the TD phase. Include the protocol for maintaining the bridge.

This step creates the following deliverables:

- **Procedure Definition (for Bridging Procedures Design)** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

5.2.2.5 BD2.5 - Define Training Requirements

- Construct the procedure/technical skill matrix:

Construct a procedure/technical skill matrix that shows the technical skills required to perform each procedure being implemented.

This step creates the following deliverables:

- **Procedure/Technical Skill Matrix** - see section 5.4.4.1 DD4.1 - Procedure/Technical Skill Matrix on page 212

- Construct a procedure/organisational role matrix:

Construct a procedure/organisational role matrix that shows the association of the procedures being implemented and the organisational roles of the people who will perform and supervise the operation of the procedures. The organisational roles will be referred to by the standard job titles used in the organisation.

Creates deliverable Procedure/Organisational Role Matrix.

- Construct a skill/organisational role matrix:

Construct a skill/organisational role matrix that shows the skills that each type of employee needs to operate a new system. The matrix is derived from the procedure/technical skill matrix and the procedure/organisational role matrix constructed earlier.

The skills needed are shown by the letter "N" in the appropriate cell in the skills/organisational role matrix. Determine whether the employee already possesses any of the required skills, and, if so, change the cell value from "N" to "Y."

Label the rows :

Label the rows with:

- Procedure names for the procedural skills
- Technical skill names for the technical skills

Label the columns :

Label the columns with the organisational roles. The cell entries are derived from the procedure/organisational role matrix and the procedure/technical skill matrix. For each organisational role (i.e., column):

- Consider each procedural skill and decide whether it is required by a person with this organisational role; find the answer in the procedure/organisational role matrix.
- Consider each technical skill and decide whether it is required by a person with this organisational role. Derive the answer from the procedure/technical skill matrix by checking whether any of the procedures listed for this organisational role require the skill in question.

This step creates the following deliverables:

- **Skill/Organisational Role Matrix** - see section 5.4.4.3 DD4.3 - Skill/Organisational Role Matrix on page 213

- Construct training requirements matrix:

Construct a training requirements matrix that shows the skills that each employee needs to operate the new procedures, and whether or not the employee already has the skills. The rows of the matrix are the same skills as those used in the skill/organisational role matrix. The columns of the matrix are employee names. A cell value of "Y" shows that the employee needs and already has the skill. A cell value of "N" shows that the employee needs but does not presently have the skill.

Determine the employees who will operate or supervise the new systems procedures, and their organisational roles (note that an employee may have more than one organisational role). For each employee, indicate the skills required on the matrix.

This step creates the following deliverables:

- **Training Requirements Matrix** - see section 5.4.4.4 DD4.4 - Training Requirements Matrix on page 215

5.2.2.6 BD2.6 - Plan Rollout

Rollout is discussed in detail in the construction and implementation stages. At this point, it is sufficient to devise a strategy that defines:

- Training requirements
- Environmental (software and hardware) requirements
- Initial location for implementation

The strategy must be compatible with the bridging and conversion procedures defined earlier.

This step creates the following deliverables:

- **Rollout Plan** - see section 5.4.2.1 DD2.1 - Rollout Plan on page 211

5.2.3 BD3 - Design Preliminary Data Structures

5.2.3.1 BD3.1 - Summarise Data Model Usage

Create summary diagram -

Select or create a subset diagram for the entity types that support the processes for the design area. Record the location and time period for which the volumes on the diagram will apply.

Identify entry point

For each process that is to be given automated support and that will execute in the selected operational environment, and for each entity type used by the process, identify the selection condition. Entities are selected in two ways: by way of one or more relationships, and from the values of one or more attributes. The attributes used are entered onto the summary diagram in a large arrow pointing to the entity type or subtype.

When design for the selected database management software is affected by whether a relationship is used in one or in both directions, use of the relationship may optionally be indicated by a smaller arrow pointing to the target end of the relationship line.

Note integrity rules -

Add, to the entity type or relationship in the diagram, the identifying numbers of any integrity rules that are cited in the process logic or that are documented in the data model.

Calculate frequency of use

Frequency of use is a function of the number of times the entity is selected by a selection condition during a single execution of the process, and the number of times the process is executed in a time period for the specified operating environment. Add the frequency to the total usage of the entry point by previously analysed processes.

Check use of identifiers

Match the attribute list of each entity type with the selection conditions. Is each identifying attribute used by the group of processes analysed in at least one entry point? Are additional selection conditions needed to support inquiry or other processes for which no detailed process logic analysis exists? Add any additional necessary entry points to the summary diagram.

This step creates the following deliverables:

- **Data Structure Specification** - see section 5.4.5.3 DD5.3 - Data Structure Specification on page 215
- **Data Structure Summary Diagram**- see section 5.4.5.4 DD5.4 - Data Structure Summary Diagram on page 216

5.2.3.2 BD3.2 - Design Data Code Values

Identify fields to be encoded:

Fields to be encoded are determined either by the existence of an algorithm that derives values from meaning (e.g., abbreviation, encryption), or by consideration of all attributes classified as designed (i.e., an attribute invented to represent some other attribute or relationship). The most common designed attribute is the

numeric identifier of an entity that has other means of identification, such as a name.

Define the code structure:

Select or identify the following:

- Method of encoding (e.g., mnemonic, encryption, binary choice, serial number, arbitrary)
- Method of decoding (e.g., relationship/linkage, table look-up, associated field)
- Zones (e.g., sub-codes within the full code)
- Data type (e.g., numeric, alphabetic, alphanumeric)
- Responsibility for code values (e.g., external organisation, department, data administration)
- Whether associations between code values are supported

Assign code values:

and define associations with other codes and meanings, if allowed.
A set of values may be derived from the values of the domain of the attribute represented (e.g., abbreviation of months of the year).

A special case of encoding is the representation of a basic set of objects in a multilingual environment. Here, the starting point may be a set of meanings for one of the languages.

Identify code control procedures:

Consider the style of code, and the responsibility for code values. Serial numbers may be allocated by system or clerical procedure. The procedure may be a current system procedure.

Procedures may need to support the life cycle of entities identified by the code (i.e., creation, identification, authorisation, transfer of responsibility or other relationship or discontinuance).

This step creates the following deliverables:

- **Preliminary Data Code Structure** - see section 5.4.5.5 DD5.5 - Preliminary Data Code Structure on page 216

5.2.3.3 BD3.3 - Prepare Preliminary Data Structure

Prepare the preliminary data structure diagram:

Detailed diagram conventions and the method of constructing the diagram vary according to the target DBMS. This subtask is composed of these elements:

- Expand entity subtypes:

Entity types that are partitioned are expanded into a hierarchy of entity types, allowing one for each subtype.

- Expand multi-valued attributes:

Expansion of multi-valued attributes (if these have been allowed to remain in the data model) involves the creation of a new entity type that has a many-to-one mandatory relationship with the multi-valued attribute's entity type. The multi-valued attribute is then transferred to the new entity type.

- Expand many:many relationships:

Many-to-many relationships (i.e., M:M) are rarely left in a data model. The reasons for such a complex relationship almost invariably provide significant attributes that belong to the intersection entity type. If the relationships do remain in the model, you will find that many-to-many relationships are not usually directly supported by a database management system. In such cases, an additional entity type and two one-to-many relationships are substituted for the natural relationships.

- Map to record types and linkages:

Draw a data structure diagram, in which:

- Each entity type (whether original or added) is mapped to a record type and placed on the data structure diagram
- Each relationship is mapped to a linkage (of the same cardinality) on the data structure diagram

Sometimes this simple mapping results in a construct that does not conform to the structuring rules for the target DBMS. You must then create extra record types and linkages to ensure conformity with the DBMS structuring rules.

- Provide entry points:

The selection conditions identified in the information usage summary are now mapped to entry points.

This step creates the following deliverables:

- **Preliminary Data Structure Diagram** - see section 5.4.5.1 DD5.1 - Preliminary Data Structure Diagram on page 215

Document design elements:

In this task, the elements are identified, named, and described. The record types, linkages, and some integrity rules are formally documented.

The main design elements are:

- Record types
- Linkages
- Fields

- Fields:

Some fields have already been identified because they appeared in entry points. Others may have been identified while you were mapping relationships, depending on the target DBMS.

Now inspect each attribute within the scope of the project and ensure that it is mapped to a field. Normally, the field will be placed in the record that corresponds to the entity type of the source attribute.

- Integrity rules:

Integrity rules are essential to the design. These rules have four main sources:

- **Integrity conditions**: These conditions, involving attributes and relationships, are defined during BAA. They are now mapped into integrity rules, which are expressed in terms of fields and linkages. This is normally a straightforward mapping.
- **Relationship optionality:** The mandatory and optional natures of relationships must be preserved in the design. When this can be achieved through the inherent structure of the target DBMS, there is no need to define an integrity rule. Otherwise, a rule that all procedures must follow is now defined.
- **Subtyping**: Subtyping results in exclusive, mandatory, and optional linkages, so the resulting integrity rules must now be defined.
- **Duplication of data**: Whenever data are duplicated, an integrity rule is required to ensure that the two or more field values are kept consistent. Whenever derived field values are stored, a rule is required to ensure consistency between the derived field value and the source field values. Duplication of data may be introduced in the preliminary data structure design in order to circumvent the DBMS's structuring rules. In such cases, a rule must now be documented. Further duplication may be introduced in technical design phase.

Refine the data structure diagram:

The resulting data structure diagram may now be altered to meet privacy or integrity requirements.

- Private attribute values:

When an entity type has some attributes whose values are to be widely available and others whose values are private, the record type may be split in two to enhance the security of the private data. Similarly, certain entity occurrences in the population may contain attribute values that have a high security level, whereas other occurrences may not. The solution could be to use two record types for the two sets of entities in the population.

- High-integrity attribute values:

When high integrity is required for certain attribute values, the answer may involve duplicating the values, or maintaining check-totals or check-digits. You can now choose to add further fields to the design, or to subdivide the record types with special integrity requirements. This topic will be considered again during the "Design system structure" task, so it is not necessary to make a final decision at this point.

- Performance criteria:

So far, the data structure design mirrors the inherent structure of the data, so that it is easy to use and more resilient to changes in processes. During technical design, the performance of the total system design is checked to make sure that the system meets the given performance criteria and that it is feasible on the chosen hardware.

This step creates the following deliverables:

- **Refined Data Structure Diagram** - see section 5.4.5.2 DD5.2 - Refined Data Structure Diagram on page 215

Quantify record types and linkages -

Document the number of occurrences of each record type and the cardinality of each linkage: average, minimum, and maximum. This documentation can be done on a data usage summary diagram.

These figures are based on the equivalent figures collected during the Business Area Analysis project. When the record type or linkage has a 1:1 correspondence with an entity type or relationship, determining the equivalence is trivial. Otherwise, the figures must be decomposed or merged where processes have been decomposed or merged to form procedures.

This step creates the following deliverables:

- **Data Usage Summary Diagram** - see section 5.4.5.4 DD5.4 - Data Structure Summary Diagram on page 216

5.2.4 BD4 - Define Data and Procedure Interactions

5.2.4.1 BD4.1 - Select Procedures for Definition

Review the proposed technique for each procedure in the design area. Most procedures have been mapped from elementary processes that they are designed to support. For these procedures, a technique may originally have been

proposed. Other procedures have been added to support control or implementation.

Select for definition of interactions all procedures that are to be automated, fully or partly. Add them to the record/procedure matrix.

For each elementary process supported by the procedures, retrieve any available process logic or action diagram.

This step creates the following deliverables:

- **Procedures for Record/Procedure Matrix** - see section 5.4.10.1 DD10.1 - Record/Procedure Matrix on page 220

5.2.4.2 BD4.2 - Summarise Data Interactions

Add all record types included in automated data stores to the record/procedure matrix. If the actions for a procedure are not completely clear, consider drawing a data access diagram.

Entity/process involvement matrix -

Refer to the entity/process involvement matrix. Examine entity actions for all elementary processes that have been mapped to procedures.

Process to procedure mapping, and entity to record type mapping -
Use the Process To Procedure Mapping technique and entity to record type mapping to determine which record types are used by each procedure. For control and implementation procedures, work from the procedure definitions.

Data flow diagrams -

Examine the data flow diagrams for all procedures. Add Retrieve actions on record types represented in data views input to a procedure from either a data store or another procedure. Add these actions - Create, Update or Delete - as appropriate, for data views output to a procedure.

This step updates the following deliverables:

- **Record/Procedure** Matrix - see section 5.4.10.1 DD10.1 - Record/Procedure Matrix on page 220

5.2.4.3 BD4.3 - Define Procedure Data Access

This subtask is not performed for all procedures.

- Determine procedures:

Define the data access actions for procedures that:

- Have data actions that are complex or not well understood
- Maintain data that have complex linkages
- Are interactive with a critical response and potentially high volumes of data access
- Are interactive with a high frequency and potentially high volumes of data access

- Draw a subset of the data structure:

Identify, by inspecting the fields included, the records involved in each data view input to or output from the procedure. Also consider the record types included in the data stores accessed by the procedure.

Draw a subset of the data structure that includes the identified record types. Consider adding other record types to which they are linked, especially if the linkage is mandatory.

- Define the data access actions:

Determine selection conditions for each record type accessed.

If procedure access is being defined in order to begin technical design, estimate the access frequency per single execution of the process.

Determine access sequences by working from the use of the input record types to record types needed to assemble the output data views.

Data access actions can be summarised in a data access diagram.

This step creates the following deliverables:

- **Data Access Diagram** - see section 5.4.6 DD6 - Data Access Diagram on page 217

5.2.4.4 BD4.4 - Refine Data Structure

This is an optional subtask. It is performed if new linkages or fields are necessary to satisfy the selection of records by procedures. Make the additions to the preliminary data structure.

If the understanding of usage volumes for record type and linkage has changed, update the usage summary.

This step updates the following deliverables:

- **Data Usage Summary Diagram** - see section 5.4.5.4 DD5.4 - Data Structure Summary Diagram on page 216

5.2.5 BD5 - Design Procedures

5.2.5.1 BD5.1 - Define Human Interface Procedures

Define dialog style and standards:

Review any company wide standards relating to dialogs or screen layouts. Review hardware, software, and user characteristics to ensure appropriate standards. Define an overall dialog style to be followed, plus other screen-related standards. This should be done for the design area, even if company wide standards are not available. Examples of standards are:

- Use of special keys
- Screen layouts
- Screen navigation
- Method of displaying help text
- Dialog style (e.g., menu system)
- Security methods

Create dialog flow diagram skeleton:

Identify a list of on-line procedures for inclusion in the diagram. The procedure definitions from the process to procedure mapping task identify which procedures are on line.

Include each procedure on the dialog flow diagram, but do not attempt to indicate flows. It is not necessary to put these procedures in any particular order, but the diagram will be easier to use if related procedures (i.e., procedures with flows between them) are grouped.

Include additional procedures for the top-level menu, plus any other levels of menu, where appropriate.

This step creates the following deliverables:

- **Dialog Flow Diagram** - see section 5.4.7.2 DD7.2 - Dialog Flow Diagram on page 217

Define procedures:

This subtask is performed for each on-line procedure in turn.

- Review the procedure definition -

Review the procedure definition generated from the process to procedure mapping. This contains details of how the procedure is to be implemented and an outline of how the user will interact with the procedure. It also lists the process or processes that it implements.

- Review the process action diagram -

Review the process action diagram (and process logic diagram, if available) for each process implemented by the procedure. This gives a more detailed understanding of the processing involved, and it also defines the import and export data views for the process.

- Define the import and export data views -

Define the import and export data views for the procedure. These can be derived mainly from the import and export views for the process. In addition, you must include any extra data to be displayed to the screen as part of the export view.

This step creates the following deliverables:

- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

Define flows between procedures:

This subtask is performed for each on-line procedure in turn.

- Identify flows -

Identify flows into and out of the procedure. Consider the possible paths the user may wish to use to enter this procedure, and the possible destinations. The process dependency diagram will provide input to this analysis, because dependencies usually imply potential dialog flows. Flows may be to or from other design areas.

This step updates the following deliverables:

- **Data Flow Diagram** - see section 5.4.1.5 DD1.5 - Data Flow Diagram on page 210

- Define flow details -

Draw the flows on the diagram and document the flow details as described in dialog flow diagram conventions. When defining data sent to a procedure, check that this corresponds to attributes in the export and import views of the sending and receiving procedure, respectively.

This step updates the following deliverables:

- **Data Flow Diagram** - see section 5.4.1.5 DD1.5 - Data Flow Diagram on page 210

- Define keys and commands-

When a flow is to be triggered by the selection of a key, the procedure definition should be updated to record the key and the associated command that is to be processed by the procedure.

This step updates the following deliverables:

- **Data Flow Diagram** - see section 5.4.1.5 DD1.5 - Data Flow Diagram on page 210

Define procedure steps:

This subtask is performed for each on-line procedure in turn.

- Review the procedure -

Review the procedure to determine if it should be broken down into procedure steps. Separate procedure steps are required when more than one interaction with the user is needed to execute the procedure. Review the import and export views to decide if it is either desirable or possible for the user to input all of the required data or accept all of the output data in one interaction. The processing logic should also be reviewed because this may also indicate that there should be more than one interaction.

- Define new procedure steps -

The import and export views from the procedure are split over the procedure steps. Note that all elements within both the import and export views for the overall procedures must be included in the import/export view of at least one step. Additional elements may need to be included because of the breakdown into steps. Other details are defined for a procedure step as for a procedure.

This step updates the following deliverables:

- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

- Define flows between procedure steps -

The flows between the procedure steps and the flows to other procedures are defined. Note that flows into the overall procedure are now implicitly flows into the first step in the procedure.

This step updates the following deliverables:

- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

Define clerical procedures:

Identify the need for manual or clerical steps immediately preceding and following the automated procedures, by considering the following:

- The source and timing of the data flows input to a procedure
- Any associated system control procedures (e.g., to monitor performance of the automated procedures, or to provide an audit trail)
- The destination of data flows output from a procedure
- The collection, filing or dispersal of inputs and outputs, and any retention needs

Identify the user role relevant to performing each step.

Write a paragraph of text describing each step and add a short overview of the clerical procedure.

Consider what references to clerical procedures need to be added to the automated procedures (e.g., to prompt for preparation procedures, or to give guidance on how to follow procedures).

This step updates the following deliverables:

- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

5.2.5.2 BD5.2 - Design Layouts

Identify layouts required:

- Screen layouts -

In defining dialogs, place on a dialog flow diagram each procedure to be implemented as an interactive on-line procedure. Complex procedures are broken down into procedure steps. Each step has an import and/or an export data view associated with it. The data view is the basis for screen layout design. Review the information from the data flow.

- Report layouts -

Each procedure is defined during process to procedure mapping and is allocated a processing technique (i.e., batch or on line). A procedure may also appear on a data flow diagram. These deliverables are reserved to identify procedures requiring report layouts.

Choose layout style:

The layout style is determined by the characteristics of its user. Extensive prompts and instructions are appropriate for casual, inexperienced users. On the other hand, an experienced, dedicated user may want a minimum of such information. When a system has different kinds of users, be prepared to provide layouts in alternative styles.

Read the documentation for the procedure for which a layout is to be designed. Points of particular interest are:

- User type
- Volumes and usage of the procedure

Set layout system defaults:

To achieve a common representation of layout elements for the user, establish default properties and edit patterns that will be applied across layouts.

- Default display properties -

Default display properties are established for:
- Fields
- Prompts
- Literals
- Errors

- Default edit patterns -

Default edit patterns are established for standard field types. Standard field types equate to domains defined in Business Area Analysis. Common default types are date, time, monetary value, and quantity.

- Default versus customisation -

For each individual element, you can use the default or you can customise the property or edit pattern. Use customisation sparingly; it reduces consistency.

- System defaults have two major benefits, to:

- Ensure consistency in the designed layouts
- Improve productivity in the design

Design templates:

If an appropriate template is available, use it. Position the common layout elements on a template screen or report layout. These will become the basis for all other layouts. Layers of templates may be used (e.g., corporate, business system). The objective of the template is to ensure that common elements are presented consistently across the business system and the enterprise.

For the target user groups, identify other business systems and products that they use. Examine the layouts and identify common traits. Exploit the commonality to establish a consistent view for the user. There may be justification for alternative sets of templates within the business system for different user types.

- Define layout areas -

In order to create consistency across layouts, agreement must be reached on the general screen layout. Divide the layout into areas, each with a defined purpose. For example, on data entry screens, you could define a customer data area, an order data area, and a summary area.

- Define separate area subdivisions -

Separate area subdivisions with different purposes are defined for layouts. For screen layouts, there may be different areas for import and export data views. Alternatively, it may be appropriate to define layout areas on functional grounds (e.g., one for customer-related layouts and one for product-related layouts). The advantage of layout areas is that the user is accustomed to finding similar information in the same position across layouts. A template actually defines the initial layout areas, usually as header and footer areas on the screen, used for title and informational purposes, respectively. Layout areas are an extension of the principle of templates for variable layout items (i.e., fields).

- Divide the layouts -

Divide the layouts into groups based on usage. Divide the layout for each group into topic areas and agree on the division with users.

This step creates the following deliverables:

- **Design Template** - see section 5.4.8.2 DD8.2 - Design Template on page 218

Position fields on layout:

Position each field on the screen layout and define its visual characteristics. The fields are defined on the import and export data views.

Select each field from the appropriate data view and position it on the layout. Determine the position by examining the usage of the layout. Refine the positioning once all fields are on the layout. Define each field as soon as a decision is made regarding the field size for presentation and the prompt, because these affect positioning and the ability to fit the layout. Add literals to the layout to provide additional static information.

This step creates the following deliverables:

- **Screen/Report Layout** - see section 5.4.8.1 DD8.1 - Screen/Report Layout on page 217

Complete field definition:

Each field positioned on the screen layout must be defined. The position, dimensions, and prompt are known. A set of standard defaults has already been defined for fields, prompts, and literals. If the default is to be used, no additional definition is required.

Customised definitions are used to highlight elements on an individual layout. The additional information required is:

- Field edit characteristics for presentation
- Display properties, both for initial presentation and for error conditions

This step updates the following deliverables:

- **Screen/Report Layout** - see section 5.4.8.1 DD8.1 - Screen/Report Layout on page 217

Review layout with users:
In an environment where prototyping is used, the prototype is the appropriate vehicle for reviewing screen layouts. Review the first layouts designed, in order to establish that the broad guidelines embodied in templates and layout areas are agreed upon.

5.2.5.3 BD5.3 - Prototype the System Behaviour

Plan the prototyping effort:

Because the effort involved in prototyping can be significant, it must be planned. This involves:

- Selecting the part of a system to be prototyped
- Estimating effort
- Allocating tasks to designers
- Identifying overhead tasks: Set up panel, model, message, table and skeleton code libraries for prototyping. Set up menu hierarchies through which dialogs are invoked.
- Identifying tasks for each dialog: Prepare the initial prototype. Review it with the user (many times). Rework the prototype (many times).

This step creates the following deliverables:

- **Prototype** - see section 5.4.9 DD9 - Prototype on page 218

Prepare for prototyping

Before a prototype is shown to users, the sessions and prototype are set up and prepared.

This step updates the following deliverables:

- **Prototype** - see section 5.4.9 DD9 - Prototype on page 218

Run a prototyping session:

First, brief the users on prototyping. Then get them to use the prototype in the following way:

1. Position at the master menu.
2. Hand over operation to the user.
3. While the user is exercising the prototype, take notes.
4. Fix any problems.

During the session, fix all problems that require no more than ten minutes. Talk the user through the changes, asking for the user's advice, as appropriate. If many changes are required, the user may prefer to do something else for an hour or so. If the changes will take more than an hour, schedule another session.

5. Hand prototype back to the user for operation.

This step updates the following deliverables:

- **Prototype** - see section 5.4.9 DD9 - Prototype on page 218

Complete a prototyping session:

Before leaving the user, verbally summarise the main conclusions from the session and the further changes you intend to implement, and get the user's agreement. Confirm your agreement in writing within the next two days. When you return, make the changes to the prototype and print the screens involved. Summarise the points of agreement on a memo, as promised, and attach the printed screens for the user's records.

Finally, telephone the user to arrange the next session, if one is necessary.

This step updates the following deliverables:

- **Prototype** - see section 5.4.9 DD9 - Prototype on page 218

5.2.5.4 BD5.4 - Define Procedure Actions

Review procedure documentation:

Review the documentation relating to the procedure being designed. Examine the task inputs for each procedure step.

Describe logic, conditions, iteration and data actions of the procedure. There are several techniques for describing these actions - Procedure Action Diagrams are one of several techniques available.

Draw procedure action diagram:

The starting point is the process action diagram from BAA. This defines the business logic that must be embodied in the procedure. A skeleton procedure action diagram is produced that includes the relevant process logic.

- One-to-one relationships -

The task of defining procedures identifies the relationship between the procedure and the process. For a one-to-one mapping, the process logic expressed in the process action diagram defines the business requirements for the procedure and is included in the procedure action diagram. In all other cases, examine the mapping to determine how the logic is split over the procedures. If a process action diagram is not available, the business logic is first expressed in process action format.

- Multi-process procedures - procedure action diagram

For multi-process procedures, the process actions are either implemented serially or as alternatives. In a serial implementation, the actions are inserted one after another.

This step creates the following deliverables:

- **Procedure Action Diagram** - see section 5.4.1.3 DD1.3 - Procedure Action Diagram on page 209

Add dialog actions:

In a dialog, control is passed to the procedure through a dialog flow. For a procedure with a single flow in and a single flow out, the dialog control is simple. When there is more than one flow, it is more complex. The action of a procedure may be different under different incoming flows. Determine which outgoing flow is to be followed by setting an exit state within the procedure action diagram.

- Commands -

Examine details of the flow in order to control conditional actions. Examine the flows entering this procedure step and note the commands. If there is more than one command, determine whether the action of the procedure is different in each case and include the relevant conditional actions on the action diagram.
Examine all flows leaving the procedure and note the exit states. Determine the conditions under which each flow is taken and include actions in the procedure action diagram to set exit states.

This step creates the following deliverables:

- **Dialog Flow Diagram** - see section 5.4.7.2 DD7.2 - Dialog Flow Diagram on page 217

Add layout actions:

Although no actions are required to process the import layout, examine the contents of a layout to determine conditional actions required. Also note any algorithms required to derive values from fields.

Export layouts must be populated before the completion of the procedure. In addition, the procedure actions may:

- Set error indicators for particular fields
- Change the default display properties defined for a field on a layout
- Indicate where the cursor is positioned on the screen

This step updates the following deliverables:

- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

Define procedure data views:

Each procedure has up to four types of data views associated with it (i.e., import, export, entity action and local views). The import and export views are defined during the "Define Dialog Procedures" and "Design Layouts" tasks. The entity action data view is constructed as data actions are defined. Local data views are constructed as local data requirements are defined. Revisions may be made to import and export data views, but the impact on layouts and dialog flows must be assessed.

Include exception actions:

Review the action diagram for exception conditions. The identification of exceptions is mandatory for procedure action diagrams. Actions must be defined for all eventualities. Many actions have exception expressions within their syntax. An action or action group must be associated with each exception to ensure that all values are handled in a logical and consistent manner. In many cases, the action block will activate some form of error or help procedure, or a set of conditional actions.

Design reusable action blocks

In building the action diagram, identify algorithms or recurring procedure actions. These are documented in separate action diagrams, known as action blocks. Use an action block in any other procedure by invoking it with a USE statement in the procedure action diagram. An algorithm will have import and export views through which it receives the data on which it operates and returns a result. A catalog of algorithms is maintained so that they can be easily referenced and used in other procedures. The action block can be internal or external to the business system.

Annotate procedure action diagram

NOTEs are added to the procedure action diagram to explain complex logic or to summarise the purpose of lengthy blocks. Annotation can also be useful to:

- Explain exception conditions and handling
- Provide a definition for an action procedure on the action diagram
- Define the expected user interaction

This step updates the following deliverables:

- **Procedure Action Diagram** - see section 5.4.1.3 DD1.3 - Procedure Action Diagram on page 209

5.2.6 BD6 - Check Design Completeness

Prepare and check record type by procedure matrix

Draw the record/procedure matrix and carry out the checks stipulated. Modify lists of record types and procedures to include additional entries required to correct missing actions in the matrix.

This step updates the following deliverables:

- **Record/Procedure Matrix** - see section 5.4.10.1 DD10.1 - Record/Procedure Matrix on page 220

Prepare and check linkage by procedure matrix

Draw the linkage/procedure matrix and update the lists of linkages and procedures accordingly.

This step creates the following deliverables:

- **Linkage/Procedure Matrix** - see section 5.4.10.2 DD10.2 - Linkage/Procedure Matrix on page 220

Prepare and check field by procedure matrix

This step is optional and may not be justifiable if no automatic facilities exist to prepare all the matrices from the information library.

This step creates the following deliverables:

- **Field/Procedure Matrix** - see section 5.4.10.3 DD10.3 - Field/Procedure Matrix on page 221

Draw the field/procedure matrix and update the lists of fields and procedures.

For each record type, produce and verify the field by procedure matrices. This is a complex task, because the problems in any one matrix can affect all the other matrices. The task is not finished until all matrices have been completed and have been shown to be correct.

This step updates the following deliverables:

- **Field/Procedure Matrix** - see section 5.4.10.3 DD10.3 - Field/Procedure Matrix on page 221

Prepare completeness quality report

Following each matrix check, include any unresolved problems or weaknesses with appropriate comments.

This step creates the following deliverables:

- **Completeness Quality Report** - see section 5.4.11.1 DD11.1 - Completeness Quality Report on page 221

Carry out comparison with current systems

Compare the design with the documentation of current systems to determine if there are any elements (e.g., procedures, files, fields, etc.) not represented in the design. If you detect any omissions, update the lists of elements and make additional entries in the completeness quality report.

Review for completeness

Any modifications to the model may have created inconsistencies elsewhere in the model. Quality assurance must now judge whether it is necessary to repeat any or all of the completeness checking and according to what criteria.

5.2.7 BD7 - Check Design Correctness

5.2.7.1 BD7.1 - Conduct Design Walkthrough

The sequence of steps during the walkthrough is as follows:

- Resources-

The moderator determines if the resources required for the walkthrough are available. If not, the walkthrough is rescheduled; if so, the walkthrough proceeds.

- Minor error logs -

The participant-prepared minor error logs (part of the correctness quality report), compiled during preparation for the walkthrough, are given to the designer. This type of error is not discussed during the formal session.
This step creates the following deliverables:

- **Minor Error Logs (part of the Correctness Quality Report)** - see section 5.4.11.2 DD11.2 - Correctness Quality Report on page 221

- Walkthrough introduction -

The moderator states what is being reviewed, records the preparation time of all participants, and briefly reviews the procedure to be followed.

- The reading -

The reader, generally someone other than the owner of the model or document being verified, is selected by the moderator before the walkthrough. The reader paraphrases the materials of the design to be reviewed. As the reading proceeds, each inspector follows the material, looking for errors or ambiguities, and questions the work as necessary.

- Problem log -

Each problem not already recorded on a walkthrough problem log form (part of the correctness quality report) is recorded on such a form by the moderator or other designated person. The person who raises the problem must agree with any documented direction that the designer plans for resolution.

The details of the problem log forms are read so that each inspector is assured that the problems identified are adequately presented.

This step updates the following deliverables:

- **Walkthrough Problem Log (subset of Correctness Quality Report)** - see section 5.4.11.2 DD11.2 - Correctness Quality Report on page 221

- Problem resolution -

The moderator asks the designer to commit to a resolution date for each problem. This information is recorded on the problem log form.

- Walkthrough completion -

The moderator determines if the walkthrough is complete. If it must be reconvened, the moderator and designer agree on a date and conclude the current session. The reconvened session will review the entire model or document, not just the portion changed by this walkthrough.

- Summary report -

The walkthrough summary report (part of the Correctness quality report) is completed and sent to management and the participants. This report is used to record the walkthrough time, the number of problems, and so on. The walkthrough problem log forms become the basis for any rework done by the designer/developer.

This step updates the following deliverables:

- **Walkthrough Summary Report (subset of Correctness Quality Report)** - see section 5.4.11.2 DD11.2 - Correctness Quality Report on page 221

- Walkthrough results -

The results from the walkthrough session(s) are transmitted to the appropriate organisation for entry into the project problem log file. Whenever reported problems are no longer considered problems, they must be communicated to the tracking organisation so that the problem log file can be corrected.

- Rework

When the rework has been completed, the designer/developer notifies the appropriate manager. If another examination is required, it is scheduled. When all rework has been verified as accurate and complete, the walkthrough process is complete.

This step updates the following deliverables:

- **Correctness Quality Report** - see section 5.4.11.2 DD11.2 - Correctness Quality Report on page 221

5.2.7.2 BD7.2 - Check Data Availability

- Prepare data timing diagrams -

The procedure dependency diagrams or proposed physical data flow diagrams are assigned relative or absolute time values that indicate the procedure dependencies and the need for sequencing the data interfaces. In many cases, the timings will be simple sequential numbers on the procedure dependency diagram or proposed physical data flow diagram, but the merging of parallel procedures and the splitting of data to be applied to separate procedures must be considered.

The most difficult part of the timing diagram is when data are cycled through a procedure or set of procedures. The normal batch data entry procedure involves yesterday's data being merged with today's data for passing to the validation procedures. At the time that yesterday's corrected data are applied, different integrity considerations exist. Certain business cycles involve daily, monthly, annual, and other timing considerations that cannot be represented by a simple sequential procedure dependency diagram.

- Determine procedure to data availability -

Once the data timing diagrams have been constructed, inspection is required for the diagrams, the data definitions, and the procedure definitions to verify the correctness of the actions.

In the data interface verification approach, the procedures are inspected in reverse order. The inputs to the terminating procedures are considered the outputs from previous dependent procedures. This reverse verification can include cyclic verification and integrity checking.

This step creates the following deliverables:

- **Data Timing Diagram** - see section 5.4.1.7 DD1.7 - Data Timing Diagram on page 210

5.2.7.3 BD7.3 - Modify Lists of Elements

- Procedures -

The walkthrough or availability check may first reveal that certain procedures are missing. The definition of these additional procedures requires modification of the data flow diagram and the creation of a new procedure logic diagram.

For example, this step may update the following deliverables:

- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209
- **Procedure Dependency Diagram** - see section 5.4.1.2 DD1.2 - Procedure Dependency Diagram on page 209
- **Data Flow Diagram** - see section 5.4.1.5 DD1.5 - Data Flow Diagram on page 210

- Record types -

The walkthrough or availability check may show that record types are missing from the documented list of current record types. Additional record types may have been created when two different but similar record types were inadvertently given the same name. The difference between record types and record subtypes may become definite by this stage of the design, and the ownership of the different subtypes may be known. In all cases, the definitions of the record types may require further specification.

- Fields -

The walkthrough or availability check may show where additional fields or interface fields are required, where the same field has been given different names, and where the definition of a field needs further specification. The procedure data flows need clear definition of the fields and their integrity rules, life cycles, and ownership. This can lead to the qualification of fields with supportive fields; for example, a currency field may be given a related currency code and implied decimal point code field. It can also lead to the division of some of these interface fields into more than one field (e.g., where an Order Number may be split to indicate the Customer Order Number and the Product Order Line Number).

When any change is introduced, the impact of that change must be given careful consideration by data administration.

In most cases, an additional field will not have a significant impact, but a change to the definition of a field may affect previous completeness work.

5.2.7.4 BD7.4 - Check Conformity to Architectures and Standards

- Business systems architecture -

The design documentation must be acceptable, and it must conform to the overall business systems architecture, because it defines the scope of the business application, the interfaces between it and other business systems, and the benefits that are expected from its development.

To conform to these ideals, the design must support the business objectives. It must also be compatible with future requirements, growth projections, and the style of business that is being addressed.

- Technical architecture -

The design must conform to the overall technical capabilities that have been defined, during Enterprise Modelling (EM), within the technical architecture. Although technical design is still to come, the technical feasibility of the business system design must be clear.

The technical architecture may have defined the capability of the terminal screen layouts, the keyboard facilities, the data management system, the degree of centralisation or decentralisation, and the nature of the human interaction or network that may be used. The layout specifications will be designed to match the physical capabilities of the equipment that is the technical target of the business system.

- Project standards -

Walkthroughs and Quality Assurance verify that the project standards have been applied in terms of the completeness, correctness, and usability of the designed system.

5.2.7.5 BD7.5 - Produce Correctness Quality Report

The weaknesses and problems in the business system design phase are listed in the correctness quality report. Many of these problems will not prevent the project from continuing while the problems are fixed. The quality assurance standards will indicate what is allowable. This report is enlarged as correctness checking proceeds.

The final report may simply state that the design is acceptable and conforms to all the objectives, or it may nominate problem areas that require further inspection, design, and confirmation.

The problems are registered in the information library as "business systems design problems."

The final step is to review the results of the correctness checking task against the objectives and success criteria for the task. Walkthroughs may have identified problem areas, but the scope of those problems may not prevent the next stage from beginning while certain repair work is undertaken.

This step updates the following deliverables:

- **Correctness Quality Report** - see section 5.4.11.2 DD11.2 - Correctness Quality Report on page 221

5.2.8 BD8 - Plan for Technical Design

5.2.8.1 BD8.1 - Analyse Benefits

Benefit analysis provides a basis for prioritising. Benefits can be summarised in a procedure/benefit matrix. Steps in the analysis are:

- Business needs and objectives -

Take each procedure (or related group of procedures) in turn and determine which business needs and objectives the procedure meets.

- Business benefits -

Determine if there are any business benefits to be gained from implementing the procedure. If it is already a manual procedure, evaluate the benefit of automating it. If it is a new procedure, evaluate the benefit of implementing it manually as well as on a machine. It if is already automated, determine the benefit of re-implementation.

- Intangible benefits -

List clear financial gains as well as intangible benefits, such as:

- Better information
- Removal of existing problems
- Exploitation of new opportunities

- Monetary values -

Put a monetary value on each intangible. If you cannot put a value on it, it is not worth listing. A wrong value is better than no value at all; as soon as it is written down, people challenge it and a consensus will develop.

This step creates the following deliverables:

- **Procedure/Benefit Matrix** - see section 5.4.12 DD13 - Procedure/Benefit Matrix on page 221

5.2.8.2 BD8.2 - Analyse Costs

Accurate costing of systems development is essential by this stage. This is achieved by employing standard ADM planning guidelines, and by making suitable allowances in them for major development cost variables.

Major development cost variables:

- Designer skill level
- Software in use
- Database size
- Database complexity
- Transaction volumes
- Transaction complexity

The steps in costing for each procedure:

Estimate operational costs -

Operational costs are based on the frequency of the procedure.

These include:

- PCs/Servers
- Data communication lines and equipment
- Other specific devices

Estimate development costs -

Development costs must be amortised, explicitly or implicitly, as an investment.

These include:

- Technical design and construction
- Data collection and conversion (if any)
- User preparation
- Installation of the system

Compare total costs against the value of the benefits. Rank the procedures by quantified benefits less estimated cost. These costs do not include those shared with other procedures.

This step creates the following deliverables:

- **Cost-Benefit Analysis** - see section 5.4.13.1 DD13.1 - Cost-Benefit Analysis on page 221

- **Cost-Benefit Summary Report** - see section 5.4.13.2 DD13.2 - Cost-Benefit Summary Report on page 222

5.2.8.3 BD8.3 - Select Procedures for Implementation

This task reduces the amount to be implemented by applying selection techniques.

Criteria for selecting procedures for automation -

A procedure that scores against a number of the following headings is a good candidate for automation.

- Automation shows a clear cost benefit
- The procedure is repeatable and well defined
- The procedure is executed frequently
- It is a current bottleneck in the organisation
- It is currently largely clerical
- It includes a lot of data handling of sub-procedures
- It involves the retrieval and collation of data from diverse sources
- There is currently no computer support
- It passes data to or receives data from procedures that already have computer support
- It is a sub-procedure common to many procedures
- It is expected to change significantly in the future

Steps for ranking procedures –

Follow these steps to rank procedures until some total budget is reached:

1. Remove any procedures now clearly not suited to computer support.
2. Score each procedure now being analysed according to the selection criteria weights.
3. Rank the procedures by their score.
4. Add the cost of each mechanism until the cost limit is reached.
5. Similarly, take the list of procedures ranked by cost benefit and work down the list until the cost limit is reached.
6. Select the procedures that now appear on both lists.
7. Reconsider the benefits and selection criteria of the remaining procedures.
8. If necessary, agree on a weighting for the cost benefit to arrive at one ranked listing.
9. Select more procedures until the cost limit is reached.

5.2.8.4 BD8.4 - Group Procedures into Implementation Areas

Perform this task by arranging the logical constraints imposed upon the implementation sequence by the need for data and related procedures. Analyse logical constraints to determine the sequence in which procedures and preliminary data structures should be implemented, and to group them into reasonable areas for implementation.

- Purpose of implementation areas -

The purpose of grouping into implementation areas is to construct units that can be implemented as a whole. Implementation units should be small to allow evolutionary, incremental development. If large, complex implementation areas are constructed, risks are increased. The time required before there is any return on investment is also increased.

- Criteria for grouping procedures, linkages and record types -

Apply the following criteria for grouping procedures, linkages and record types into an implementation area:

- They should all be related to the same business need, objective, or benefit
- All data needed to support high-priority procedures should be included; consolidate procedure action diagrams
- All procedures required to support other necessary data should be included
- They should form a consistent, coherent whole
- The area should be small

- Sequence for implementing areas -

Define implementation areas so that each can be designed and implemented as a unit.

Build up the sequence for implementing these areas by considering the sequence in which individual procedures have to be implemented and by grouping related procedures. If you move a procedure from one implementation area to another, it may change the sequence in which the areas have to be implemented.

One of two situations arises:

- You have already implemented the necessary procedures and data.
- You must carry your design back into this new area and implement these procedures first.

- Identification of needed data -

Identify what data are needed. Once you have an initial list of procedures for each implementation area, you must check that the areas make sense in terms of data. Produce a definitive list of the record types, fields, and linkages to be incorporated in each implementation area.

For each procedure in an implementation area, take the procedure action diagram and add all the record types, linkages, and fields it uses to a list for the area. Two excellent rules are: always start at the top of a hierarchy and work down, and do Create functions first.

If you identified your procedure data flows correctly, these constraints should be consistent with the data flows. At worst, they will be additional because you overlooked something. If they conflict, check if the data flows and the model are right. This cross-check is often useful for unearthing misunderstandings.

- Identification of common records and procedures -

A question that will arise from time to time during the BSD is: "What is the scope of this procedure?" The designer has been able to specify many procedures that represent one process, or one procedure that has been generalised to represent many processes. Use cluster analysis to identify where record types can be combined, where common procedures can be found, and where two or more procedures may be combined into a single procedure.

- Common records -
Using the record/procedure matrix, you can identify the record types that are always used by a common set of procedures. Within the constraints of the logical constraint analysis, you can combine those record types into a single record type that is used by the set of procedures. This reduction in data structuring complexity will reduce the development costs, and the simplified procedures will be cheaper to maintain.

- Procedures that can be combined -

The linkage/procedure matrix and procedure dependency diagrams will indicate the serial structure of procedures that may be combined into a single procedure. In many cases, artificial procedures are defined that are only steps in a bigger and more comprehensive procedure. Cluster analysis enables procedures to be combined up to their dependency constraints.

- Common procedures -

Common procedures are more difficult to detect because the data that are acted upon have different names. As an example, the procedure to update the customer name and address may be similar to the procedure to update the supplier name and address. The field/procedure matrices will indicate when similar fields are manipulated in the same way. By designing the procedure to act on either the customer or supplier data, you can make one procedure do the work of two.

This step creates the following deliverables:

- **Implementation Area** - see section 5.4.2.2 DD2.2 - Implementation Area on page 211

5.2.8.5 BD8.5 - Sequence the Implementation Areas

Cluster analysis indicates a logical sequence for implementation areas. The logic is based on a desire to minimise dependencies or interfacing requirements between areas. The areas that are relatively independent can be implemented before those with many dependencies, but business priorities may override the logical sequence. A countervailing force then comes into play because sequences other than the logical are likely to be more costly to implement. This is because the non-logical sequence will have a greater dependence on current systems for the (temporary) provision of information; building bridges to old systems is usually expensive.

- Assess alternative implementation sequences -

Assess the cost implications of alternative sequences. The costs associated with each area are already known, but the implementation cost for each sequence is not. Assessing each possible sequence is generally a waste of effort. At the most, consider four strategies.

For each of these, perform an implementation analysis. The results are presented to the end users. From this evidence, they choose their preferred sequence, which then becomes the agreed-upon one for planning.

The four strategies that should be considered are:

- The preferred business sequence
- The logical sequence
- A minimum-risk sequence
- A level-resourcing sequence

- Choose an implementation strategy -

If you choose an implementation strategy at one end of the spectrum, you must carry out a total conversion all at once. At the other end, you can evolve, transaction by transaction. In general, an incremental approach is better. Extra costs may be involved in developing programs to convert from old data formats to new, and possibly back again to maintain some existing systems, but the reduction in risk is usually worthwhile.

The ideal strategy is to be able to add, replace, or remove a small group of programs and data items at a time. This approach minimises risk and eases the burden of change on each user.

Reasons for choosing an incremental approach:

- A partial conversion, when a number of smaller steps are implemented one at a time, carries lower risks.
- It gives a quicker return on investment.
- It allows each user to comment on the system early enough to allow changes to be made.

- It eases the introduction, allowing each user to become familiar with the new system in easy stages.

This step updates the following deliverables:

- **Implementation Strategy** - see section 4.3.7.4 AD7.4 - Implementation Strategy on page 132

5.2.8.6 BD8.6 - Plan for Technical Design and Construction

Planning for technical design and construction is essentially the assembly and publishing of material created in the earlier tasks, with the addition of preliminary project plans. Follow a standard planning procedure. Steps are:

- Review the standard tasks -

Take note of any new or unique tasks that have to be built into the plan, and then create a planning chart. Standard tasks include:

- Physical database design
- Conversation and transaction dialog design
- Message size and interaction frequency design
- Computer resource requirements design for each interaction
- Screen layout timing and frequency design
- Report layout timing and frequency design

- Estimate design resources -

Take account of the software to be used and the skills of your available staff, as well as of application details.

- Estimate technical design time scales -

These will depend upon the estimates of design resources and on the availability of the appropriate people.

- Describe risks -

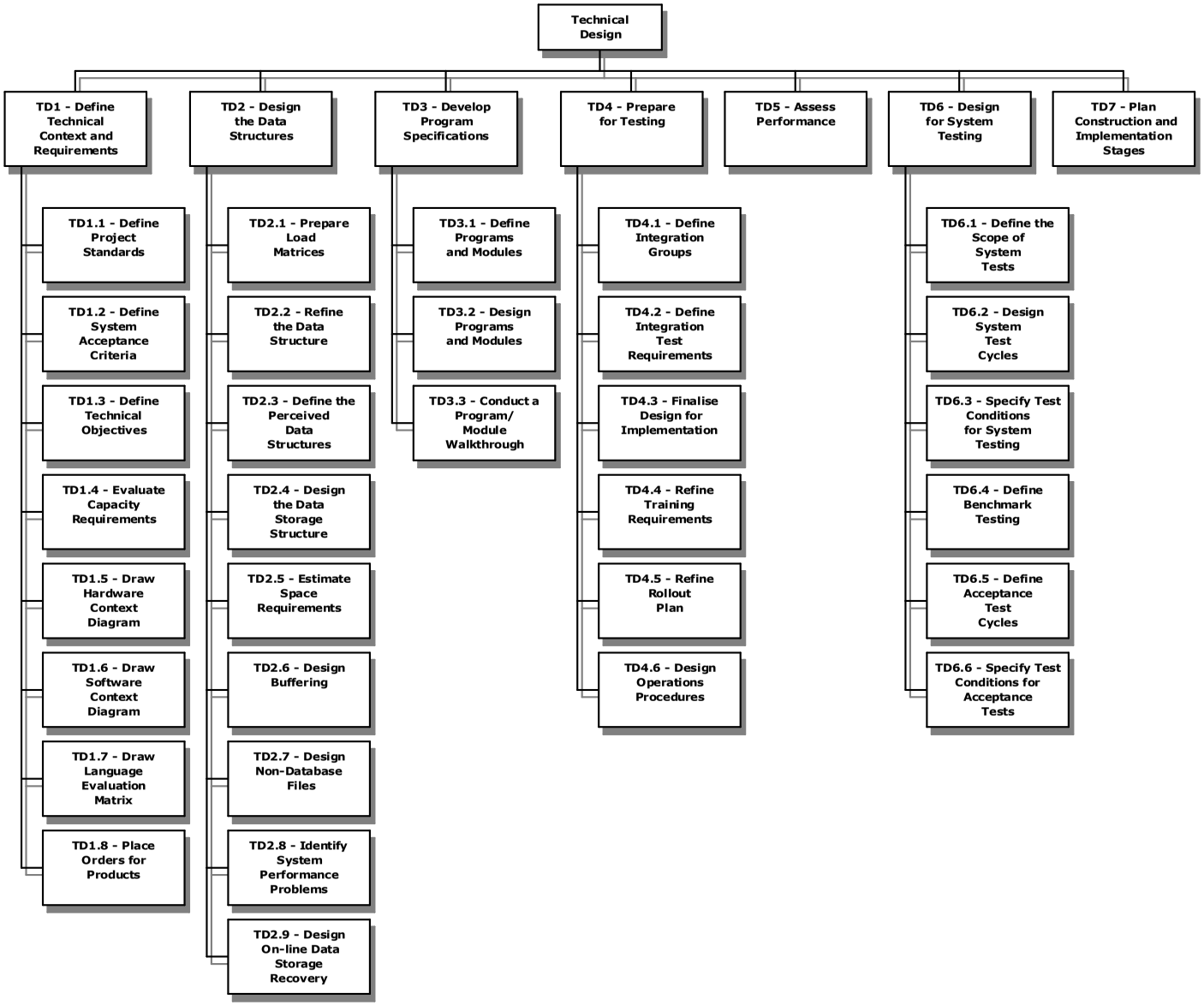
Take actions to control risks and amend the plan if necessary.

This step creates the following deliverables:

- **Technical Design Phase Plan** - see section 5.4.14 DD14 - Technical Design Phase Plan on page 222

5.3 Technical Design

Schematically, the structure of the steps in the Technical Design phase is:


Technical Design
TD1 - Define Technical Context and Requirements
TD2 - Design the Data Structures
TD3 - Develop Program Specifications
TD4 - Prepare for Testing
TD5 - Assess Performance
TD6 - Design for System Testing
TD7 - Plan Construction and Implementation Stages
TD1.1 - Define Project Standards
TD1.2 - Define System Acceptance Criteria
TD1.3 - Define Technical Objectives
TD1.4 - Evaluate Capacity Requirements
TD1.5 - Draw Hardware Context Diagram
TD1.6 - Draw Software Context Diagram
TD1.7 - Draw Language Evaluation Matrix
TD1.8 - Place Orders for Products
TD2.1 - Prepare Load Matrices
TD2.2 - Refine the Data Structure
TD2.3 - Define the Perceived Data Structures
TD2.4 - Design the Data Storage Structure
TD2.5 - Estimate Space Requirements
TD2.6 - Design Buffering
TD2.7 - Design Non-Database Files
TD2.8 - Identify System Performance Problems
TD2.9 - Design On-line Data Storage Recovery
TD3.1 - Define Programs and Modules
TD3.2 - Design Programs and Modules
TD3.3 - Conduct a Program/ Module Walkthrough
TD4.1 - Define Integration Groups
TD4.2 - Define Integration Test Requirements
TD4.3 - Finalise Design for Implementation
TD4.4 - Refine Training Requirements
TD4.5 - Refine Rollout Plan
TD4.6 - Design Operations Procedures
TD6.1 - Define the Scope of System Tests
TD6.2 - Design System Test Cycles
TD6.3 - Specify Test Conditions for System Testing
TD6.4 - Define Benchmark Testing
TD6.5 - Define Acceptance Test Cycles
TD6.6 - Specify Test Conditions for Acceptance Tests

5.3.1 TD1 - Define Technical Context and Requirements

5.3.1.1 TD1.1 - Define Project Standards

Standards will exist within the enterprise that are outside the scope of the rules and conventions detailed in the Application Development Methodology (ADM). These standards may relate to specific software and hardware products and how they are to be used. Naming standards are devised for the production and development environments. Define development procedures, which cover the use of the development environment. It may not be possible to complete this until the first three steps of system generation have been completed.

This task is necessary only when new computing and development environments are to be used. Otherwise, standards will already exist.

This step creates the following deliverables:

- **Project Standards** - see section 5.4.15.1 DD15.1 - Project Standards on page 222

5.3.1.2 TD1.2 - Define System Acceptance Criteria

The overall criterion is that the system must operate as described in the business system design specification. This does not need to be recorded because that is assumed. However, variations from the business system design specification that may be tolerated should be recorded within the acceptance criteria.

Additionally, include any particular aspects of the specification or tasks that the user wants to emphasise.

Input to defining the acceptance criteria are those who will use or operate the new system, including the:

- End-user department (or departments) that will use (and pay for) the new system
- Business auditors who audit the efficiency of the end user departments
- Computer operations department that will operate the centralised facilities of the new system
- Communications department that will control the network used by the new system (if this is different from the computer operations department)

The computer auditors will supply input and approve the acceptance criteria.

This step creates the following deliverables:

- **Acceptance Criteria** - see section 5.4.15.2 DD15.2 - Acceptance Criteria on page 223

5.3.1.3 TD1.3 - Define Technical Objectives

During BAA and BSD, the business objectives and end-user view of the system are described. At the outset of the TD phase, these must be interpreted from a technical viewpoint, and a set of technical objectives must be established. The TD objectives for a strategic decision support system to evaluate a short-term construction project will be different from those for an operational-level system that must support thousands of daily transactions for many years.

The factors that will affect the technical objectives are:

- System category (strategic, planning and analysis, monitoring and control, operational)
- Planned life
- Types of use
- Expected volumes
- Expected transaction rates
- Expected communication loads
- Business cycle

This step creates the following deliverables:

- **Technical Objectives** - see section 5.4.15.3 DD15.3 - Technical Objectives on page 223

5.3.1.4 TD1.4 - Evaluate Capacity Requirements

At the outset of the TD phase it is not possible to evaluate the eventual capacity requirements of the business system accurately. During TD, as a complete picture of how the system will operate emerges, capacity requirements are assessed using the performance assessment technique. Because of the long lead-time required to purchase and install certain equipment, however, it is necessary to identify equipment requirements as early as possible. The outputs of BSD should indicate the size and operational demands of the system. Statistics from BSD phase can be used to approximate capacity requirements, especially when compared with systems constructed in a similar technical context. The greater the similarity between the technical contexts, the greater the confidence in the results. The approximation of capacity should be accompanied by a list of any assumptions made during the evaluation. These assumptions can be monitored during the project, and any deviation can be noted. The following statistics from BSD can be compared with previous experience:

- Number of automated procedures
- Procedure execution frequency
- Type and number of data accesses
- Procedure logic complexity

- Inter-location data flows
- Volumes of record types

This step creates the following deliverables:

- **Capacity Evaluation Assessment** - see section 5.4.15.7 DD15.7- Capacity Evaluation Assessment on page 226

5.3.1.5 TD1.5 - Draw Hardware Context Diagram

The technical architecture diagrams and descriptions developed during Enterprise Modelling stage are updated continually as new products are investigated and incorporated into the architecture. The current version of the technical architecture is the starting point in identifying the technical context for the business system. The Enterprise Modelling stage and BAA projects will have identified locations for data stores, processing and users.

Separate hardware contexts may be specified for the production and development environments. The hardware context is summarised in a hardware context diagram. The production environment may incorporate multiple sites and multiple hardware facilities. The development environment must include at least one occurrence of each hardware facility that occurs in the production environment.

Construction may take place on a hardware facility other than the target facility. The target facility must be available for system and acceptance testing.

This step creates the following deliverables:

- **Technical Architecture** - see section 5.4.15.4 DD15.4 - Technical Architecture on page 223

- **Hardware Context Diagram** - see section 5.4.15.5 DD15.5 - Hardware Context Diagram on page 226

5.3.1.6 TD1.6 - Draw Software Context Diagram

A software context diagram is drawn for each processing device according to the conventions. The software elements required and the facilities offered by each element are identified. These are summarised on the facility/software element matrix.

- Provision of facilities -

In some cases a facility may be provided by more than one product included in the context. For example, transaction logging and recovery may be handled separately by a transaction processing system and by the database management system. Ultimately, software elements may cooperate to support a facility, or a

decision may be made to allow one element to handle it. If this decision has not already been made in the technical architecture or technical standards, it must be made during the TD phase. These decisions should be recorded on the facility/software element matrix.

This step creates the following deliverables:

- **Facility/Software Element Matrix** - see section 5.4.15.6 DD15.6 - Facility/Software Element Matrix on page 226

- Development and production diagrams -

Separate software context diagrams may be defined for the development and production environments. The development environment may include some products used only during the system construction. The production environment should not include any software elements not available in the development environment.

This step creates the following deliverables:

- **Software Context Diagram** - see section 5.4.17.5 DD17.5 - Software Context Diagram on page 229

5.3.1.7 TD1.7 - Draw Language Evaluation Matrix

The final task needed to define the software context fully is to make recommendations on the use of languages. Gone are the days when entire business systems were developed in only one language.

In this task, the right language for each job is determined. To do this, the generic procedure types are listed in a language evaluation matrix against the languages available. In this context, only a broad recommendation for language use is necessary. The recommendation is used in the software design step, when a decision is made for each module.

This step creates the following deliverables:

- **Language Evaluation Matrix** - see section 5.4.15.8 DD15.8 - Language Evaluation Matrix on page 226

5.3.1.8 TD1.8 - Place Orders for Products

When long lead-times on product delivery and work completion are problems, orders must be placed as early as possible. The capacity requirements identified in the subtask Evaluate Capacity Requirements are a reasonable estimate, at best. For this reason, only provisional orders should be placed for the minimum number of products required in the early stages of construction. Suppliers will

normally be willing to negotiate on a declaration of intent rather than a firm order for the additional products.

Products to be ordered at this stage include hardware, software and facilities. A provisional order may include a work order to have cables laid internally or to have system software loaded.

5.3.2 TD2 - Design the Data Structures

5.3.2.1 TD2.1 - Prepare Load Matrices

There are three types of load matrices: record type load matrix, search field load matrix and linkage load matrix. The load matrices are prepared from the contents of the procedure action diagrams. The contents of the load matrices may be summarised by annotation of the preliminary data structure diagram with the load statistics.

This step creates the following deliverables:

- **Record Type Load Matrix** - see section 5.4.16.1 DD16.1 - Record Type Load Matrix on page 228
- **Search Field Load Matrix** - see section 5.4.16.2 DD16.2 - Search Field Load Matrix on page 228
- **Linkage Load Matrix** - see section 5.4.16.3 DD16.3 - Linkage Load Matrix on page 228

5.3.2.2 TD2.2 - Refine the Data Structure

The preliminary data structure, consisting of a diagram and information library entries, is developed into a refined data structure, consisting of a refined data structure diagram and updates to the information library contents.

The load statistics collected in the subtask Prepare Load Matrices helps the designer choose between the various design options provided by the database software. The details of this task vary according to which database management system is to be used in the production system.

This step updates the following deliverables:

- **Refined Data Structure Diagram** - see section 5.4.5.2 DD5.2 - Refined Data Structure Diagram on page 215

5.3.2.3 TD2.3 - Define the Perceived Data Structures

The details of this step depend upon the database management system or file access method adopted. There are three main approaches adopted by those who implement the database management system:

- No perceived record concept exists -

If this approach is applicable, this subtask is omitted.

- Only perceived records may be referenced by programs -

In this approach, a set of ideal perceived records is designed for each procedure within the implementation area. It may be practical to allow each procedure to have its own set of perceived records, which will enhance program and data independence considerably.

If it is impractical to support so many sets of perceived records, the perceived records are aggregated into a unified collection that will support all processing requirements of the implementation area. Do this by identifying duplicated record types and by merging record types having the same key. The database designer has to define to the database management system how the perceived records are to be built from the underlying records (i.e., those defined in the refined data structure).

- Programs may reference perceived and underlying records -

In the third approach, perceived records are defined only when it is useful for program development, maintenance, or security, or for end-user query languages and analysis packages. The database designer needs to define to the database management system how the perceived records are to be constructed from the underlying records.

This step creates the following deliverables:

- **Program Specification** - see section 5.4.17.1 DD17.1 - Program Specification on page 228
- **Perceived Data Structure Diagram** - see section 5.4.5.8 DD5.8 - Perceived Data Structure Diagram on page 216

5.3.2.4 TD2.4 - Design the Data Storage Structure

Specify the data sets (i.e., physical files) that will be used to store the database. This involves defining which database records will be allocated to each data set, deciding all the physical characteristics of the data sets, and determining the physical arrangement of these data sets on the storage medium. A data storage summary is prepared.

Because the amount of choice available depends upon the database management software and operating system, details of this subtask will vary.

This step creates the following deliverables:

- **Data Storage Structure** - see section 5.4.18.1 DD18.1 - Data Storage Structure on page 230

5.3.2.5 TD2.5 - Estimate Space Requirements

The disk space required for the design area is calculated from the standard rules for the database management system to be used.

The space required for the first implementation area is calculated. Then the extra space required for each subsequent implementation area is calculated.
The calculations must take into account the expected growth in number of entities of each type. This was estimated during Business Area Analysis (BAA) in the Analyse Distribution task. The archiving approach to be adopted needs to be known before this step can be undertaken. The archiving procedures should have been designed during BSD.

This step creates the following deliverables:

- **Data Storage Space Specification** - see section 5.4.18.2 DD18.2 - Data Storage Space Specification on page 230

5.3.2.6 TD2.6 - Design Buffering

Most database management systems and operating systems provide some control over the organisation of the buffers used temporarily to hold data store blocks being written to secondary storage or being read from secondary storage. Buffer design can be crucial to system performance, and buffer tuning provides one of the simplest ways of improving system performance.
Buffer design may be delayed until it's time to construct the data stores. It is included here as a separate subtask to emphasise that it is an essential part of TD and that enough time is allocated for its completion. If the system loading shows high variance between different times of the day, different buffer arrangements should be designed for various periods of the day.

This step creates the following deliverables:

- **Buffer Design** - see section 5.4.18.3 DD18.3 - Buffer Design on page 230

5.3.2.7 TD2.7 - Design Non-Database Files

The files that do not form a part of the shared pool of data are designed (e.g., intermediate files, stream files, temporary files and data collection files).

5.3.2.8 TD2.8 - Identify System Performance Problems

A formal analysis of system performance will be carried out later in the Assess Performance task, but it is useful, even at this stage, to spot serious performance problems and adjust the design accordingly. If in doubt about whether there is a performance problem, or whether you have found a solution, do not attempt to change the design yet.

All data structure modifications that move away from the preliminary data structure design will reduce the long-term flexibility of the enterprise's data resource. Moving away from the structure generated from the ERD, by the CASE tool is discouraged. If a different structure is needed, change the CASE model and generate the required structure.

Situations that may cause performance problems include:

- An on-line procedure step that performs a large number of database actions (more than twenty) on each execution
- An index that is subjected to a heavy load of creates and deletes
- A procedure that needs to be executed more than once a second during the peak hour
- A specific record or linkage occurrence that is updated more than ten times per second during some period of the day
- A batch program that updates the database, which is concurrently updated on line

5.3.2.9 TD2.9 - Design On-line Data Storage Recovery

Define the procedures for recovering data in an on-line environment.

This step updates the following deliverables:

- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209
- **Procedure Dependency Diagram** - see section 5.4.1.2 DD1.2 - Procedure Dependency Diagram on page 209
- **Data Flow Diagram** - see section 5.4.1.5 DD1.5 - Data Flow Diagram on page 210

5.3.3 TD3 - Develop Program Specifications

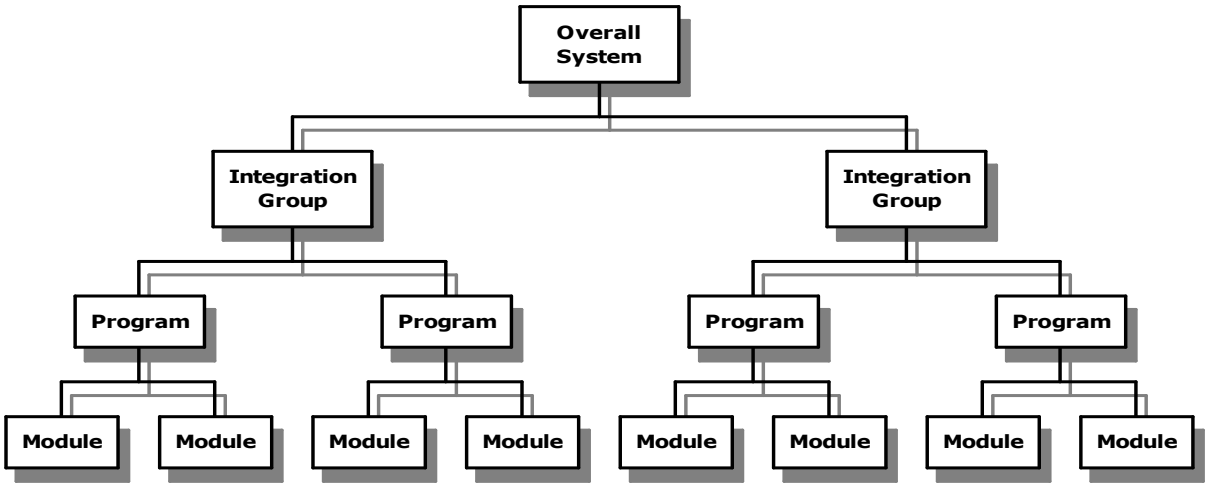
5.3.3.1 TD3.1 - Define Integration Groups

An Integration Group is a group of programs which use the same data stores, formed together in order to be scheduled, developed and tested as a whole.

This step creates the following deliverables:

- **Integration Group Definition** - see section 5.4.19.1 DD19.1 - Integration Group Definition on page 231

Schematically, the relationship between the overall system being developed, integration groups, programs and modules is:



5.3.3.2 TD3.2 - Define Programs and Modules

Decide structured approach

First, establish the principles of how particular functions or procedures will be developed. Determine the options for the various languages available for each type of function.

Define programs

Map each procedure to a program, taking into account the constraints imposed by the project standards. If necessary, use more than one program, particularly if the procedure fulfils more than one type of function. Note that this program is recorded as a module in the information library. Identify the programming language in which the module should be written. If component modules are identified in the next subtask (i.e., Define Component Modules), then this module is the module that collects the component modules.

This step updates the following deliverables:

- **Program Specification** - see section 5.4.17.1 DD17.1 - Program Specification on page 228

Define component modules

Map each elementary procedure step to a module. Note that the same module may be included in more than one program. Also identify the language in which the module should be written.

This step updates the following deliverables:

- **Module Specification** - see section 5.4.17.2 DD17.2 - Module Specification on page 229

Construct the software structure derivation matrix

Construct the software structure derivation matrix to show how the program and modules were derived from the procedures and elementary procedure steps. Use this matrix to check for completeness.

This step creates the following deliverables:

- **Software Structure Derivation Matrix** - see section 5.4.17.3 DD17.3 - Software Structure Derivation Matrix on page 229

Define common logic modules

Do not consider database accesses as worthy of independent modules. However, if an error path, integrity rule or algorithm is complex, consider it a candidate for a common logic module.

Apply two tests for each error path, integrity rule and algorithm. First, determine if it is applied more than once in the business system or if it is likely to be applied in future business systems. Second, determine if it is complex. If it fulfils both criteria, a common logic module is required. Add the details to the software hierarchy diagram.

This step creates the following deliverables:

- **Software Hierarchy Diagram** - see section 5.4.17.4 DD17.4 - Software Hierarchy Diagram on page 229

Construct software hierarchy diagram

Record programs and modules on a software hierarchy diagram in preparation for the next tasks.

This step updates the following deliverables:

- **Software Hierarchy Diagram** - see section 5.4.17.4 DD17.4 - Software Hierarchy Diagram on page 229

5.3.3.3 TD3.3 - Design Programs and Modules

Select a module

Within an integration unit, specify modules from the bottom up. This ensures that all requirements are known when specifying the interfaces between modules. Select the module from the software hierarchy diagram, taking the lowest-level module not yet specified. Ascertain the chosen language by referring to the software structure derivation diagram.

Identify the steps covered by the module

Refer to the software structure derivation diagram to ascertain the steps covered by a module. Obtain the action diagrams and screen and report layouts describing these steps.

Ascertain which parts are covered by called modules

Because of the bottom-up specification sequence, all modules called within the module currently being specified will already have been specified. The type of call is determined from the called module specification. Replace any part of the action diagram covered by a called module by the called-module-name.

This step updates the following deliverables:

- **Module Specification** - see section 5.4.17.2 DD17.2 - Module Specification on page 229

Alter record layout actions

The procedure action diagrams were based on the preliminary data structure design. Record actions need to be brought into line with the current data structure design (i.e., the refined data structure and/or perceived data structures).

This step updates the following deliverables:

- **Refined Data Structure Diagram** - see section 5.4.5.2 DD5.2 - Refined Data Structure Diagram on page 215

Specify how the module is called

Select the method for calling the module.

Specify the interface for calling the module

Every field used within a module must be either completely processed within the module or transferred to and/or from the module.

For data transferred to and/or from the module, record a list of fields (or field groups) to specify the interface. Any modules calling this module will generally use the same list.

Specify additional program detail

If the module is a program, the following additional steps are required.

- Define objects required by the DBMS

Program specification statements are subsets of the perceived data structures. They must be restricted to the processing requirements of the called modules. Equivalent objects may be required for other database management systems.

- Prepare record action list

A record action list for the program shows the number of times a record is likely to be accessed during one execution of the program.

The method for preparing the record action list depends on the method used to turn procedures and elementary procedure steps into programs and modules, and on the format of the specifications, which depends on the language used. Frequencies and volumes were recorded for procedures, elementary procedure steps and records in the BSD phase of the design stage. It is easy to follow the associations in the information library to ascertain to which step a record action belongs.

This step updates the following deliverables:

- **Program Specification** - see section 5.4.17.1 DD17.1 - Program Specification on page 228

Specify interfaces to called modules

The interface requirements for a module that this module calls has already been specified, if the specification sequence has been adhered to. Ensure that all the fields required by the called module are available. Use the same names for fields as those specified for the called module.

Replace the "external step box" within the action diagram with the details of the interface and a USE statement.

This step updates the following deliverables:

- **Module Specification** - see section 5.4.17.2 DD17.2 - Module Specification on page 229

Produce draft specification

Produce a specification in line with the agreed-upon standards. It must either contain or refer to all the deliverable information.

This step updates the following deliverables:

- **Module Specification** - see section 5.4.17.2 DD17.2 - Module Specification on page 229

Verify and produce final specification

Inspect the specification and make necessary corrections to ensure that:

- It is understandable
- It is complete
- It conforms to standards
- The design accurately reflects the data structures and screen and report layouts
- The design accesses the database and files in a logical and feasible manner
- Create, delete and update actions do not break integrity rules
- The resulting module, when collated with its called modules, will be testable
- All errors are handled in a user-friendly manner according to project standards

5.3.3.4 TD3.4 - Conduct a Program/Module Walkthrough

This step updates the following deliverables:

- **Module Specification** - see section 5.4.17.2 DD17.2 - Module Specification on page 229

5.3.4 TD4 - Prepare for Testing

5.3.4.1 TD4.1 - Define Integration Groups

Identify integration groups

Programs that use the same data stores are grouped to form units that can be discretely tested to a high level. For instance, programs that create, modify, delete and report on a particular record type form a good integration group. If possible, choose the integration groups so that a common logic module is in only one group. Add the details to the software hierarchy diagram.

This step creates the following deliverables:

- **Integration Group Definition** - see section 5.4.19.1 DD19.1 - Integration Group Definition on page 231

Verify the software hierarchy diagram

For clarity, redraw the software hierarchy diagram, clustering the programs belonging to the same integration group. Verify that all modules belong to programs and that every program belongs to an integration group.

This step updates the following deliverables:

- **Software Hierarchy Diagram** - see section 5.4.17.4 DD17.4 - Software Hierarchy Diagram on page 229

Estimate effort for design and construction

Estimating depends on many factors, such as language, software, and testing facilities. Estimating methods are not detailed here.

Use the software hierarchy diagram to record estimates of actual time for design, construction and integration testing for each program and module. This is used in order to ascertain the following:

- Actual time for design of an integration group
- Actual time for generation of an integration group
- Actual time for integration testing of an integration group
- Elapsed time for integration testing of an integration group

This step creates the following deliverables:

- **System Test Plan** - see section 5.4.20.1 DD20.1 - System Test Plan on page 231

Devise the integration group dependency diagram

The easiest way to test a system, particularly a database/online system, is to use the system to construct the data. A part of the system that has been fully tested can be used with confidence to load test data for subsequent parts of the system. Data store construction has identified the basic building blocks upon which the system rests, as has the division of the business system into implementation areas at the end of BSD.

Identify the data requirements for each integration group and construct an integration group dependency diagram to show this dependency, entering the elapsed time estimated for integration testing.

This step creates the following deliverables:

- **Integration Group Dependency Diagram** - see section 5.4.19.2 DD19.2 - Integration Group Dependency Diagram on page 231

Identify critical factors in design and construction

The integration group dependency diagram shows the sequence in which designed and constructed integration groups should be produced for testing. By backward projection of actual times estimated for design and generation of integration groups, identify critical steps in the design and construction tasks.

5.3.4.2 TD4.2 - Define Integration Test Requirements

Select an integration group

The integration group dependency diagram shows when an integration group should be tested, compared with other groups. If this approach is used, the system can be used to construct the data needed for testing.

It is not necessary to devise the test conditions in this order, but it is usually prudent to do so for scheduling.

Define test conditions

Most test conditions must be derived from the action diagrams describing the procedures in the BSD because these give a description of the provided system.

Assign to lowest-level module

For each test condition, determine the lowest-level module to which the test could possibly be applied.

Reassign according to testability

Starting at the lowest-level modules, check each test condition for ease of testability. If it is not easily tested with the simplest testing facilities available, transfer the test condition to its calling module.

If the current module is a common logic module, transfer all the tests to the same calling module.

Select test conditions for a module's initial test

Start by assigning all the test conditions to the module's detailed tests. Unless inspecting a module as an initial test is sufficient, some of the conditions must be reassigned to the initial test.

Group into test cycles

All the conditions have now been assigned to a module and a test type within it. The test conditions are grouped into test cycles so that a test cycle contains only test conditions of the same test type and those assigned to the same module.

Construct integration test cycle dependency diagram

Start constructing the integration test cycle dependency diagram for an integration group, based on the software hierarchy diagram.

This step creates the following deliverables:

- **Test Cycle Dependency Diagram** - see section 5.4.20.5 DD20.5 - Test Cycle Dependency Diagram on page 234

Estimate effort for testing a test cycle

Estimate the effort and minimum elapsed time to complete a test cycle for each cycle on the integration test cycle dependency diagrams, and add the estimates to the diagram.

Revise estimates on integration group dependency diagram

Examine the integration test cycle dependency diagram and total the elapsed time along the shortest path through the diagram. Revise the estimates on the integration group dependency diagram. This diagram shows the integration group dependencies and the elapsed time estimates for integration testing. The system may be used to construct the data needed for testing.

For more information about this diagram, refer to the Devise the Integration Group Dependency Diagram subtask in the previous task (i.e., Define Integration Groups).

This step updates the following deliverables:

- **Integration Group Dependency Diagram** - see section 5.4.19.2 DD19.2 - Integration Group Dependency Diagram on page 231

5.3.4.3 TD4.3 - Finalise Design for Implementation

Revise implementation strategy

An implementation strategy was chosen in the EM stage and revised in the BAA stage and again in the BSD phase of the design stage. A review is now undertaken to revise the strategy to take account of decisions made in the BSD phase and to deal with the transition between one implementation area and another.

This step updates the following deliverables:

- **Implementation Strategy** - see section 4.3.7.4 AD7.4 - Implementation Strategy on page 132

Identify interface data:

- Identify input to the implementation area -

Identify the data required as input to the implementation area from existing systems. The output from this subtask is a list of the fields in the interfaces. Every field in the implementation area must be put into a category to reflect its expected status at the time of implementation. If implementation is being phased, fields that are implemented in some locations but not in others must be considered.

Categories of fields

Every field in the implementation area fits into one of the following categories:

- Input directly by human resources from data external to existing systems
- Derived from data in systems to be replaced by the project
- Already present on the databases from a system implemented from the ADM
- Already present on the databases from previously implemented phases of this system
- Data necessary to enter the system

- Identify input to existing systems

Identify the data required as input to existing systems from the implementation area. Every item established (created, updated or deleted) in the area must be considered for its impact on existing systems and must be described on the interface field list.

This step updates the following deliverables:

- **Module Specification** - see section 5.4.17.2 DD17.2 - Module Specification on page 229

Design implementation algorithms

For each field in the interface field input, an implementation algorithm from which the field can be derived must be designed.

Design bridging procedures

This subtask involves designing procedures to cause the bridging to happen. Every field in the implementation area required in bridging has been identified previously, so it is now necessary to ascertain the frequency of bridging from one system to another. This, together with the data, will indicate the types of procedure required. For each bridging procedure, produce the full range of design documentation required and incorporate all the relevant implementation algorithms. All procedures designed in this stage must be developed with standard Application Development Methodology (ADM) techniques and must be included within the scope of the project and the specification.

This step updates the following deliverables:

- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

Design conversion procedures

This subtask is concerned with designing procedures to enable existing data (held in files or databases that are to be replaced by the new system) to be converted into a form suitable for loading into the new databases. These procedures may include unique load programs to achieve the population of the new databases from each source. Every field in the implementation area for which conversion is required has been identified previously, and the algorithms required will have

been defined. Many of the algorithms will also be common to the bridging procedures.

This step updates the following deliverables:

- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

Design operational control procedures

Design procedures to create manage the load libraries of executable modules for the testing environment - operational control procedures.

This step updates the following deliverables:

- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

5.3.4.4 TD4.4 - Refine Training Requirements

Training requirements were outlined in a series of matrices during the BSD "Design for implementation" task. The current subtask finalises these requirements. The starting point, however, may need to be a subset of the matrices particular to an implementation area.

- Update the procedure/technical skill matrix -

Update the procedure/technical skill matrix prepared during BSD that shows the technical skills required to perform each procedure being implemented.
This step updates the following deliverables:

- **Procedure/Technical Skill Matrix** - see section 5.4.4.1 DD4.1 - Procedure/Technical Skill Matrix on page 212

- Update the procedure/organisational role matrix -

Update the procedure/organisational role matrix prepared during BSD that shows the association of the procedures being implemented and the organisational roles of the employees who will perform and supervise their operation. The organisational roles are referred to by the standard job titles used in the organisation.

This step updates the following deliverables:

- **Procedure/Organisational Role Matrix** - see section 5.4.4.2 DD4.2 - Procedure/Organisational Role Matrix on page 213

- Update the skill/organisational role matrix

Update the skill/organisational role matrix prepared during BSD that shows the skills required by each type of employee to operate a new system. The matrix is updated from the procedure/technical skill and the procedure/organisational role matrices revised earlier.

The skills needed are shown by the letter N in the appropriate cell in the skills/organisational role matrix. Determine whether the employee already possesses any of the required skills, and if this is the case, change the cell value from N to Y.

Label the rows and columns of the matrix

- Label the rows with procedure names for the procedural skills, or technical skill names for the technical skills
- Label the columns with the organisational roles

Fill-in the cells

The cell entries are derived from the procedure/organisational role and procedure/technical skill matrices.

For each organisational role (column):

- Consider each procedural skill and decide whether it is required by a person with this organisational role; the answer is given in the procedure/organisational role matrix.
- Consider each technical skill and decide whether it is required by a person with this organisational role. Derive the answer from the procedure/technical skill matrix by checking whether any of the procedures listed for this organisational role requires the skill in question.

This step updates the following deliverables:

- **Skill/Organisational Role Matrix** - see section 5.4.4.3 DD4.3 - Skill/Organisational Role Matrix on page 213

- Update the Training Requirements Matrix

Update the training requirements matrix prepared during BSD that shows the skills required by each employee to operate the new procedures, and whether or not the employees already have the skills. The rows of the matrix are the same skills as those used in the skill/organisational role matrix. The columns of the matrix are employee names. A cell value of Y shows that the employee requires and already has the skill. A cell value of N shows that the employee requires but does not presently have the skill.

First determine the employees who will operate or supervise the new systems procedures and their organisational roles. (An employee may have more than one organisational role.) For each employee, indicate on the matrix the skills required.

This step updates the following deliverables:

- **Training Requirements Matrix** - see section 5.4.4.4 DD4.4 - Training Requirements Matrix on page 215

5.3.4.5 TD4.5 - Refine Rollout Plan

Rollout is discussed in detail in the implementation stage and in the System Configuration Control technique. The plan was outlined in BSD, and it now needs to be refined to reflect any changes necessitated by TD or by the division of the design area into implementation areas.

This step updates the following deliverables:

- **Rollout Plan** - see section 5.4.2.1 DD2.1 - Rollout Plan on page 211

5.3.4.6 TD4.6 - Design Operations Procedures

Design security and control procedures

Security and control procedures may be an integral part of the existing computer environment. General procedures may therefore also apply to any new environment installed for a new business system. The new business system must incorporate these general procedures in its design, but it may also require the introduction of new procedures, particularly in the areas of logical access control and system integrity. The level of security to be provided by these procedures has been identified in BSD.

Identify an individual within the user department who will have responsibility for security in the new business system. This person, who should be a senior user, will eventually have responsibility for authorising new users and reviewing security reports. The person is identified at this stage so that the design and construction of the business system security processes can be supervised by a responsible individual.

This step creates the following deliverables:

- **Security And Control Procedures** - see section 5.4.21.1 DD21.1 - Security and Control Procedure Definition on page 236

Design security modules

Security features are normally included in the business system when the required security level is not available from a system product or when the integrity of the system software product may be compromised. For example, if a very high level of internal security is required and the system software product security must be administered by the computer system rather than by business-unit personnel, it

may be necessary to include an additional level of security in the business system.

Security facilities within the business system must be designed in separate modules to business process code. Ideally, they should fit independently into the software structure, and the structure should be designed in order to prevent operation without the security module. This may necessitate some rework of the software structure. All action diagrams must be amended to accommodate the security requirements, which must also be tested.

- Attempted Security Violations -

When an attempt is made to violate security, the attempt must be reported to the security officer. It is unlikely that an individual security officer will be on duty 24 hours a day, 365 days a year. Clerical instructions must be written to be used by any person to whom the computer may report an attempted security violation. Instructions should include how to recognise a security violation and what to do about it. The person concerned must have these instructions.

This step creates the following deliverables:

- **Security Modules Specification** - see section 5.4.21.2 DD21.2 - Security Module Specification on page 236

Design contingency procedures

Contingency procedures are effective when the computer system has failed or has experienced a partial failure. The level of contingency planning depends on the criticality of the business system. Critical business processes require that mirror images of the data and processes prevent any system down time. Systems not critical to daily business processes may await recovery of the computer system.

When designing contingency procedures, the technical design team must determine the criticality of the business system in terms of the acceptable time the business may be without the system. If the business must always have the system available, then the processing and data must be duplicated and on-line constantly. If only a portion of the business system is critical, then contingency plans should be developed for only that portion. Recovery plans should be developed for the remaining portion of the system that did not operated during the system failure.

This step updates the following deliverables:

- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

Select recovery techniques

Designing and constructing recovery software are complex and costly activities, and should be unnecessary. Transaction logging and recovery should be a

feature of the database or teleprocessing software. Both roll-forward and rollback recovery should be available.

- Backup - Procedures have been devised for maintaining backup copies.

Clerical instructions must be produced to document the parts of the procedures not performed by the computer. They include such information as when to make a backup copy, what to make it of, and where to keep it. It is likely that these instructions will detail in which safe and/or bank the copy is kept and how to put it there.

- Recovery - Clerical instructions must be written to address the following topics:

- How to recognise that recovery is required and whom to tell
- How to work with personnel concerned with fallback and who these people are
- How to identify which backup copies are required (which versions of what files)
- How to get the copies from the safe or bank (subject to security measures)
- What computer runs to perform
- How to recognise successful recovery and whom to tell
- What procedures to adopt to return to normal running (working with fallback personnel)

- Fallback - The following topics must be addressed in clerical instructions:

- How to recognise an error
- How to tell that recovery is required
- Whom to inform to have recovery initiated
- Which existing procedures to discontinue while awaiting recovery
- What fallback procedures to use
- How to change from existing procedures to fallback
- How to change from fallback to existing procedures
- What additional procedures are required as a result of recovery, when existing procedures are resumed

Identify backup points

The operation of a business system can be divided into a number of processing cycles. A processing cycle is normally time based (e.g., daily, weekly, monthly and annual backups). The objective of contingency procedures is to recover system data after a failure.

To do this, you must repeat the processing in a cycle that rebuilds a copy of the data. It is desirable to be able to offer more than just the last cycle. For example, in a system with daily, weekly, half-yearly and annual cycles, you may keep daily backups for the last ten days, weekly backups for the last six weeks, and the last half-yearly and yearly backups.

Select performance monitoring measures

To meet the objectives for which it was designed, the business system must maintain an effective operating level. System availability, throughput and response time must be maintained at the required level. To maintain the required operating level, the performance of the system must be monitored and recorded constantly. Performance can be reviewed in terms of computer performance, business system performance and system operator actions.

The most important system performance can be measured in terms of how many times users experience system errors and how many times users request help. Analysis of these measures may intimate adjustments to user training or the system to provide more effective support.

This step creates the following deliverables:

- **Performance Monitoring Measures** - see section 5.4.22.1 DD22.1 - Performance Monitoring Measure on page 237

Design Performance Monitoring Procedures

Performance monitoring should be designed in separate modules, and it should be possible to switch it off in certain circumstances.

Manual procedures must be designed to record and analyse the performance measures. Plotting key performance indicators can show dramatic changes or trends that will affect response time before they are noticed by the terminal user.

This step creates the following deliverables:

- **Performance Monitoring Procedures** - see section 5.4.22.2 DD22.2 - Performance Monitoring Procedure Definition on page 237

Design operating instructions

The business system will consist of on-line modules and batch-process modules that may operate at any number of locations. To assure that the system and its data are fully supported when the business requires support, a schedule of operations is required. The technical design team must:

- Determine the required schedule of availability of system support required by the business
- Determine the allowable lag in data concurrency as determined by the business cycle
- Determine the schedule of data-bridging updates, batch processing and on-line processing that satisfies the (above stated) business requirements
- Determine the schedule of database backup operations compatible with the operating schedule and provide the required data protection

5.3.5 TD5 - Assess Performance

Decide on assessment technique to be used

Decide which assessment technique to use to assess the software performance.

List the procedures to be assessed

In a typical business system, a few procedures account for a large proportion of the system resources. To reduce the effort required for performance assessment, confine the detailed study to the procedures most likely to present problems, while allowing for the resources used by the others. List the procedures, the reasons for their selection and the performance objectives.

- Select the procedures to be assessed -

Make a list of procedures for assessment. Select the procedures to be assessed based on the following characteristics:

- Those with a high degree of activity, such as a large number of:

  Step executions per hour for an on-line procedure
  Records processed in each step in a batch procedure
  Print lines in a print procedure

- Those with a large number of database actions per transaction (e.g., more than ten)
- Those with complex database search patterns
- Those with long absolute run times (e.g., more than one hour for a batch procedure)
- Those with exceptionally short performance objectives (e.g., less than one second system response time)
- There are a large number of transactions per hour for an on-line procedure

Predict system performance

Estimate response times and other essential performance measures, using the technique decided on earlier. If the analytic technique is implemented, use an extended record action list.

Identify performance problems and bottlenecks

For each step or procedure that fails to meet its response time objectives or that is suspect, try to identify the source of the problem. Record conclusions in a performance assessment summary.

Recommend solutions

After the sources of bottlenecks and problems have been identified, the performance assessment team adds recommended solutions to the performance assessment summary.

This step creates the following deliverables:

- **Performance Assessment Summary** - see section 5.4.22.3 DD22.3 - Performance Assessment Summary on page 237

5.3.6 TD6 - Design for System Testing

5.3.6.1 TD6.1 - Define the Scope of System Tests

Prepare a high-level list of the tests needed in each of the following test areas:

- System test environment testing
- Integration group retesting
- Merging of the integration groups
- Breakdown testing
- Volume testing
- Bullet-proofing
- Full procedure testing

5.3.6.2 TD6.2 - Design System Test Cycles

The testing requirements listed in the previous subtask, Define the Scope of System Tests, should now be divided or merged into manageable test sessions (i.e., test cycles). For system testing, it is preferable to identify test cycles before test conditions, unlike integration testing.

Draw a draft system test cycle dependency diagram to show the sequence in which the cycles are to be performed and the purpose of each test cycle. Try to use data created by one test cycle as input data to another test cycle.

This step creates the following deliverables:

- **System Test Cycle Dependency Diagram** - see section 5.4.20.2 DD20.2 - System Test Cycle Dependency Diagram on page 234

5.3.6.3 TD6.3 - Specify Test Conditions for System Testing

Define the test conditions to be tested within each test cycle. Define both the condition and the outcome.

The elapsed time and effort (person-days) required to conduct each test cycle can then be estimated and added to the dependency diagram. Now draw the formal dependency diagram.

Remember that because the modules have already undergone integration testing, it is not necessary to reproduce every integration test condition again as a system test condition. Show that the normal routes through each procedure work correctly and that two or three error situations are properly handled.

This step updates the following deliverables:

- **System Test Cycle Dependency Diagram** - see section 5.4.20.2 DD20.2 - System Test Cycle Dependency Diagram on page 234

5.3.6.4 TD6.4 - Define Benchmark Testing

Define the objectives and performance criteria for a benchmark test of the system.

This step creates the following deliverables:

- **Benchmark Test Plan** - see section 5.4.20.6 DD20.6 - Benchmark Test Plan on page 235

5.3.6.5 TD6.5 - Define Acceptance Test Cycles

Each organisational unit that is to accept the system sets its own acceptance criteria and devises its own tests. The acceptance criteria have already been laid down previously in the task Define Technical Context and Requirements. Now the tests are actually specified.

The tests are defined in the form of test cycles. A test cycle is a practical, on-line testing session or a batch test run or perhaps an entirely clerical trial procedure. A test cycle normally includes many valid transactions plus transactions with user errors. Other test cycles can test only one severe error situation, because that error will halt the system.

Each participating organisational unit constructs a draft test cycle dependency diagram to show the sequence in which the tests will be performed. The boxes are labelled with the purpose of each test.

This step creates the following deliverables:

- **Acceptance Test Requirement** - see section 5.4.20.3 DD20.3 - Acceptance Test Requirement on page 234

- **Acceptance Test Cycle Dependency Diagram** - see section 5.4.20.4 DD20.4 - Acceptance Test Cycle Dependency Diagram on page 234

5.3.6.6 TD6.6 - Specify Test Conditions for Acceptance Tests

Define the test conditions to be tested within each test cycle. Define both the condition and the outcome of each test condition.

After doing this, estimate the elapsed time and effort (person-days) required to conduct each test cycle and add the estimates to the dependency diagram. Now draw the formal acceptance test cycle dependency diagram.

This step creates the following deliverables:

- **User Acceptance Test Plan** - see section 5.4.20.7 DD20.7 - User Acceptance Test Plan on page 235

This step updates the following deliverables:

- **Acceptance Test Requirement** - see section 5.4.20.3 DD20.3 - Acceptance Test Requirement on page 234
- **Acceptance Test Cycle Dependency Diagram** - see section 5.4.20.4 DD20.4 - Acceptance Test Cycle Dependency Diagram on page 234

5.3.7 TD7 - Plan Construction and Implementation Stages

Prepare implementation plan

Prepare the implementation plan that should include a schedule covering the construction and implementation stages.

This step creates the following deliverables

- **Implementation Plan** - see section 5.4.2.4 DD2.4 - Implementation Plan on page 211

Review costs and benefits

Prepare a cost-benefit summary report. The remaining development costs can be accurately estimated from the implementation plan, so a new figure for the total expected development costs can now be produced.

The production costs were estimated at the end of BSD. More reliable production costs can now be estimated, based on the performance assessment summary prepared in the Assess Performance task of TD.

The benefits of installing the implementation area's system were previously quantified in BSD. These benefits should be reviewed, but they should not have changed unless the scope of the system has changed significantly.

Gain management approval

Senior managers review the implementation plan and the cost/benefit summary. As a result, they may give the go-ahead, cancel the project, postpone the project, or seek changes to the project scope or plans.

If senior managers seek changes to the project scope or plans, they should state:

- What they do not approve of in the plan
- Areas they recommend for modification
- The nature of the modification required (e.g., postponement of some less-essential procedure, faster transition to some procedures, less-sophisticated terminals)
- Whether several alternative plans are to be prepared
- Required return on investment

This step updates the following deliverables:

- **Implementation Plan** - see section 5.4.2.4 DD2.4 - Implementation Plan on page 211

Iterate

This subtask is performed only if requested by senior management in the previous subtask, Gain Management Approval.

- The project team investigates the recommendations made by senior management and prepares:
- Proposed modifications to the current design, by repeating earlier BSD and TD tasks
- One or more new implementation plans

If the new plan is approved by management, the design is formally filed in the information library.

This step creates the following deliverables:

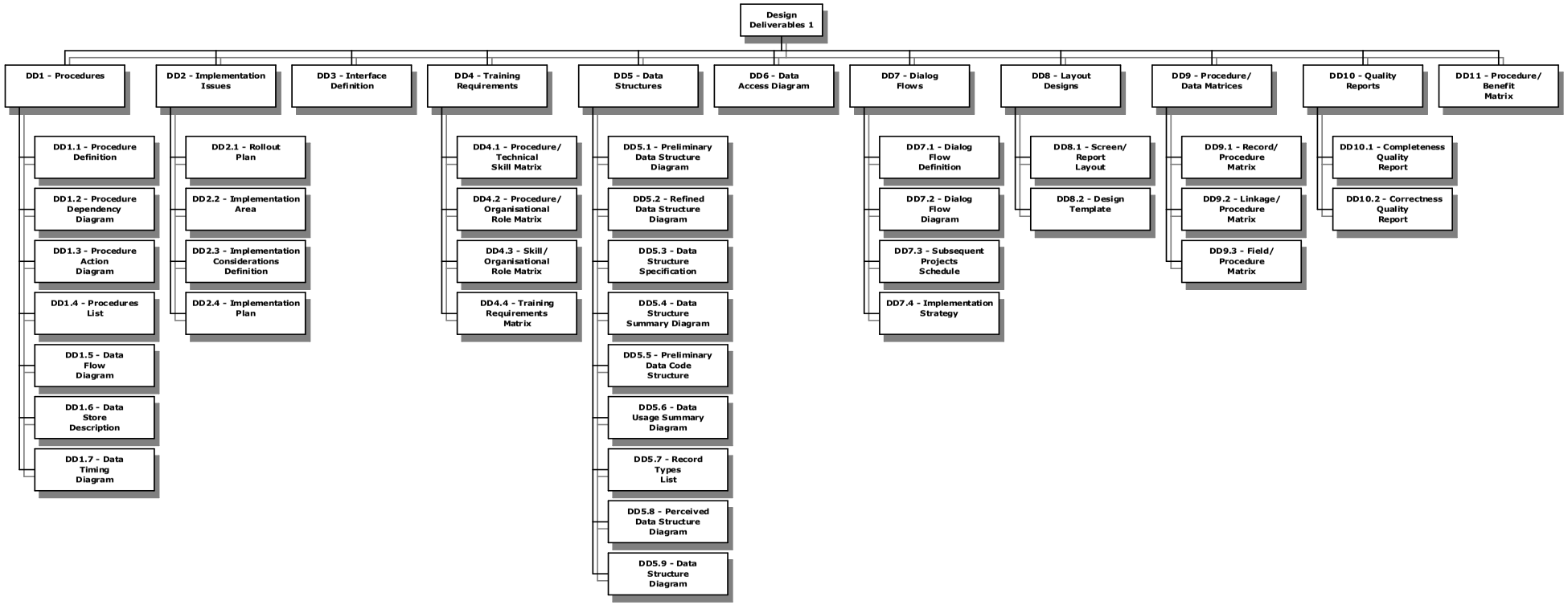
- **Construction Plan** - see section 5.4.23 DD23 - Construction Plan on page 237

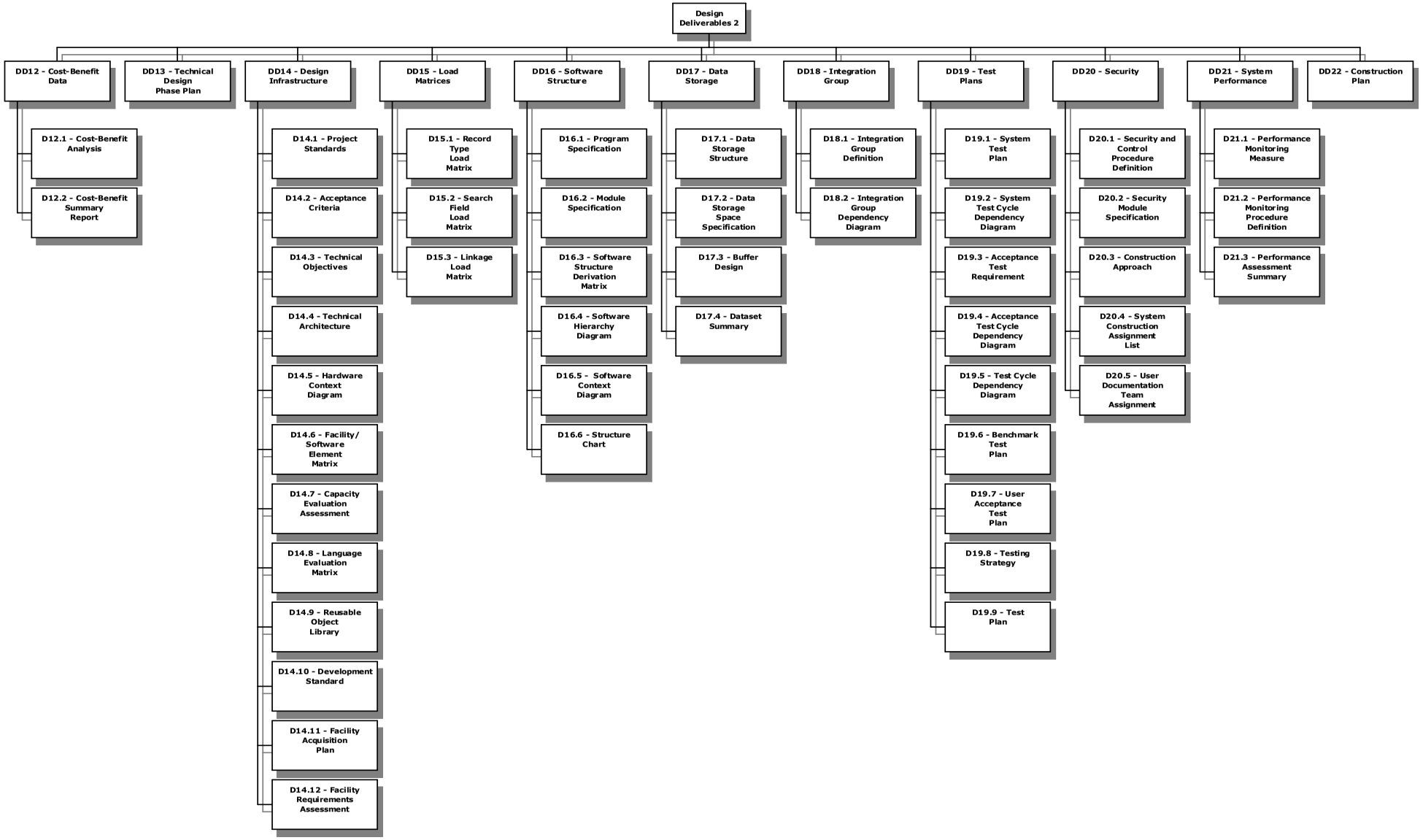
This step updates the following deliverables:

- **Implementation Plan** - see section 5.4.2.4 DD2.4 - Implementation Plan on page 211

5.4 Design Phase Deliverables

Schematically, the structure of the Design Deliverables is:


Design Deliverables 1
DD1 - Procedures
DD1.1 - Procedure Definition
DD1.2 - Procedure Dependency Diagram
DD1.3 - Procedure Action Diagram
DD1.4 - Procedures List
DD1.5 - Data Flow Diagram
DD1.6 - Data Store Description
DD1.7 - Data Timing Diagram
DD2 - Implementation Issues
DD2.1 - Rollout Plan
DD2.2 - Implementation Area
DD2.3 - Implementation Considerations Definition
DD2.4 - Implementation Plan
DD3 - Interface Definition
DD4 - Training Requirements
DD4.1 - Procedure/ Technical Skill Matrix
DD4.2 - Procedure/ Organisational Role Matrix
DD4.3 - Skill/ Organisational Role Matrix
DD4.4 - Training Requirements Matrix
DD5 - Data Structures
DD5.1 - Preliminary Data Structure Diagram
DD5.2 - Refined Data Structure Diagram
DD5.3 - Data Structure Specification
DD5.4 - Data Structure Summary Diagram
DD5.5 - Preliminary Data Code Structure
DD5.6 - Data Usage Summary Diagram
DD5.7 - Record Types List
DD5.8 - Perceived Data Structure Diagram
DD5.9 - Data Structure Diagram
DD6 - Data Access Diagram
DD7 - Dialog Flows
DD7.1 - Dialog Flow Definition
DD7.2 - Dialog Flow Diagram
DD7.3 - Subsequent Projects Schedule
DD7.4 - Implementation Strategy
DD8 - Layout Designs
DD8.1 - Screen/ Report Layout
DD8.2 - Design Template
DD9 - Procedure/ Data Matrices
DD9.1 - Record/ Procedure Matrix
DD9.2 - Linkage/ Procedure Matrix
DD9.3 - Field/ Procedure Matrix
DD10 - Quality Reports
DD10.1 - Completeness Quality Report
DD10.2 - Correctness Quality Report
DD11 - Procedure/ Benefit Matrix


Design Deliverables 2
DD12 - Cost-Benefit Data
D12.1 - Cost-Benefit Analysis
D12.2 - Cost-Benefit Summary Report
DD13 - Technical Design Phase Plan
DD14 - Design Infrastructure
D14.1 - Project Standards
D14.2 - Acceptance Criteria
D14.3 - Technical Objectives
D14.4 - Technical Architecture
D14.5 - Hardware Context Diagram
D14.6 - Facility/ Software Element Matrix
D14.7 - Capacity Evaluation Assessment
D14.8 - Language Evaluation Matrix
D14.9 - Reusable Object Library
D14.10 - Development Standard
D14.11 - Facility Acquisition Plan
D14.12 - Facility Requirements Assessment
DD15 - Load Matrices
D15.1 - Record Type Load Matrix
D15.2 - Search Field Load Matrix
D15.3 - Linkage Load Matrix
DD16 - Software Structure
D16.1 - Program Specification
D16.2 - Module Specification
D16.3 - Software Structure Derivation Matrix
D16.4 - Software Hierarchy Diagram
D16.5 - Software Context Diagram
D16.6 - Structure Chart
DD17 - Data Storage
D17.1 - Data Storage Structure
D17.2 - Data Storage Space Specification
D17.3 - Buffer Design
D17.4 - Dataset Summary
DD18 - Integration Group
D18.1 - Integration Group Definition
D18.2 - Integration Group Dependency Diagram
DD19 - Test Plans
D19.1 - System Test Plan
D19.2 - System Test Cycle Dependency Diagram
D19.3 - Acceptance Test Requirement
D19.4 - Acceptance Test Cycle Dependency Diagram
D19.5 - Test Cycle Dependency Diagram
D19.6 - Benchmark Test Plan
D19.7 - User Acceptance Test Plan
D19.8 - Testing Strategy
D19.9 - Test Plan
DD20 - Security
D20.1 - Security and Control Procedure Definition
D20.2 - Security Module Specification
D20.3 - Construction Approach
D20.4 - System Construction Assignment List
D20.5 - User Documentation Team Assignment
DD21 - System Performance
D21.1 - Performance Monitoring Measure
D21.2 - Performance Monitoring Procedure Definition
D21.3 - Performance Assessment Summary
DD22 - Construction Plan

5.4.1 DD1 - Procedures

5.4.1.1 DD1.1 - Procedure Definition

Procedure definitions are short descriptions of a system procedure. These descriptions include the name of the procedure, several sentences describing how the activity is done and what is accomplished, an indication of the technique employed (e.g., manual or automated, batch or interactive), and frequency/timing constraints.

Special Types of Procedure Definitions

Bridging procedure definition

Specifications are produced of procedures necessary to link existing system data structures to the new data structures as well as specifications necessary to maintain the bridge (e.g., changes in code structure and values).

Conversion procedure definition

Specifications are produced of the procedures necessary to convert from or to existing system data structures and new data structures.

- Security and Control Procedure Definition
- Performance Monitoring Procedure Definition

5.4.1.2 DD1.2 - Procedure Dependency Diagram

A diagram that shows the interrelationship of procedures and why, for each procedure, an execution may depend upon the prior execution of other procedures.

5.4.1.3 DD1.3 - Procedure Action Diagram

A formal representation of the logic of a procedure, containing a series of action statements, which describe actions to data (i.e., entity types, relationship types or attribute types) or control the execution of the procedure. The diagram builds on the process action diagram by using an extended version of the same syntax and the same constructs. The process actions can be used directly within the procedure.

A procedure action diagram is a rigorous and detailed definition of the logic in a procedure or procedure step. It includes:

- Data views
- Procedure actions

- Reference to business processes

Dialog and Layout Actions

The procedure action diagram must comprehend the dialog and layout design. In dialog design, the flow between procedure steps is defined. The detailed logic for each step is defined in a procedure action diagram, and the exit states that control the flows are set within the procedure step.

Each field on a layout is defined to have a set of video properties that govern how it is displayed. These can be modified from within the procedure to highlight certain fields for the next redisplay only. For each field in a screen layout, a standard set of properties is defined to indicate that the value in the field is in error. The procedure will also have control over the position of the cursor.

5.4.1.4 DD1.4 - Procedures List

5.4.1.5 DD1.5 - Data Flow Diagram

A diagram that shows the data object types output by one activity and later used by another activity.

A data flow diagram is a representation of the flow of data into, out of, and between procedures, subsystems or systems. The diagram is similar to the process dependency diagram.

Process dependency diagrams show the dependencies that exist for business processes and do not indicate how the dependencies may be manifested. Data flow diagrams show the flow of actual data between designed business activities and data stores, thus depicting how two activities interact.

In BAA Current Systems Procedure Analysis, a diagram should be constructed for the procedure level, possibly with other diagrams abstracted to show subsystems and systems.

5.4.1.6 DD1.6 - Data Store Description

Description of a data store, containing:

- Definition
- Content in terms of entity types
- Type of storage (i.e., manual or computerised)
- Special requirements (e.g. availability, security, size, etc.)

5.4.1.7 DD1.7 - Data Timing Diagram

A process dependency diagram or data flow diagram annotated to reflect when procedures are executed. The data timing diagram will reflect business cycles involving daily, monthly, annual, and other timing considerations that cannot be represented by a simple sequential procedure dependency diagram or data flow diagram.

5.4.2 DD2 - Implementation Issues

5.4.2.1 DD2.1 - Rollout Plan

Plan detailing the implementation of a system within an organisation.

5.4.2.2 DD2.2 - Implementation Area

A portion of the business' activities, data, organisations and business locations for which a system is designed to be built and implemented.

5.4.2.3 DD2.3 - Implementation Considerations Definition

The Implementation Considerations Definition lists issues that potentially impact the development and cutover of the proposed application. These issues are identified in an implementation planning workshop by user participants.

5.4.2.4 DD2.4 - Implementation Plan

The Implementation Plan describes the process for converting the new application into production status. It outlines the business activities required to: prepare for cultural changes; design Conversion and Production procedures; create user and technical documentation and user training.

Relation Between Implementation Plan and Project Plan

The planning and design of the implementation aspects of a project are typically addressed in more than one development stage. For example, in the RAD path, ideas are first recorded at the end of user design and are subsequently detailed during rapid construction.

Because a project plan typically addresses only one stage, the implementation plan has been defined as a cross-stage deliverable. When all implementation-related tasks are managed as part of one project, as in RAD, the project plan will completely incorporate the implementation plan.

Contents of Implementation Plan

The implementation plan consists of a description of the sequence in which implementation areas will be tackled, and the reasoning behind these decisions. When necessary, any sequencing of implementation within an implementation area should be described. It can be seen as the Stage Report for a BSD phase of the design stage.

- State the date when the system will be in production (or whether it will be possible to meet the target date set for production)
- Estimate the remaining development costs accurately
- Identify skill shortages
- Plan the acquisition and reorganisation of resources (i.e., people, equipment, office space)

Major Headings

- Construction and implementation tasks
- Start and end date for each task
- Resources required for each task

Tasks Shown

- Environment construction
- Data store construction
- Software generation
- Document generation
- Integration testing
- Integration test review
- System testing
- Acceptance testing
- Implementation phasing (rollout)
- User training
- Post Implementation Assessment
- Components of the Implementation Plan
- Rollout plan
- Implementation strategy
- Resource profile
- Cost and time scales

5.4.3 DD3 - Interface Definition

Interfaces to and from the implementation area under consideration are identified. This deliverable is a list of fields in each interface.

5.4.4 DD4 - Training Requirements

5.4.4.1 DD4.1 - Procedure/Technical Skill Matrix

A matrix that indicates the technical skills needed in carrying out each of a set of procedures

5.4.4.2 DD4.2 - Procedure/Organisational Role Matrix

A matrix that indicates the staff roles to be allocated in carrying out each of a set of procedures.

5.4.4.3 DD4.3 - Skill/Organisational Role Matrix

A matrix that shows the procedural and technical skills needed in each organisational role associated with a designed system.

Roles and Staffing Profiles

IM tasks are assigned to new and existing organisation units in the future information environment. The type and expected number of staff are also given. Profiles are defined for new staff positions. These profiles show:

- Objectives or outline of the staff position
- Responsibility and authority
- Task overview: main steps to be executed
- Skills required: knowledge and experience

Example of Profiles of New IM Roles

Development Coordinator

The development coordinator binds the various development projects together and supports a common development infrastructure.

Specific tasks are:

- To support the development and maintenance of the architectures
- To provide and support common methods and tools
- To provide the means of sharing data
- To coordinate Information Engineering projects
- To assure the quality of Information Engineering projects

Data Administrator

The data administrator develops and maintains an unambiguous and consistent set of definitions of data, activities and their relationships to facilitate common understanding and to provide a consistent framework for information systems within the enterprise.

Specific tasks are:

- To establish and support standards and methods for Information Modelling
- To support the development and maintenance of the Information Architecture
- To maintain and control the central information library, which contains the architectures
- To promote data sharing
- To establish procedures and guidelines for the authorisation, security, and privacy of the data environment

Communications Manager

The communications manager controls the integrated telecommunications environment (voice, text, data, image) within the enterprise and its interface to the outside world.

Specific tasks are:

- To plan and control the use of communications facilities
- To design and maintain a communications infrastructure
- To establish guidelines for the use of communications facilities
- To promote and support the use of telecommunications

Coordinator of End-User Support

The coordinator promotes and supports the use of facilities by end users to develop and use their own information environment.

Specific tasks are:

- To provide facilities for personal computing by end users
- To provide training to end users
- To coordinate the access to data
- To assist users in the use of development and processing facilities

Information Manager

The information manager has a functional responsibility for the information environment as a whole. The manager generally reports to the board or may even be a member of the board.

Specific tasks are:

- To plan and control enterprise modelling and information systems development throughout the enterprise
- To plan and control the use of a common data infrastructure (data as a corporate resource)
- To plan and control the use of a common technical infrastructure (e.g., communications and computer facilities)
- To provide guidelines and procedures to establish and maintain the information environment

5.4.4.4 DD4.4 - Training Requirements Matrix

A matrix that indicates the skills that must be associated with each role in an organisation. On an organisational role level, the matrix is called Skills/Organisational Role Matrix.

5.4.5 DD5 - Data Structures

5.4.5.1 DD5.1 - Preliminary Data Structure Diagram

An initial representation of the eventual organisation of the data for access. This data structure normally represents a fully normalised data model.

5.4.5.2 DD5.2 - Refined Data Structure Diagram

The Entity Relationship Model transformed into a structure that is appropriate for a certain type of data base management system e.g. relational. This data structure usually represents a denormalisation of the data model.

Stages of a Data Structure Diagram

- Preliminary Data Structure Diagram
- Refined Data Structure Diagram
- Perceived Data Structure Diagram - see section 5.4.5.8 DD5.8 - Perceived Data Structure Diagram on page 216

5.4.5.3 DD5.3 - Data Structure Specification

This specification consists of the definitions and properties of all the data structure design objects stored in the information library.

Stages of the Data Structure Specification

- Preliminary Data Structure Specification
- Refined Data Structure Specification
- Perceived Data Structure Specification

Optimised Data Structure and Data Storage Structure

The data structure and data storage structure may be adjusted as a result of performance assessment, in order to remove any performance bottlenecks. When you are satisfied that you have a system design that meets the agreed-upon performance objectives, the data and data storage structures that underpin this design are called "optimised." whether or not these structures were adjusted to bring about a satisfactory predicted performance.

"Optimised" does not imply that the predicted hardware utilisation is minimised; your objective is to provide acceptable performance from a design that mirrors the inherent structure of the business world.

Data Structure

A data structure is a scheme for organising the data required to support one or more business systems. It is independent of the way in which the data are physically arrayed in secondary storage. It is expressed in terms of these object types: data store, record type, linkage, field, entry point and integrity rule.

Data Storage Structure

A data storage structure defines how the data specified in the data structure are to be physically organised. It is expressed in terms of these object types: data set (physical file) and storage mapping.

5.4.5.4 DD5.4 - Data Structure Summary Diagram

A summary version of deliverable Refined Data Structure Diagram.

5.4.5.5 DD5.5 - Preliminary Data Code Structure

5.4.5.6 DD5.6 - Data Usage Summary Diagram

Data usage summary of a Data Structure Diagram. A diagram that shows the frequency of usage in a specified time period and the volumes of objects concerned in either a conceptual or designed data structure.

Data Usage (BSD)

The frequency of use of entry points and linkages may influence the data structure and storage structure design. Total use of each selection condition and relationship (and entry points and linkages based on them), by all of the processes (and the system procedures) to be supported by the data structure, may be derived from detailed process logic analysis and from volumes of entity types and record types based on them.

5.4.5.7 DD5.7 - Record Types List

5.4.5.8 DD5.8 - Perceived Data Structure Diagram

A data structure diagram representing the perceived data structure made with the data definition syntax of the target database management system.

5.4.5.9 DD5.9 - Data Structure Diagram

The Entity Relationship Model transformed into a structure that is appropriate for a certain type of data base management system e.g. relational.

Stages of a Data Structure Diagram

- Preliminary Data Structure Diagram
- Refined Data Structure Diagram
- Perceived Data Structure Diagram

5.4.6 DD6 - Data Access Diagram

A map showing the path that may be taken through a data structure during the execution of a procedure. For complex procedures, use a data access diagram to understand how a procedure uses the record types. The data access diagram is not a formal deliverable of the business system design phase, but it will help you understand the procedure.

5.4.7 DD7 - Dialog Flows

5.4.7.1 DD7.1 - Dialog Flow Definition

A formal description of all aspects of a dialog flow.

5.4.7.2 DD7.2 - Dialog Flow Diagram

A representation of the steps within a dialog, their sequence, and the screen layouts used by each step.

5.4.8 DD8 - Layout Designs

5.4.8.1 DD8.1 - Screen/Report Layout

The screen and report layouts describe the appearance of the screens and reports associated with the proposed system. The position of individual fields on the screen or report are identified. Fields are represented by fill characters rather than the actual attribute values; the fill character indicates the type of editing that will be performed on the field. Options open to the user for moving to other screens are defined. The layout may be based on a template.

Each field on a layout has a definition that details how it will appear on the layout. These details include position, size, display conditions, and characteristics. The definition is specific to a layout.

- Special Fields
- Literals

5.4.8.2 DD8.2 - Design Template

Layout template used during the design of screens and reports. Templates are created to position common layout items consistently across layouts. A corporate template may already exist to ensure consistency for certain elements across business systems. If a corporate template exists, use it as a basis for the system templates. If it does not exist, consider creating one. Corporate templates generally do not contain fields, although the business system templates may.

Items Included On a Template

- Special fields
- Title literals (e.g., enterprise name, business system name)
- Fields

Example of a Design Template

Consistency of visual layout across the system is important in order to gain user acceptance. In many organisations, it is desirable to have consistent interfaces across the whole organisation; that way, staff can move from one area to another without having to learn new formats and practices. Certain aspects of design should be consistent across layouts.

Templates are defined to different levels. In some business systems, totally different layout styles exist for different user groups - for example, trading and back office users in a bank.

Templates can be layered - a business system template can be "laid down" on an organisation unit or corporate template.

5.4.9 DD9 - Prototype

This is an early, simplified version of a software application that has some of the functions of the eventual application but not the scale or performance. Specific types of prototypes are described below.

Application Concept Prototype

An application concept prototype is a system developed to test and refine the underlying premise of an application.

Application Package Prototype

This is a "canned" application tried out with a small group of users before its full-scale implementation.

Calculations and Logic Prototype

These are examples of the computations for complex logic or calculations in an application.

Data Entry Prototype

This is a shell routine or system to check the speed and accuracy of data entry, and to test validity and integrity checks.

Data System Prototype

This is a small-scale database to test database contents and organisation.

Dialog Prototype

This is a simulation of the interplay of user actions and the intended system response.

Partial-System Prototype

A partial-system prototype emulates a facet of an application.

Pilot

A pilot is a system that is not yet fully operational. Whereas a prototype is used when the functions and detailed design of a system are not fully understood, in a pilot system the functions and design are thought to be understood, but the system is cut over in limited form so that experience can be gained with it before the full system is implemented.

If the final system is to have five hundred terminals, a pilot system might be initially operated with five. If the system is to be installed in one thousand dealerships, it might first be operated in ten. As a result of pilot operation, modifications are usually made before the system is fully deployed.

Reporting System Prototype

This is a mock-up of reports provided to users.

Screen Prototype

A screen prototype is a working model of the screens and their interaction that simulates the operation of the system once it is ultimately developed. The prototype lacks logic to process or display data. It can, however, demonstrate

movements from one screen to another, simulating navigation through a real version of the system.

System Prototype

A system prototype is an executable version of the system procedures. The prototype is based on the source code developed to implement the system, which is itself derived from the design specifications describing the procedure. A prototype of the system will be produced to whatever level of detail is required. At a minimum, this will consist of an agreed-upon sequence of dialogs for the system, in whole or in part.

For each dialog in a prototype, the following will be produced:

- Menu screens
- Data entry screens
- Verification rules
- Procedures for invoking screens

Although the word "prototyping" suggests an analogy with engineering (where prototypes are used extensively), the prototyping of software is fundamentally different from the prototyping of machines.

In engineering, a machine prototype usually takes longer to build and is much more expensive than the ultimate product. Because the ultimate product may come from a mass-production line, the prototype would be needed for testing the product before the production line is built.

In software, there is no production line. Prototyping is practical only if the prototype can be built quickly and cheaply. Unlike production engineering, a software prototype is not a full-scale version of the eventual system; it is usually a simplified version that has some of the functions of the working system but not the scale or performance. It lacks features like security, auditability, recoverability, and the ability to handle large volumes, large databases, or many users.

5.4.10 DD10 - Procedure/Data Matrices

5.4.10.1 DD10.1 - Record/Procedure Matrix

Specific form of Involvement Matrix that shows the procedures in which each record is involved and the actions concerned.

5.4.10.2 DD10.2 - Linkage/Procedure Matrix

Specific form of an Involvement Matrix that shows, the procedures in which each linkage is involved and the actions concerned.

5.4.10.3 DD10.3 - Field/Procedure Matrix

Specific form of an Involvement Matrix that shows the procedures in which each field is involved and the actions concerned.

5.4.11 DD11 - Quality Reports

5.4.11.1 DD11.1 - Completeness Quality Report

This report lists any outstanding weaknesses and problems in the business system design phase. Many of these problems will not prevent the project from continuing while the problems are fixed. Quality assurance standards will indicate what is allowable.

5.4.11.2 DD11.2 - Correctness Quality Report

This contains any unresolved weaknesses and problems in the design that have been identified during the task. Many of these problems will not prevent the project from continuing while the problems are fixed. The Quality assurance manager is responsible for such decisions.

5.4.12 DD13 - Procedure/Benefit Matrix

A matrix that indicates the quantified benefits to be achieved by providing automated support to each of the identified procedures.

5.4.13 DD13 - Cost-Benefit Data

5.4.13.1 DD13.1 - Cost-Benefit Analysis

A parameter-based estimate of the total cost to develop and cutover the proposed area, based on the present definition of the areas scope, against the quantitative estimate of benefits that could be achieved by the area. The analysis can be done for several scopes, depending on the task involved. Typical scopes are per business area, design area and/or procedure.

Costs accrue from the implementation of an activity or entity type. As with benefits, costs are either tangible or intangible. Tangible costs are directly caused

by the implementation of an activity or entity type and can be attached to a direct monetary value.

Other tangible costs include things like training and implementation. Intangible costs are often more important than tangible costs. They include things like disruption to the enterprise and consequential loss of customer service, and non-availability of staff.

Accurate costing of systems development from analysis information is generally not possible. Even so, estimates are essential. Most organisations have rules of thumb accumulated by experience, but development costs can change significantly for many reasons, including:

- The programmer or designer employed
- The software in use
- Database size
- Database complexity
- Transaction volumes
- Transaction complexity
- Experience with the technology to be used

Components of the Cost-Benefit Analysis

- Cost-Benefit Summary Report
- Benefit Analysis Matrix
- Benefit Analysis Table

5.4.13.2 DD13.2 - Cost-Benefit Summary Report

A report that summarises the findings of a cost-benefit analysis.

5.4.14 DD14 - Technical Design Phase Plan

A plan for conducting the technical design phases of the design stage. The plan includes the deliverables to be produced, the tasks to be performed, the resources required to perform the tasks and the time-line for the tasks.

5.4.15 DD15 - Design Infrastructure

5.4.15.1 DD15.1 - Project Standards

The set of development rules and standards that are to be used during the project. These may include standards that determine the answers to the following questions:

- Which techniques will be employed if there are alternatives?

- When will the tasks be carried out?
- Which team members will be involved in the tasks and how much of their time is to be spent on each task?

5.4.15.2 DD15.2 - Acceptance Criteria

A table that lists the criteria that a designed system must meet in order to be acceptable to the organisational units that must approve the system.

5.4.15.3 DD15.3 - Technical Objectives

The set of technology oriented objectives for the system.

5.4.15.4 DD15.4 - Technical Architecture

A structure that summarises the mixture of hardware, software and communications facilities that supports or will support the business systems and other parts of the information environment within the company.

The Technical Architecture includes descriptions of:

- Existing Technology Inventory
- Technical Requirement/Business System Matrix
- Technical Requirement/Data Store Matrix
- Technical Requirement/Location Matrix
- Inter-location Communications Matrix
- Technical Architecture Options Evaluation Matrix
- Technical Architecture Diagram

The **Existing Technology Inventory** is an inventory of products used for each technical component. The inventory shows, for each Technical Component, the products presently used within the enterprise. It gives details about the number of products, location of installation, the status (whether owned or rented and the contract expiration date), year of acquisition, and performance.

The following Technical Components must be considered:

Processing facilities

Processing facilities include hardware, operating systems and processing support software in:

- Common central facilities
- Departmental facilities (e.g., office equipment)
- Process automation facilities
- Single-user facilities (e.g., personal computers, professional workstations, terminals)

Communication facilities

- Network protocols (wide area, local area in factories or offices)
- Network processors
- Routers
- PBX equipment
- Facsimile equipment
- Network cabling

Database management software

- Hierarchical, codicil, network relational systems
- Multi-user, single-user systems
- Data dictionary

Systems development facilities

- Analysts' workbench
- Following-generation languages
- (End-user) query languages
- Application generators

Office support software

- Text processing
- Electronic mail
- Information retrieval

Decision support software

- Statistical, O. R. software
- Spreadsheets
- Expert system shells

Technical support software

- Computer-aided design tools
- Computer-aided manufacturing and testing

Outside resources

- Information retrieval services
- Application services

Communications Network

Another technology inventory deliverable is the layout of the communications network (data, voice, text) as a pictorial generation of network connections, nodes, and facilities.

The **Technical Requirement/Business System Matrix** shows the technical facilities required for each Business.

The **Technical Requirement/Data Store Matrix** shows the technical facilities required for each Data Store. For each data store defined in the BSD, an overview is given of technical facilities required. These include:

-
- Storage facilities (type - e.g., mass storage)
- Database management software, file servers
- Security system software
- Special facilities for data representation depending on data type (e.g., voice, image)

General facilities are also identified (e.g., data dictionary).

The **Technical Requirement/Location Matrix** shows the technical facilities required at each location type within the enterprise. Note: It does not state whether the facilities must actually be present at a location. That is, they may be provided from another location.

The **Interlocation Communications Matrix** shows the characteristics of the communication flow of data, text, voice, and image between various location types. The originating and receiving locations are shown in a matrix. Each entry expresses the type of communication (e.g., interactive, batch) and expected volume (e.g., high, medium, low).

The **Technical Architecture Options Evaluation Matrix** is an evaluation, by relative impact factors, of alternative options within the Technical Architecture.

The **Technical Architecture Diagram** is a diagram that summarises the mixture of hardware, software and communications facilities that supports or will support the business systems and other parts of the information environment within the enterprise. It is a list of technical components that will be considered in the Technical Architecture. It includes the following categories:

- Processing facilities (hardware, operating systems, systems support software)
- Terminals and workstations
- Communication facilities
- Database management systems
- System development facilities
- Office support software
- Decision support software
- Technical support software
- Outside resources

Feasible Technical Products is a version of the Technical Architecture Diagram showing technical products instead of technical facilities.

5.4.15.5 DD15.5 - Hardware Context Diagram

This diagram indicates the hardware elements by location and the network lines available to support the business system. The diagram should indicate which environments will be used for construction, testing and production.

Hardware Elements Included in the Diagram

- Computer environments
- Processors
- Storage devices
- Workstations and personal computers
- Communications facilities
- Input/output devices
- Automation (e.g., robotics)

5.4.15.6 DD15.6 - Facility/Software Element Matrix

A matrix that indicates the software elements that support each technical facility.

5.4.15.7 DD15.7- Capacity Evaluation Assessment

The assessment of the availability of required hardware capacity for a designed system. The assessment contains the Capacity Requirements Table.

Hardware Components to be Considered

- Storage devices
- Workstations and personal computers
- Communications facilities
- Input/output devices
- Automation (e.g., robotics)
- Processors

5.4.15.8 DD15.8 - Language Evaluation Matrix

A summary of the programming language(s) that can be used to support each type of procedure in the designed system.

5.4.15.9 DD15.9 - Reusable Object Library

During the development effort many design objects are produced that could be useful in implementing other applications later. These design objects include screen layouts, data structures and processing logic definitions.

Storage and Access of Reusable Object Library

These design objects are stored in the repository, available for future use as needed. They may also be recorded in other formats for later reference. Practical access to the Reusable Object Library requires some form of index to allow rapid identification and classification of the contents of the library. The form of this index will vary depending on the CASE toolset in use.

5.4.15.10 DD15.10 - Development Standard

Any form of standard and procedure that influences the execution or content of development tasks. Development standards are established throughout the organisation but may be developed externally (e.g. RUP, Iconix, UML, etc.). Additional requirements, not just the project standards, can be added. Included are definition and naming conventions, requirements for display, storage, transmission and documentation and environment standards.

Specific standards and procedures include the following:

Layout Defaults

Layout defaults are default display properties and standard edit patterns.

Methodology Standards

These standards describe how tasks are performed and how the deliverables are documented. They will include the Information Engineering Methodology rules and conventions and may include additional standards special to the enterprise or the tools used.

Project Management Standards

These standards include progress review procedures and administrative procedures.

5.4.15.11 DD15.11 - Facility Acquisition Plan

The Facility Acquisition Plan briefly identifies each acquisition situation associated with the proposed new application. The plan identifies the steps necessary to acquire and install the facility, provides a schedule for performing these steps, and identifies responsibilities for ensuring the steps are completed.

5.4.15.12 DD15.12 - Facility Requirements Assessment

The Facility Requirements Assessment summarises the expected requirements of the proposed system for specific hardware and system software. Facility categories include central processors, data storage, communications,

workstations, and systems software. The assumptions on which the projections of facility requirements are based are described in this document.

5.4.16 DD16 - Load Matrices

A matrix that summarises the usage of record types, linkages and entry points of the preliminary data structure by the procedures included within the design area. For an equivalent representation an Annotated Preliminary Data Structure Diagram may be used.

5.4.16.1 DD16.1 - Record Type Load Matrix

To determine the usage load on record types for each procedure, create a matrix like the example below which reflects the volume of reads and writes to each record type from each procedure for a specified time period. The time period can be a day, week, month, peak period or any other meaningful time period.

5.4.16.2 DD16.2 - Search Field Load Matrix

To determine the usage load on search field for each procedure, create a matrix like the example below which reflects the volume of accesses to each search field from each procedure for a specified time period. The time period can be a day, week, month, peak period or any other meaningful time period.

5.4.16.3 DD16.3 - Linkage Load Matrix

To determine the usage load on each search field for each procedure, create a matrix that reflects the volume of uses of each search field to access a record occurrence from each procedure during a specified time period. The selected time period can be a day, week, month, peak period or any other meaningful time period.

5.4.17 DD17 - Software Structure

5.4.17.1 DD17.1 - Program Specification

A comprehensive definition of a program, its behaviour and its interactions.

Contents of the Specification

It may contain:

- Program structure diagram

- Details of data to be accessed, including program specification blocks, if appropriate
- Details of interfaces between calling and called modules
- Provision to add test conditions
- Record action list

5.4.17.2 DD17.2 - Module Specification

A comprehensive definition of a module, its behaviour and its interactions.

Contents of the Module Specification

- Module Structure Diagram
- Module Action Diagram

5.4.17.3 DD17.3 - Software Structure Derivation Matrix

A matrix that shows the relationship between programs and modules and the procedures and elementary steps from which they are derived.

5.4.17.4 DD17.4 - Software Hierarchy Diagram

A diagram that shows which programs and modules each integration group consists of and to which integration group the common logic modules belong.

5.4.17.5 DD17.5 - Software Context Diagram

This diagram shows the systems and generic application software elements in each processor. The diagram is accompanied by a short description of the facilities offered by each software element. These facilities are summarised on the facility/software element matrix.

Elements Included on the Software Context Diagram

Procedure execution management

This is system software to control the execution of application software (e.g., operating system and transaction processing monitor).

Data management

This is software to define and manage access to data on a computer system (e.g., database management system, information retrieval system).

Network management

This is software to define, manage, diagnose, and correct the use of communications facilities (e.g., Communications Architecture).

Development tools and languages

This is software to generate, code, manage, or protect application programs (e.g. language, application generator, code library manager).

- End-user facilities
- Information centre tools

5.4.17.6 DD17.6 - Structure Chart

A diagram that shows the functional decomposition of (a part of) a business system by showing the component modules and their interrelationships.

5.4.18 DD18 - Data Storage

5.4.18.1 DD18.1 - Data Storage Structure

The way in which data are organised in storage. It determines the performance of the business systems in their use of data, but is not known to programs.

The data storage structure defines how the data specified in the data structure design are to be physically organised on disk (or on the secondary storage medium selected). It involves deciding on the data sets, data-set characteristics, and placement of records and control fields (e.g., pointers) within these data sets. It includes determining the storage mapping.

Ideally, you can alter the data storage structure without affecting the data structure design. This is a characteristic of most database management systems.

5.4.18.2 DD18.2 - Data Storage Space Specification

5.4.18.3 DD18.3 - Buffer Design

5.4.18.4 DD18.4 - Dataset Summary

A table listing the datasets to be used for a designed system and specifying their characteristics.

5.4.19 DD19 - Integration Group

5.4.19.1 DD19.1 - Integration Group Definition

A group of programs which use the same data stores, formed together in order to be scheduled, developed and tested as a whole.

5.4.19.2 DD19.2 - Integration Group Dependency Diagram

A diagram that shows the dependency of integration groups on one another for testing.

5.4.20 DD20 - Test Plans

5.4.20.1 DD20.1 - System Test Plan

This is a document describing the scope, approach, resources, and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning.

A series of actions in sequence intended to prove that a new business system as a whole operates as anticipated in the design specification. The system test plan describes the test scenarios, test conditions and test cycles that must be performed to ensure that system testing follows a precise schedule and that the system is thoroughly tested before moving into production.

The technical design stage defines the scope of system tests to verify that the new business system meets the design specifications. The more definitive the system test plan, the easier it is to execute system tests. If the system is modified between the development of the test plan and the time the tests are executed, the test plan must be updated. Whenever the new business system changes, the system test plan should be updated to reflect the changes.

The project team has ultimate responsibility and accountability for execution of the tests. It is both convenient and cost effective to use clerical or administrative personnel to execute the tests. Regardless of who actually runs the tests, the project team leader must accept responsibility for ensuring that the tests are executed according to the documented test plan.

Components of the Fully Completed System Test Plan

- System Test Requirements
- System Test Cycle Dependency Diagram

The following is an indicative outline test plan:

Test Plan Identifier

- A unique identifier

Introduction

- Summary of the items and features to be tested
- Need for and history of each item (optional)
- References to related documents such as project authorisation, project plan, QA plan, configuration management plan, relevant policies, relevant standards
- References to lower level test plans

Test Items

- Test items and their version
- Characteristics of their transmittal media
- References to related documents such as requirements specification, design specification, users guide, operations guide, installation guide
- References to bug reports related to test items
- Items which are specifically not going to be tested (optional)

Features to be Tested

- All software features and combinations of features to be tested
- References to test-design specifications associated with each feature and combination of features

Features Not to Be Tested

- All features and significant combinations of features which will not be tested
- The reasons these features won't be tested

Approach

- Overall approach to testing
- For each major group of features of combinations of featres, specify the approach
- Specify major activities, techniques, and tools which are to be used to test the groups
- Specify a minimum degree of comprehensiveness required
- Identify which techniques will be used to judge comprehensiveness
- Specify any additional completion criteria
- Specify techniques which are to be used to trace requirements
- Identify significant constraints on testing, such as test-item availability, testing-resource availability, and deadline

Item Pass/Fail Criteria

- Specify the criteria to be used to determine whether each test item has passed or failed testing
- Suspension Criteria and Resumption Requirements
- Specify criteria to be used to suspend the testing activity
- Specify testing activities which must be redone when testing is resumed

Test Deliverables

- Identify the deliverable documents: test plan, test design specifications, test case specifications, test procedure specifications, test item transmittal reports, test logs, test incident reports, test summary reports
- Identify test input and output data
- Identify test tools (optional)

Testing Tasks

- Identify tasks necessary to prepare for and perform testing
- Identify all task interdependencies
- Identify any special skills required

Environmental Needs

- Specify necessary and desired properties of the test environment: physical characteristics of the facilities including hardware, communications and system software, the mode of usage (i.e., stand-alone), and any other software or supplies needed
- Specify the level of security required
- Identify special test tools needed
- Identify any other testing needs
- Identify the source for all needs which are not currently available

Responsibilities

- Identify groups responsible for managing, designing, preparing, executing, witnessing, checking and resolving
- Identify groups responsible for providing the test items identified in the Test Items section
- Identify groups responsible for providing the environmental needs identified in the Environmental Needs section

Staffing and Training Needs

- Specify staffing needs by skill level
- Identify training options for providing necessary skills

Schedule

- Specify test milestones
- Specify all item transmittal events
- Estimate time required to do each testing task
- Schedule all testing tasks and test milestones

- For each testing resource, specify its periods of use

Risks and Contingencies

- Identify the high-risk assumptions of the test plan
- Specify contingency plans for each

Approvals

- Specify the names and titles of all persons who must approve the plan
- Provide space for signatures and dates

5.4.20.2 DD20.2 - System Test Cycle Dependency Diagram

A diagram that shows why, for each test cycle, an execution may depend upon the prior execution of other test cycles.

5.4.20.3 DD20.3 - Acceptance Test Requirement

Test cycles assigned to systems each of which consists of conditions to be tested by its user.

5.4.20.4 DD20.4 - Acceptance Test Cycle Dependency Diagram

Specific form of Test Cycle Dependency Diagram used during acceptance testing.

5.4.20.5 DD20.5 - Test Cycle Dependency Diagram

Diagram showing the sequence in which the system and acceptance test cycles (test sessions) can be performed. The diagrams also show the estimated effort required to complete each test cycle, and the minimum elapsed time for each test cycle.

The conventions for drawing a system or acceptance test cycle dependency diagram are like those for drawing an integration test cycle dependency diagram.

Specific Variants of the Test Cycle Dependency Diagram

- Integration Test Cycle Dependency Diagram
- System Test Cycle Dependency Diagram
- Acceptance Test Cycle Dependency Diagram

5.4.20.6 DD20.6 - Benchmark Test Plan

The plan for conducting a test of the system to determine whether it satisfies the defined performance criteria.

5.4.20.7 DD20.7 - User Acceptance Test Plan

The **user acceptance** test plan describes the test scenarios, test conditions and test cycles that must be performed to ensure that acceptance testing follows a precise schedule and that the system is thoroughly tested before moving into production.

The technical design stage defines the scope of acceptance tests to verify that the new business system meets the design specifications. The more definitive the acceptance test plan, the easier it is to execute the acceptance tests. If the system is modified between the development of the test plan and the time the tests are executed, the test plan must be updated. Whenever the new business system changes, the acceptance test plan should be updated to reflect the changes.

The end users have ultimate responsibility and accountability for execution of the tests. It is both convenient and cost effective to use project team, clerical and/or administrative personnel to execute the tests. Regardless of who actually runs the tests, the end users must accept responsibility for ensuring that the tests are executed according to the documented test plan.

5.4.20.8 DD20.8 - Testing Strategy

The Testing Strategy is developed in the User Design stage. It describes the general approach that will be adopted in testing the system during development. The document identifies the key considerations on which the strategy is based, and defines the types of testing that will be employed. If appropriate, this document also outlines a general testing scenario that will be followed by all tests during development. The strategy will in a later stage be the input for the Test Plan.

5.4.20.9 DD20.9 - Test Plan

The Test Plan describes the test scenarios, test conditions and test cycles that must be performed to ensure that integration testing, system testing and acceptance testing follow a precise schedule and that the system is thoroughly tested before moving into production.

The Three Components of the Test Plan

- Integration Test Plan
- System Test Plan

- User Acceptance Plan

The technical design phase of the design stage defines the scope of system tests to verify that the new business system meets the design specifications. The more definitive the system test plan, the easier it is to execute system tests. If the system is modified between the development of the test plan and the time the tests are executed, the test plan must be updated. Whenever the new business system changes, the system test plan should be updated to reflect the changes.

The project team has ultimate responsibility and accountability for execution of the tests. It is both convenient and cost effective to use clerical or administrative personnel to execute the tests. Regardless of who actually runs the tests, the project team leader must accept responsibility for ensuring that the tests are executed according to the documented test plan.

5.4.21 DD21 - Security

5.4.21.1 DD21.1 - Security and Control Procedure Definition

Definition of a procedure describing security and control aspects. An intermediate deliverable is the Access Restrictions.

5.4.21.2 DD21.2 - Security Module Specification

A Module Specification of a module concerned with ensuring the secure use of a system.

5.4.21.3 DD21.3 - Construction Approach

Documentation and definition of the suggested construction approach, either through parallel development or `versions' of the system. The work plan for construction shows the timelines and effort required.

5.4.21.4 DD21.4 - System Construction Assignment List

The System Construction Assignment List identifies the individuals assigned to participate in the construction of the proposed system. The specific responsibilities assigned to each individual are identified on this list, in terms of the work steps that they will participate in.

5.4.21.5 DD21.5 - User Documentation Team Assignment

The User Documentation Team Assignment List identifies the individuals assigned to develop the manual procedures documentation, training materials and operational instructions.

5.4.22 DD22 - System Performance

5.4.22.1 DD22.1 - Performance Monitoring Measure

A list of units of measures valid for performance monitoring.

5.4.22.2 DD22.2 - Performance Monitoring Procedure Definition

A description of how system performance is to be evaluated and monitored.

5.4.22.3 DD22.3 - Performance Assessment Summary

A table that details any performance-related issues and the actions to be taken on each procedure or step in a designed system.

An intermediate deliverable is the Procedures for Assessment List.

5.4.23 DD23 - Construction Plan

A plan describing the deliverables, tasks, resources and time-line of the construction stage for an implementable system.

6. Construction Phase

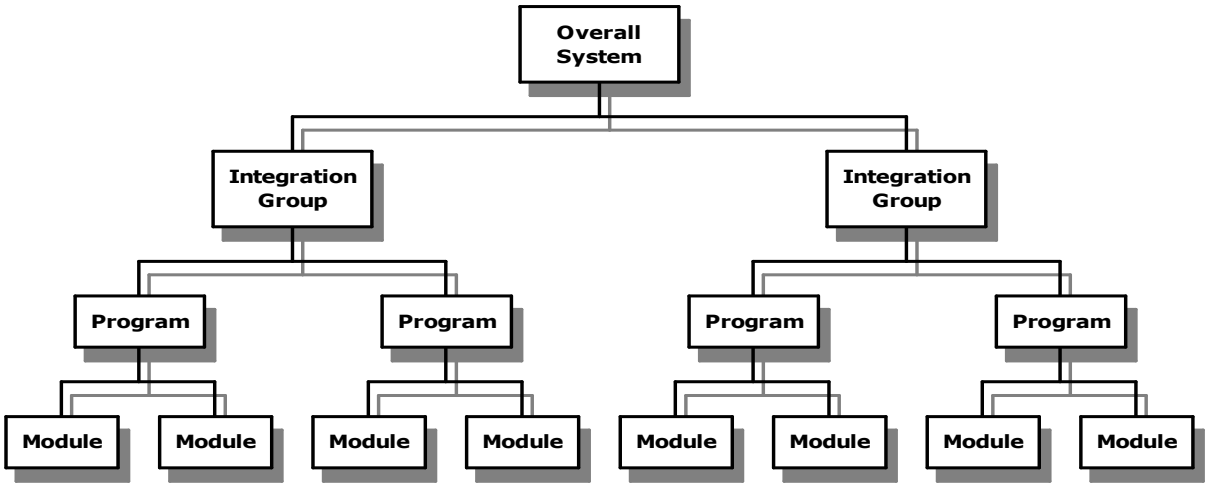
6.1 Introduction

The information system and the data structures which it will utilise are created. The deliverable of this stage is referred to as an implementable system. In addition, the implementation plans, operational documentation and operational training for the implementable system are created in this stage. The unit of work represented by this stage is referred to as a construction project.

The scope of the phase is the developed system.

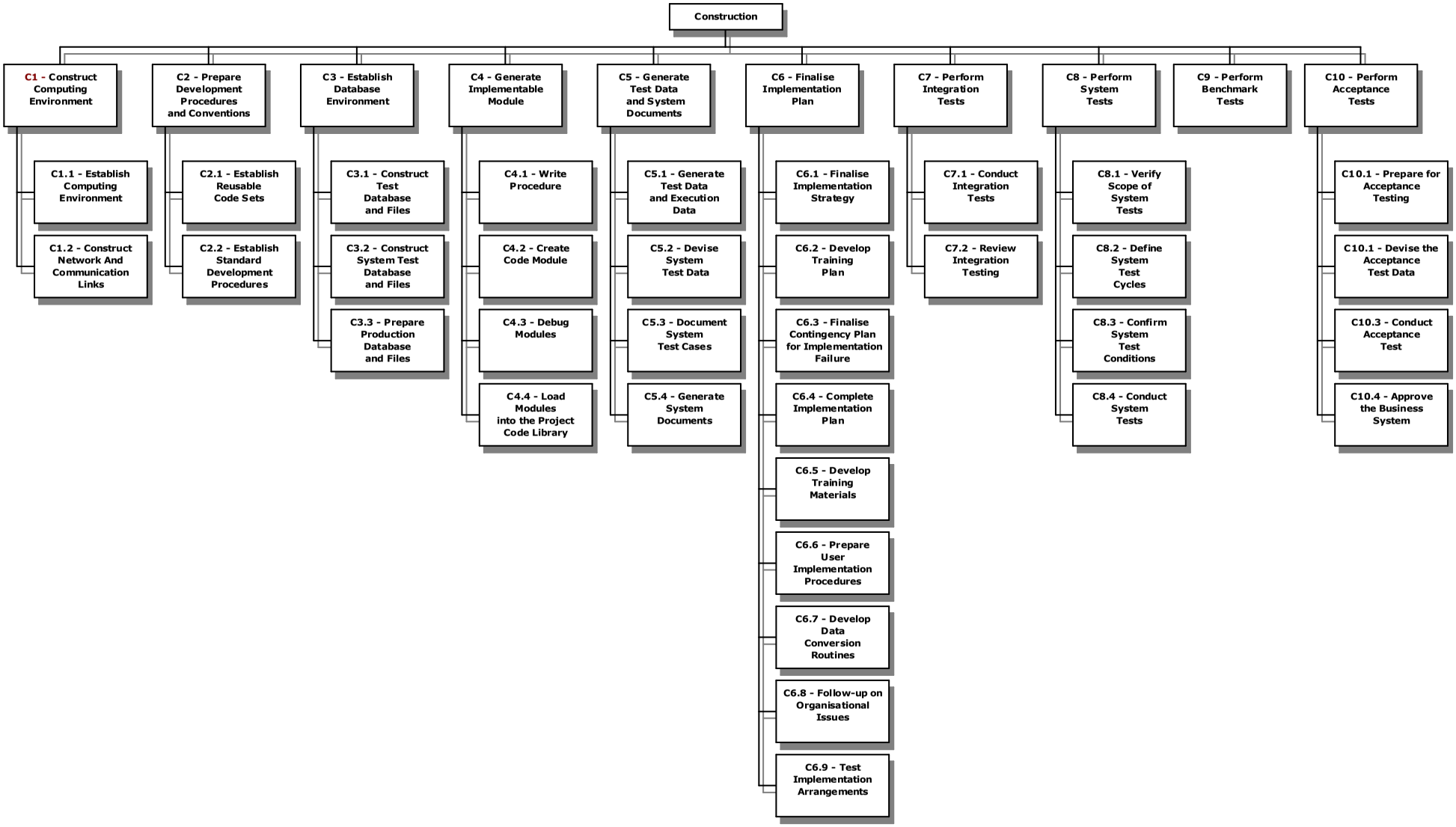
The phase input is the design generated from the design phase.

Schematically, the relationship between the overall system being developed, integration groups, programs and modules is:



The construction phase will be iterated for each unique set of developed code such as module, program, integration group and overall system.

Schematically, the structure of the steps in the Construction phase is:


Construction
C1 - Construct Computing Environment
C1.1 - Establish Computing Environment
C1.2 - Construct Network And Communication Links
C2 - Prepare Development Procedures and Conventions
C2.1 - Establish Reusable Code Sets
C2.2 - Establish Standard Development Procedures
C3 - Establish Database Environment
C3.1 - Construct Test Database and Files
C3.2 - Construct System Test Database and Files
C3.3 - Prepare Production Database and Files
C4 - Generate Implementable Module
C4.1 - Write Procedure
C4.2 - Create Code Module
C4.3 - Debug Modules
C4.4 - Load Modules into the Project Code Library
C5 - Generate Test Data and System Documents
C5.1 - Generate Test Data and Execution Data
C5.2 - Devise System Test Data
C5.3 - Document System Test Cases
C5.4 - Generate System Documents
C6 - Finalise Implementation Plan
C6.1 - Finalise Implementation Strategy
C6.2 - Develop Training Plan
C6.3 - Finalise Contingency Plan for Implementation Failure
C6.4 - Complete Implementation Plan
C6.5 - Develop Training Materials
C6.6 - Prepare User Implementation Procedures
C6.7 - Develop Data Conversion Routines
C6.8 - Follow-up on Organisational Issues
C6.9 - Test Implementation Arrangements
C7 - Perform Integration Tests
C7.1 - Conduct Integration Tests
C7.2 - Review Integration Testing
C8 - Perform System Tests
C8.1 - Verify Scope of System Tests
C8.2 - Define System Test Cycles
C8.3 - Confirm System Test Conditions
C8.4 - Conduct System Tests
C9 - Perform Benchmark Tests
C10 - Perform Acceptance Tests
C10.1 - Prepare for Acceptance Testing
C10.1 - Devise the Acceptance Test Data
C10.3 - Conduct Acceptance Test
C10.4 - Approve the Business System

6.2 Steps in Construction Phase

6.2.1 C1 - Construct Computing Environment

6.2.1.1 C1.1 - Establish Computing Environment

This task should proceed in parallel with analysis and design projects. Such projects are established during the EM stage. The nature and timing of this task can fluctuate greatly, depending on three circumstances.

Existence of the Computing Environment

The computing environment to be used may already exist. The task is then limited to reorganising system code libraries, selecting operating system options, selecting network options, etc. This must be done in order to support any additional workload.

Establishment of the Computing Environment

A local computing environment may have to be established for the first time. The hardware, software and links will be installed and tested at one local site in parallel with technical design stage. Other sites can be constructed for implementation and rollout.

This step creates the following deliverables:

- **Computing Environment** - see section 6.3.1.1 CD1.1 - Computing Environment on page 257

Creation of a Major New Computer Site

A major new computer site, consisting of computer platforms and peripherals, may have to be established. The work would consist of:

- Constructing or extending the location for the facilities, if required
- Installing the hardware
- Installing the software
- Testing the installation
- Establishing the operations staff

This step updates the following deliverables:

- **Computing Environment** - see section 6.3.1.1 CD1.1 - Computing Environment on page 257

6.2.1.2 C1.1 - Construct Network And Communication Links

Communication links between computing environments may be required for development or production. Ideally, the need for network links would have been recognised during the EM stage, and a special project would have been established to implement or extend the communications network. If the links have already been established by previous projects, no linkage construction work will be required.

When new links need to be established, plan as early as possible, usually before the beginning of the BSD stage. This is necessary because it takes a long time to acquire the hardware and circuits and to test the links. This work should proceed in parallel with BSD to prevent system construction and testing delays caused by the absence of operational links.

This step creates the following deliverables:

- **Network And Communication** Links - see section 6.3.1.2 CD1.2 - Network and Communication Links on page 257

6.2.2 C2 - Prepare Development Procedures and Conventions

If a development project uses existing development environments, the task is limited to the establishment of project code libraries and new procedures unique to the project. The code libraries need to be separated into testing, production and possibly other (depending on project complexity) libraries.

6.2.2.1 C2.1 - Establish Reusable Code Sets

When new hardware and new application development tools will be used, the installation and testing should proceed in parallel with the BSD phase. In many cases, in-house subroutines, macros, and skeleton programs may be available. Their use may improve development productivity.

The adoption of in-house software, as well as the incorporation of new application software, must be done long before module generation and testing.

6.2.2.2 C2.2 - Establish Standard Development Procedures

Adapt procedures for data naming, module naming, coding, code migration and testing to comply with the Application Development Methodology (ADM) requirements. In many cases, you can adopt in-house procedures for test database dumping and restoring, file comparison, copy code library maintenance, version control and module generation. This can improve productivity.

Procedures must be established long before module generation and testing. Incorporate the development environment into the construction standards.

This step creates the following deliverables:

- **Development Standard** - see section 6.3.2.1 CD2.1 - Development Standard on page 257
- **Project Code Library** - see section 6.3.2.2 CD2.2 - Project Code Library on page 257
- **Code Migration Procedure** - see section 6.3.2.3 CD2.3 - Code Migration Procedure on page 257

6.2.3 C3 - Establish Database Environment

6.2.3.1 C3.1 - Construct Test Database and Files

Create a test database for unit and initial integration testing which includes the following components.

Database Definition Statements

Generate database definition statements from the information library, or code them by hand if no automated generation tool is available. Check the results manually.

Record Layout Definitions

Record layout definitions are generated from the information library, or they are prepared manually. Check the layouts manually and by compilation, possibly into a dummy program.

Data Required to Create Initial Test Database

Generate the data required to create an initial test database. Populate the initial test database with a minimum number of data records. Use the application programs to populate the database. Design the construction sequence so that test data can always be regenerated through previously constructed modules.

Non-Database Files

The new business system may use non-database files. For integration testing, construct these files as part of module test data generation.

6.2.3.2 C3.2 - Construct System Test Database and Files

Create a test database that will be used for system, benchmark and acceptance testing. Conduct system testing using separate code libraries from unit and

integration testing. Conditions for system testing should be the same as those for the production system.

Database Definitions and Record Layouts

Place database definitions and record layout definitions in the system test code libraries.

This step updates the following deliverables:

- **Project Code Library** - see section 6.3.2.2 CD2.2 - Project Code Library on page 257

System Test Database

Create the system test database using the procedure that will be used to create the initial production database. If the database management system and application allow, begin with an empty preformatted database.

New Database as an Extension of an Existing Database

If the new database is an extension of an existing database, unload and reload the existing database. Unload the test database, which is structured like the current production database, and reload it with the new database structure.

Population of the System Test Database

Populate the system test database using the programs that will be used to populate the production databases, such as conversion programs and on-line and batch update programs. You may also need to define non-database files, such as flat files, in code libraries before they can be used during system testing.

This step creates the following deliverables:

- **Test Cases and Test Data** - see section 6.3.4.1 CD4.1 - Test Cases and Test Data on page 258

6.2.3.3 C3.3 - Prepare Production Database and Files

The resources required by each data set of the production database and files were determined in technical design. The following tasks are also performed for non-database files, but it is usually not necessary to generate data definitions.

Database Definition Statements

Create the database definition statements and process them to create the database definitions. The statements will be identical to those used in system testing, except for parameters that vary according to database size, or for different naming standards for the production and system test objects.

Execution Data

Prepare the execution data used to create the initial production database by adjusting the data used for system testing.

This step creates the following deliverables:

- **Executable Program Module** - see section 6.3.3.1 CD3.1 - Executable Program Module on page 257
- **Production Data Structure** - see section 6.3.3.2 CD3.2 - Production Data Structure on page 257
- **System Documentation** - see section 6.3.3.3 CD3.3 - System Documentation on page 257

6.2.4 C4 - Generate Implementable Module

An implementable module may consist of either or both automated or manual components.

6.2.4.1 C4.1 - Write Procedure

If the module is to consist of manual procedures, write the procedural steps necessary to achieve the objective of the module within the known constraints.

6.2.4.2 C4.2 - Create Code Module

Specification details required to code the modules are stored in the information library. The use of various programming languages will dictate whether the application code is hand prepared or automatically generated. Once the code is written, prepare the initial module documentation.

This step creates the following deliverables:

- **Implementable Module** - see section 6.3.3.6 CD3.6 - Implementable Module on page 257

6.2.4.3 C4.3 - Debug Modules

The code statements are compiled into executable modules. The executable modules are then tested as units so that errors can be corrected. This testing and debugging can be done by individual programmers on the project team or by a separate group of programmers assigned to perform this subtask. The party performing the testing must understand the business system objectives in order to ensure that the system is built correctly. If necessary, correct the module documentation.

This step updates the following deliverables:

- **Implementable Module** - see section 6.3.3.6 CD3.6 - Implementable Module on page 257
- **System Documentation** - see section 6.3.3.3 CD3.3 - System Documentation on page 257

This step creates the following deliverables:

- **Test Results** - see section 6.3.4.2 CD4.2 - Test Results on page 258

6.2.4.4 C4.4 - Load Modules into the Project Code Library

Place the debugged executable module into the development project code library. The generated modules are now ready for integration testing. Update the operational control procedures to reflect the appropriate set of test data.

This step updates the following deliverables:

- **Project Code Library** - see section 6.3.2.2 CD2.2 - Project Code Library on page 257
- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

6.2.5 C5 - Generate Test Data and System Documents

Generate test data for all components (executable modules and procedures) of an implementable module.

6.2.5.1 C5.1 - Generate Test Data and Execution Data

Complete the following steps using the database constructed in the Construct Database task, earlier.

Prepare data

To test a module, prepare both execution-control and test data. Execution-control data requirements (such as job control software) are specified in TD. The execution data are now prepared and stored in appropriate code libraries.

Prepare test case

The test cycles for integration testing a module are also specified in TD. For each test condition, prepare a test case consisting of input data and anticipated results. A comparison of anticipated and actual test results will show if the result is correct or erroneous.

Errors will normally require individual test cases. More than one test case may sometimes be required to prove that the test condition operates correctly. Whenever possible, verify several correct test conditions in a single test case.

Document test case

For each test case, document:

- Test conditions that the test case is verifying
- Field values for input screens, documents and records
- Initial database state
- Field values for output screens, reports and records
- Resulting database state

Define database state

The formats of the test data will vary, for example, with modification to the test database, dialog inputs, or inputs to a dialog simulator. Rather than define the initial database state and the resulting database state for each test case, it may be easier to do it once for the entire test cycle.

Define test data

If a module is not involved in integration testing, it is not necessary to define test data for that module. The module will be tested when its calling module is tested.

6.2.5.2 C5.1 - Devise System Test Data

The system test requirements define a series of conditions to be tested during each test cycle. Prepare a test case for each test condition, consisting of input data and anticipated results. A comparison of anticipated and actual test results will show whether or not the test conditions have been correctly executed. Whenever possible, automate the creation of input data. Increase productivity by using the anticipated results (or the resulting state of the database) from one test cycle as input for another cycle.

By keeping groups of test cycles independent, it is possible to continue testing one area while other areas await correction. Too much data interdependence will increase the overall elapsed system test time. You may perform this step in parallel with the Generate implementable module task.

This step updates the following deliverables:

- **Test Cases and Test Data** - see section 6.3.4.1 CD4.1 - Test Cases and Test Data on page 258

6.2.5.3 C5.2 - Document System Test Cases

For each system test case, document:

- Test condition that the test case is verifying
- Field values for input screens, documents and records
- Initial database state
- Expected field values for output screens, reports and records
- Resulting database state

This step updates the following deliverables:

- **Test Cases and Test Data** - see section 6.3.4.1 CD4.1 - Test Cases and Test Data on page 258

6.2.5.4 C5.3 - Generate System Documents

Produce manuals, quick reference cards, course slides and other documents. The documents will vary according to the natures of the organisation and the new business system.

Typical Requirements

The following documents are usually required:

- User's guide
- User's quick reference card
- Instructor's training manual and slides
- Operations manual
- User's training course materials
- Database administrator's manual

Help Screens

Help screens will reduce the amount of detail needed in users' guides and operations manuals. The use of on-line manuals and interactive training is increasing. Some of the material for standard documents my be generated from the system information library.

Document Production

Produce documents using the following steps:

10. Define the documents required.
11. Define and review the objectives and scope of each document.
12. Define a list of section headings.
13. Define and review content.

This step creates the following deliverables:

- **Operations Manual** - see section 6.3.3.4 CD3.4 - Operations Manual on page 257

This step updates the following deliverables:

- **Test Cases and Test Data** - see section 6.3.4.1 CD4.1 - Test Cases and Test Data on page 258
- **Test Results** - see section 6.3.4.2 CD4.2 - Test Results on page 258
- **System Documentation** - see section 6.3.3.3 CD3.3 - System Documentation on page 257
- **Information Library** - see section 6.3.3.5 CD3.5 - Information Library on page 257

6.2.6 C6 - Finalise Implementation Plan

6.2.6.1 C6.1 - Finalise Implementation Strategy

A general strategy for implementing the new application software in place of existing applications and procedures was developed previously through discussions with users and review of conversion requirements. This strategy should be reviewed once more with users before proceeding further with implementation planning. Any new considerations based on further design progress should be discussed. The strategy for implementation should be finalised upon completion of this subtask.

This step creates the following deliverables:

- **Implementation Plan** - see section 5.4.2.4 DD2.4 - Implementation Plan on page 211

6.2.6.2 C6.2 - Develop Training Plan

A plan for conducting the user training necessary to support operation of the application in production is developed.

The training plan identifies the types of training that must be provided, and the criteria for selecting the personnel to receive it. A training curriculum is prepared consisting of a outline and the topics to be covered in the training session(s). The materials necessary to support the training are also described, as are the qualifications of the instructor who will present each topic.

This step creates the following deliverables:

- **Training Plan** - see section 6.3.5.2 CD5.2 - Training Plan on page 258

6.2.6.3 C6.3 - Finalise Contingency Plan for Implementation Failure

A plan for recovering from the failure of the new application after it is placed into production operation should be developed for all critical applications. This recovery usually involves restoring the systems and procedures previously in place back to their original status prior to the initiation of the implementation effort.

The need for a contingency plan, and the required level of detail to be included in it, depends greatly on the circumstances surrounding the application being converted. If the application is simple, and its operation is not critical to the functioning of the organisation, then a contingency plan is probably unnecessary. If the application is complex and has a significant probability of failure, or if the application is essential to the day-to-day functioning of the organisation, then a contingency plan is mandatory.

6.2.6.4 C6.4 - Complete Implementation Plan

A detailed plan for performing the implementation is completed. This plan lists the specific steps necessary to place the application in production status. The plan reflects the general implementation strategy defined previously.

The implementation plan is based on the preliminary version developed in BSD. Estimates of the resources required to complete each step are listed on the work plan. Planned start and completion dates for each work step also are listed. A milestone chart identifying key events in the overall process is presented.

This step updates the following deliverables:

- **Implementation Plan** - see section 5.4.2.4 DD2.4 - Implementation Plan on page 211

6.2.6.5 C6.5 - Develop Training Materials

Prepare materials necessary to support training of the end-users of the application during the implementation stage. This activity generally involves development of presentation slides, instruction manuals, wall charts, etc. that will be used in the training sessions. It also involves obtaining copies of relevant documents, particularly sections of the user documentation, for distribution to the users during the training sessions.

This step creates the following deliverables:

- **Training Material** - see section 6.3.5.1 CD5.1 - Training Material on page 258

6.2.6.6 C6.6 - Prepare User Implementation Procedures

Detailed procedures are required to support the execution of the implementation work plan. These procedures will define the detailed steps necessary to perform the various tasks that user personnel will be asked to execute. Many of these procedures involve data preparation tasks that must be performed manually. Other procedures cover operational tasks, such as reconfiguration or relocation of equipment, establishment of communications arrangements, and backup of critical data. Procedures necessary to implement the contingency plan for restoration of the old systems and procedures after the new application has failed also must be developed.

Implementation procedures are usually developed in printed form. These procedures must be explained to the users who will be expected to invoke them.

This step creates the following deliverables:

- **Implementation Procedure** - see section 6.3.7.1 CD7.1 - Implementation Procedure on page 258

6.2.6.7 C6.7 - Develop Data Conversion Routines

Implementation of many applications involves conversion of existing data into a format suitable for loading into the application's data structures. This automated processing will often involve development of routines designed for this specific purpose. These data conversion routines will be coded and tested in this subtask.

This step creates the following deliverables:

- **Data Conversion Routine** - see section 6.3.7.2 CD7.2 - Data Conversion Routine on page 258

6.2.6.8 C6.8 - Follow-up on Organisational Issues

During BSD phase, various organisational issues were identified, and steps initiated to resolve them. During the construction stage any new organisational issues resulting from design changes or implementation arrangements should be identified and resolved. Action on any open organisational issues that must be resolved prior to implementation should be resolved.

6.2.6.9 C6.9 - Test Implementation Arrangements

A test of the implementation plan should be conducted prior to initiating the implementation stage. In this test all steps necessary to perform the implementation should be executed in the same sequence as will occur in the actual implementation event.

The purpose of this test is to uncover problems before the actual implementation process is initiated. Users who will actually be performing the implementation

tasks should participate in the test; this will help to train them in the process, and provide an opportunity to clear up any misunderstandings that they might have. All computer routines should be executed to verify their correct operation. Attention should be paid to the time required to perform each implementation task, and the planned schedule for the implementation verified.

This step updates the following deliverables:

- **Test Cases and Test Data** - see section 6.3.4.1 CD4.1 - Test Cases and Test Data on page 258
- **Test Results** - see section 6.3.4.2 CD4.2 - Test Results on page 258

6.2.7 C7 - Perform Integration Tests

Test the integration of all components (executable modules and procedures) of an implementable module. This includes testing the interface of new modules to existing modules.

6.2.7.1 C7.1 - Conduct Integration Tests

Perform each test cycle of the integration test as soon as the executable module and the called modules are programmed and available. Compare the anticipated results with the actual results.

All discrepancies require modification to the modules' source statements. Generate a new executable module after making modifications to the source. Alternatively, the test condition, test cases or anticipated results may be in error and may have to be corrected. For system auditing, produce hard copy of every test cycle using the final version of the module.

This step updates the following deliverables:

- **Test Results** - see section 6.3.4.2 CD4.2 - Test Results on page 258

6.2.7.2 C7.2 - Review Integration Testing

Inspect the results of integration testing to ensure that the testing is complete and has been properly conducted.

Integration tests will not be repeated during system testing. Before system testing starts, verify that all the modules and system test execution data have been placed in the system test code libraries. If there are time constraints, you may start system testing before all the modules have been constructed, but this can cause scheduling and communication problems.

6.2.8 C8 - Perform System Tests

6.2.8.1 C8.1 - Verify Scope of System Tests

System testing must cover the following areas:

- Testing the computing environment of the business system, which should be as close as possible to the production environment
- Integration group retesting
- Merging the integration groups, breakdown testing and recovery procedures
- Volume testing
- Implementation testing
- Full-procedure testing
- System interface testing

6.2.8.2 C8.2 - Define System Test Cycles

Split all the types of test requirements noted in the scope into manageable test sessions or cycles. Prepare a dependency diagram to show the sequence in which these cycles are to be performed.

6.2.8.3 C8.2 - Confirm System Test Conditions

When the review of changes occurring between BSD and integration testing is complete, estimate the elapsed time and the effort required to conduct each test cycle and add the estimates to the dependency diagram prepared in the subtask Review Test Cycles.

Because the test team members responsible for system testing may not have performed the integration tests, they should satisfy themselves that those tests have been completed.

6.2.8.4 C8.4 - Conduct System Tests

As each test is executed, compare the output with the anticipated results. If any errors are found, have the programmers correct them. The amount of retesting required depends on the nature of the problems. Except in the most severe cases, do not perform another complete set of integration tests on the faulty integration group or repeat all the system tests for every correction.

Continue system testing on one part of the new business system while defects are being corrected in another part. Testing is complete only when all test cycles have been performed successfully and when the information management is confident enough to hand the new business system to the users.

This step updates the following deliverables:

- **Test Cases and Test Data** - see section 6.3.4.1 CD4.1 - Test Cases and Test Data on page 258
- **Test Results** - see section 6.3.4.2 CD4.2 - Test Results on page 258

This step creates the following deliverables:

- **Working Tested System** - see section 6.3.4.3 CD4.2 - Working Tested System on page 258

6.2.9 C9 - Perform Benchmark Tests

Benchmark tests are performed to determine the processing throughput and transaction response time of a system. The purpose of benchmark tests is to ensure that the time-related and volume-related business requirements have been met. The scope of the benchmark test is usually the same as that employed for the system as a whole.

This step updates the following deliverables:

- **Test Cases and Test Data** - see section 6.3.4.1 CD4.1 - Test Cases and Test Data on page 258
- **Test Results** - see section 6.3.4.2 CD4.2 - Test Results on page 258
- **Working Tested System** - see section 6.3.4.3 CD4.2 - Working Tested System on page 258
- **Operations Manual** - see section 6.3.3.4 CD3.4 - Operations Manual on page 257

6.2.10 C10 - Perform Acceptance Tests

6.2.10.1 C10.1 - Prepare for Acceptance Testing

Identify the acceptance test team to prepare the test cases. Members should include representatives from operations, auditing, and the user departments that will perform the procedures of the new system. Ideally, perform the testing where the production system will be used.

Plan and construct an environment to enable the testing to be performed. Test newly installed equipment. If existing equipment and facilities are to be used, devise methods for transferring the equipment and facilities between live and test data. Consider the best times of day for testing.

6.2.10.2 C10.2 - Devise the Acceptance Test Data

For each test cycle in the acceptance testing, requirements will define a series of conditions to be tested. For each test condition, prepare a test case, consisting of input data and expected results, that will prove whether or not the test condition is correctly executed by the system. Individual users prepare their own test cases and data with the aid of information management personnel.

Also prepare expected results.

This step updates the following deliverables:

- **Test Cases and Test Data** - see section 6.3.4.1 CD4.1 - Test Cases and Test Data on page 258

6.2.10.3 C10.3 - Conduct Acceptance Test

Perform acceptance testing and check actual results against the expected results. Ask programmers to correct any errors. Decisions must now be made on whether it is better to accept the system as it is, or to insist on the changes, thus delaying the completion of the project. Depending on this decision and on the problems, you may need to rework various BSD and TD tasks.

This step updates the following deliverables:

- **Test Results** - see section 6.3.4.2 CD4.2 - Test Results on page 258

6.2.10.4 C10.4 - Approve the Business System

When the acceptance testing has been completed, the system must receive formal approval from the users before implementation starts.
This ensures that implementation can progress without immediate changes.
Prepare and maintain formal documentation for system acceptance in case questions arise during Production.

This step creates the following deliverables:

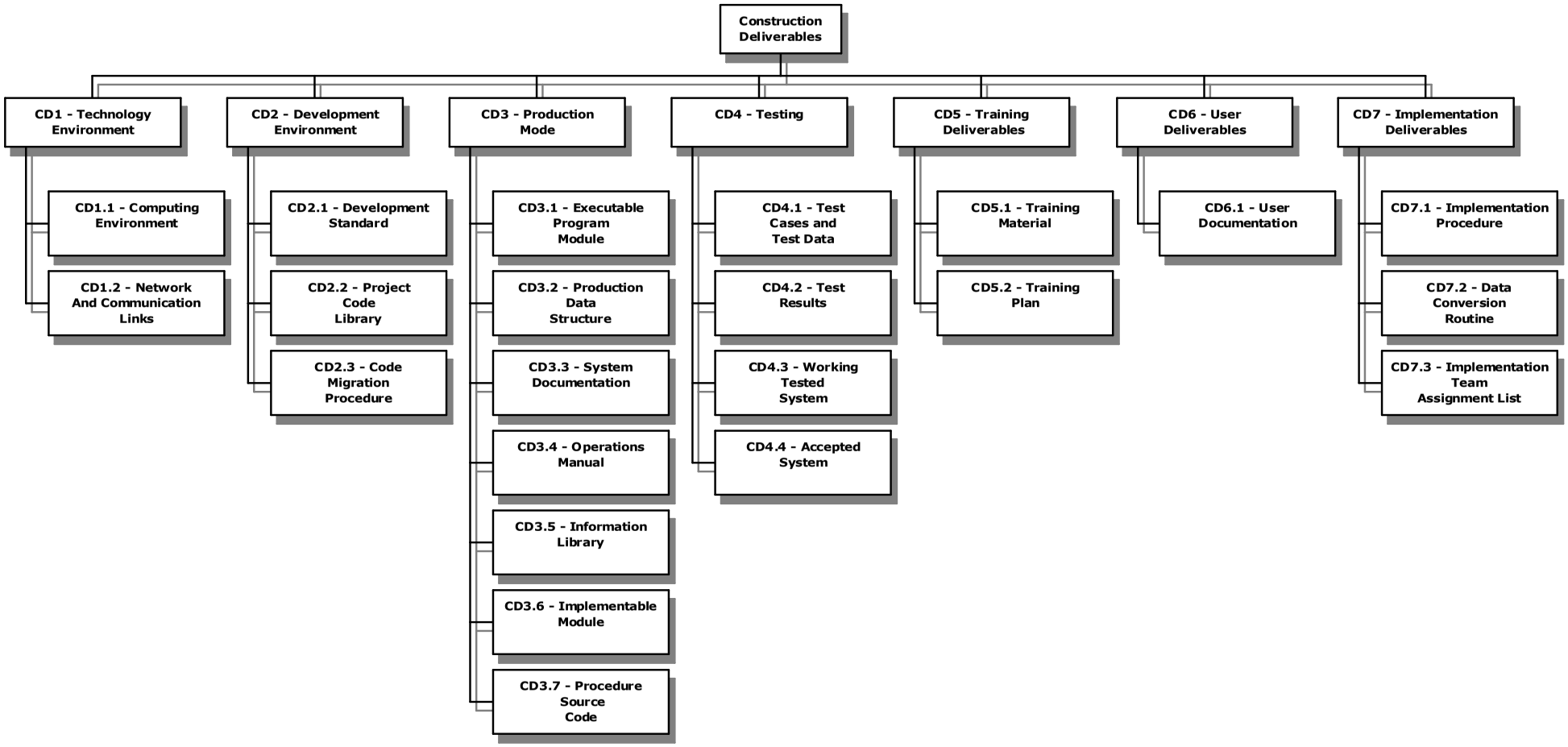
- **Accepted System** - see section 6.3.4.4 CD4.3 - Accepted System on page 258
- **User Documentation** - see section 6.3.6.1 CD6.1 - User Documentation on page 258
- **System/Procedure Replacement Agreement** - see section 7.3.4.1 ID4.1 - System/Procedure Replacement Agreement on page 272
- **Warranty and Support Agreement** - see section 7.3.4.3 ID4.3 - Warranty and Support Agreement on page 273
- **Pricing Agreement** - see section 7.3.4.2 ID4.2 - Pricing Agreement on page 273

This step updates the following deliverables:

- **Operations Manual** - see section 6.3.3.4 CD3.4 - Operations Manual on page 257
- **Information Library** - see section 6.3.3.5 CD3.5 - Information Library on page 257

6.3 Construction Phase Deliverables

Schematically, the structure of the deliverables in the construction phase is:


Construction Deliverables
CD1 - Technology Environment
CD1.1 - Computing Environment
CD1.2 - Network And Communication Links
CD2 - Development Environment
CD2.1 - Development Standard
CD2.2 - Project Code Library
CD2.3 - Code Migration Procedure
CD3 - Production Mode
CD3.1 - Executable Program Module
CD3.2 - Production Data Structure
CD3.3 - System Documentation
CD3.4 - Operations Manual
CD3.5 - Information Library
CD3.6 - Implementable Module
CD3.7 - Procedure Source Code
CD4 - Testing
CD4.1 - Test Cases and Test Data
CD4.2 - Test Results
CD4.3 - Working Tested System
CD4.4 - Accepted System
CD5 - Training Deliverables
CD5.1 - Training Material
CD5.2 - Training Plan
CD6 - User Deliverables
CD6.1 - User Documentation
CD7 - Implementation Deliverables
CD7.1 - Implementation Procedure
CD7.2 - Data Conversion Routine
CD7.3 - Implementation Team Assignment List

6.3.1 CD1 - Technology Environment

6.3.1.1 CD1.1 - Computing Environment

6.3.1.2 CD1.2 - Network and Communication Links

6.3.2 CD2 - Development Environment

6.3.2.1 CD2.1 - Development Standard

6.3.2.2 CD2.2 - Project Code Library

6.3.2.3 CD2.3 - Code Migration Procedure

6.3.3 CD3 - Production Mode

6.3.3.1 CD3.1 - Executable Program Module

6.3.3.2 CD3.2 - Production Data Structure

6.3.3.3 CD3.3 - System Documentation

6.3.3.4 CD3.4 - Operations Manual

6.3.3.5 CD3.5 - Information Library

6.3.3.6 CD3.6 - Implementable Module

6.3.3.7 CD3.7 - Procedure Source Code

6.3.4 CD4 - Testing

6.3.4.1 CD4.1 - Test Cases and Test Data

6.3.4.2 CD4.2 - Test Results

6.3.4.3 CD4.2 - Working Tested System

6.3.4.4 CD4.3 - Accepted System

6.3.5 CD5 - Training Deliverables

6.3.5.1 CD5.1 - Training Material

6.3.5.2 CD5.2 - Training Plan

6.3.6 CD6 - User Deliverables

6.3.6.1 CD6.1 - User Documentation

6.3.7 CD7 - Implementation Deliverables

6.3.7.1 CD7.1 - Implementation Procedure

6.3.7.2 CD7.2 - Data Conversion Routine

6.3.7.3 CD7.3 - Implementation Team Assignment List

7. Implementation Phase

7.1 Introduction

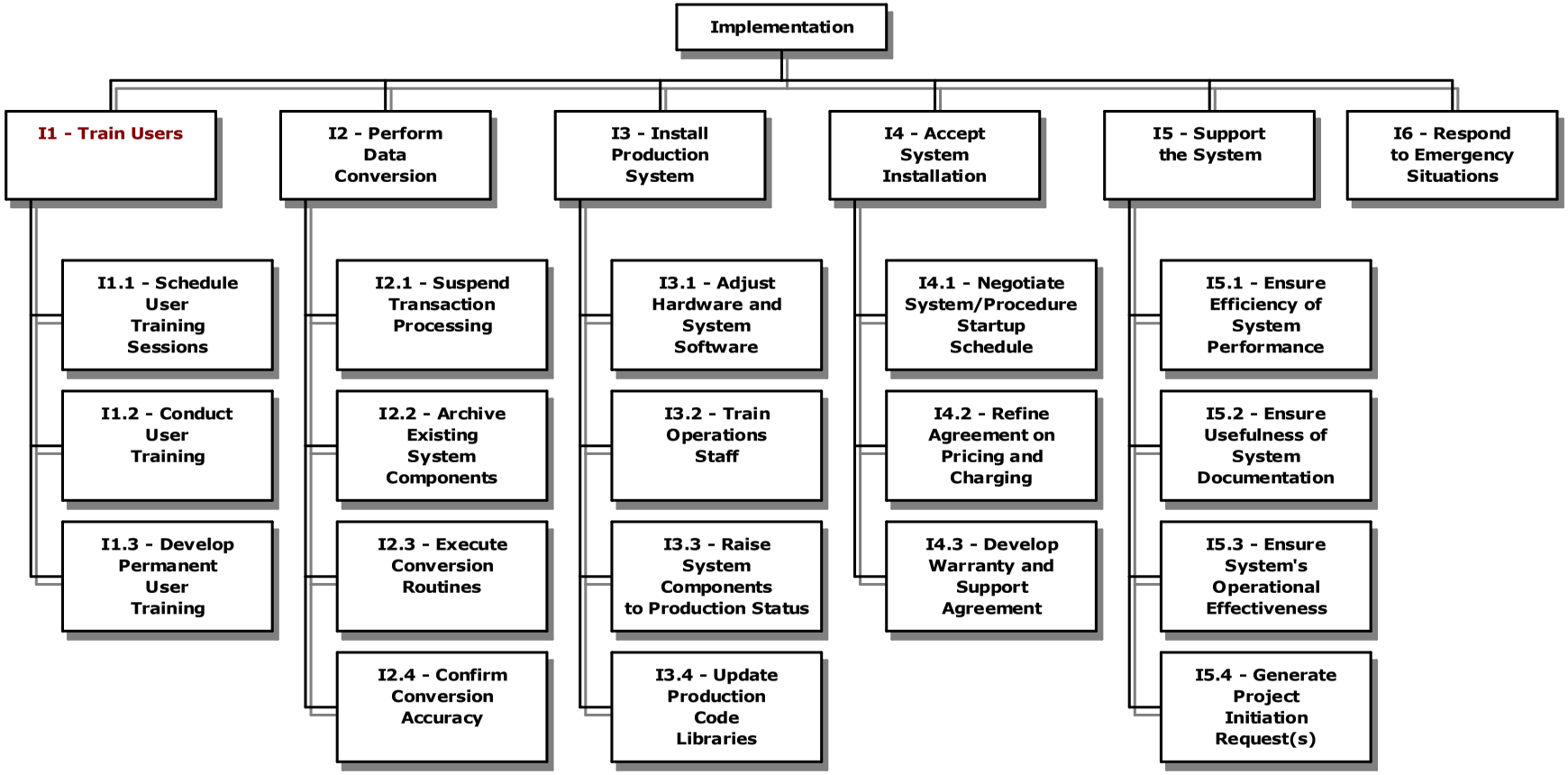
The implementable module and its documentation are deployed and placed in operational use. This includes the information system software, files and databases that are utilised by the information system, operational documentation, system and program documentation, trained staff and integration of the foregoing into the operational environment.

The scope of the phase the implementable system defined in a construction project.

The phase input is the system developed and tested during the construction phase.

The phase output is the operational system integrated with pre-existing operational systems

Schematically, the structure of the steps in the Implementation phase is:


Implementation
I1 - Train Users
I2 - Perform Data Conversion
I3 - Install Production System
I4 - Accept System Installation
I5 - Support the System
I6 - Respond to Emergency Situations
I1.1 - Schedule User Training Sessions
I1.2 - Conduct User Training
I1.3 - Develop Permanent User Training
I2.1 - Suspend Transaction Processing
I2.2 - Archive Existing System Components
I2.3 - Execute Conversion Routines
I2.4 - Confirm Conversion Accuracy
I3.1 - Adjust Hardware and System Software
I3.2 - Train Operations Staff
I3.3 - Raise System Components to Production Status
I3.4 - Update Production Code Libraries
I4.1 - Negotiate System/Procedure Startup Schedule
I4.2 - Refine Agreement on Pricing and Charging
I4.3 - Develop Warranty and Support Agreement
I5.1 - Ensure Efficiency of System Performance
I5.2 - Ensure Usefulness of System Documentation
I5.3 - Ensure System's Operational Effectiveness
I5.4 - Generate Project Initiation Request(s)

7.2 Steps in Implementation Phase

7.2.1 I1 - Train Users

7.2.1.1 I1.1 - Schedule User Training Sessions

The schedule for conducting the requisite training sessions is established by the training manager, based on the requirements outlined in the implementation plan.
The individuals who will receive the training are first identified, based on the criteria described in the training plan. Each individual is then contacted to explain the purpose of the training and to determine the most convenient schedule for their attendance.

The number of each type of training session is determined, based on the number of personnel to be trained and the desirable size of each session. A schedule for performing the training is worked out, according to the availability of the participants, the instructors and the training facilities.

Notification of individuals who will receive training should follow the organisational structure where appropriate. For example, the arrangements should be coordinated through supervisors of users, rather than going directly to the users. Supervisors will likely be involved in determining the most convenient training schedule.

All organisational issues should be resolved before training is started. Omission of a particular individual from a training session because of a planned but unannounced future organisational change, for example, could cause obvious morale problems.

This step creates the following deliverables:

- **Training Schedule** - see section 7.3.1.1 ID1.1 - Training Schedule on page 270

7.2.1.2 I1.2 - Conduct User Training

The training sessions are conducted as planned. The sessions follow the outline described in the Training Curriculum. The training materials prepared in the construction phase of the project are used in the sessions.

This step updates the following deliverables:

- **Training Material** - see section 6.3.5.1 CD5.1 - Training Material on page 258

7.2.1.3 I1.3 - Develop Permanent User Training

Permanent arrangements for training future new users of the application are established. These permanent arrangements are based on the experience of training users for the implementation of the system into production. The training curriculum may be changed as a result of the review of production operations. Usually permanent training materials are merely copies of the training materials used for implementation training that are retained for use by new employees. If a significant future training requirement is identified, these materials can be modified to reflect experience gained in conducting the initial training. A common change is deletion of any references to the prior system in the materials, since these would be irrelevant to a new employee.

This step updates the following deliverables:

- **Training Material** - see section 6.3.5.1 CD5.1 - Training Material on page 258

7.2.2 I2 - Perform Data Conversion

7.2.2.1 I2.1 - Suspend Transaction Processing

During implementation from an existing to a replacement system, a period in which all activities associated with the application are suspended is usually necessary. During this period the data associated with the old application is converted for use by the new application.

For many applications suspension of operations during the implementation process causes major business disruptions. For this reason the suspension must be carefully planned, and the period of suspension minimised.

Users should be informed well in advance of the pending suspension of application activity. Operations personnel are instructed to disable the existing application at the appropriate time.

In certain circumstances, complete suspension of all processing activity associated with an application in implementation is not possible due to valid business considerations. This situation should have been identified in the planning of the implementation, and a strategy worked out to deal with it. Usually this strategy involves some type of phased implementation, where some subdivision of users or application functionality is converted first, followed by another subset, etc.

A phased conversion implies that the old and new applications must be operated in parallel for some period of time. A phased conversion of this nature is difficult to execute, generally requires more resources to complete, and must be carefully planned and managed.

7.2.2.2 I2.2 - Archive Existing System Components

The system components of the developed software, purchased software, files and hardware of the old system should remain easily accessible for reinstallation if the new business system fails. Even when returning to the old system is impossible, you should still keep the documentation, files and software. In either case, implement archiving or storage procedures. Attend closely to these three points:

- All files should have their data-set names changed during the period between conversion to the new system and the point of no return. This change prevents inadvertent use of the old system.
- Key transactions from the old system that are not used in the new system should be changed as part of the conversion to display a message directing the user to a new transaction and documentation.
- Data files and enough documentation and software to decipher the files should be kept to support audit or government reporting requirements, as well as management requests for trend analysis. Keep the data well past the point of no return.

This step creates the following deliverables:

- **Existing System Back-Up** - see section 7.3.2.2 ID2.2 - Existing System Back-Up on page 270

7.2.2.3 I2.3 - Execute Conversion Routines

Data is converted for use by the new system in this subtask, using procedures and routines developed earlier.

Manual conversion tasks may require many personnel to perform; if so, their availability should be carefully planned. A schedule for completing large manual conversion efforts is prepared, and actual progress is tracked against it. Detailed instructions, and training if necessary, should be available to assist manual conversion efforts.

Automated conversion efforts may require extensive use of computer resources on a one-time basis. Arrangements to obtain the required resources should be made well in advance. Plans for recovering from failures of conversion routines should be in place before actual execution of the routines begins.

This step creates the following deliverables:

- **Converted System Data** - see section 7.3.2.1 ID2.1 - Converted System Data on page 270

7.2.2.4 I2.4 - Confirm Conversion Accuracy

The result of the manual and/or automated conversion effort is carefully reviewed before there is a commitment to production operations. Methods for checking the accuracy and completeness of the conversion should be available, and personnel should be assigned to review the results of each conversion step.

7.2.3 I3 - Install Production System

7.2.3.1 I3.1 - Adjust Hardware and System Software

Operation of the system may require changes in the systems environment in which it operates. These adjustments are performed in this subtask.

Typical hardware changes include the installation of additional workstations to accommodate users of the new application, addition of disk storage devices to the current environment, replacement of current central processors and addition of new communications equipment.

System software modifications usually involve changes to the operating system to accommodate hardware configuration changes, installation of new database management or teleprocessing software, or installation of new communications software.

This step updates the following deliverables:

- **Hardware Configuration** - see section 7.3.3.1 ID3.1 - Hardware Configuration on page 271
- **System Software Configuration** - see section 7.3.3.3 ID3.3 - System Software Configuration on page 271

7.2.3.2 I3.2 - Train Operations Staff

Certain tasks associated with the operation of the application within the computer centre must be performed by the operations staff. These tasks were identified previously, and detailed instructions were prepared to guide operations personnel in performing them.

These instructions are explained to the operations staff by the IS personnel on the implementation team. Make any necessary revisions to operational schedules to accommodate the new system.

7.2.3.3 I3.3 - Raise System Components to Production Status

Place in protected code libraries the components of the new system, such as documentation, source and load modules, diagrams, tables, specifications, test data, test cases, test results and job control software. Do not allow any more changes to these modules. Make any corrections or modifications by creating new versions of the modules or related modules. This control status change should occur regardless of plans to rollout the system to another site at a later date.

For audit and quality-assurance measurements, do not allow any module installed in production to be directly changed. A new version, perhaps created from a copy of the old version, is the only acceptable way to get new or different functionality from application software in production.

This step creates the following deliverables:

- **Production Code Library** - see section 7.3.3.2 ID3.2 - Production Code Library on page 271

7.2.3.4 I3.4 - Update Production Code Libraries

Production applications are usually executed from code libraries containing production versions of the software executable modules and job control statements. Before the application can become operational, the production version of the executable modules and job control statements is loaded into these code libraries using the code migration procedure.

Operational control procedures are usually established by operations personnel to prevent corruption of the production code libraries. The project manager ensures that these procedures are properly followed while installing the production software. An audit trail of the operational control procedure is established.

Changes to the job control statements are often required when converting from a test to production environment. These changes generally involve redirecting references from test to production code libraries and identification of test data sets to production data sets instead.

This step updates the following deliverables:

- **Production Code Library** - see section 7.3.3.2 ID3.2 - Production Code Library on page 271

7.2.4 I4 - Accept System Installation

7.2.4.1 I4.1 - Negotiate System/Procedure Startup Schedule

Negotiate a user agreement to replace existing systems and procedures. The user will have participated throughout the development and testing of the system.

This step creates the following deliverables:

- **System/Procedure Replacement Agreement** - see section 7.3.4.1 ID4.1 - System/Procedure Replacement Agreement on page 272

7.2.4.2 I4.2 - Refine Agreement on Pricing and Charging

This step is only performed there are internal cross-charging procedures in place where users are charged for the use of IT facilities.

During the cost-benefit analysis in design stage, you estimated the production costs of the system for items such as transactions, reports, conversions, mass updates, backup and reorganisation, and access to help and documentation. Refine these figures after construction.

The costs to be covered in setting prices are based on these estimates and include communications, computer processing, storage (e.g., direct access or other), support (e.g., operators and programmers), backup and recovery, reorganisation, performance monitoring and the billing system itself.

Pricing Considerations

Data sources versus data exploiters

There is generally a group of users or programs, whose primary purpose in an application is to provide data values for the objects of interest. This group acts as data sources. A second group of users is generally concerned with manipulating and analysing data values provided by the data sources. This second group acts as data exploiters.

The benefits of the system are usually felt by the data exploiters rather than the data sources. To ensure that the operating costs of the system do not outweigh the benefits, the exploiters are generally surcharged to cover the greater part of the system costs. Data sources are usually provided with their part of the system at or below the processing costs.

Return on investment

The costs of developing the system should have been paid before production. During negotiations on production pricing and charging, establish a plan to measure the benefits of the system in addition to the costs, and determine the real return on investment. Use the results to help you make a schedule and changes to the system, as well as to review the accuracy of the original forecast and the method for making the forecast.

User perception

At all times, users must perceive that the information management organisation is interested in, capable of and actively helping the users reduce costs or increase efficiencies. This is especially important during pricing negotiations.
Incentives

Consider financial incentives based on quality and/or productivity for data sources, exploiters and support staff. The incentives may be bonuses, or they may be included in regular performance, salary and wage reviews.
Frequency of pricing reviews

As new data exploiters and ways of exploiting the system are added, review the cost/pricing scheme. Changes in the costs of computers, communications or support personnel are also grounds for reassessment.

This step creates the following deliverables:

- **Pricing Agreement** - see section 7.3.4.2 ID4.2 - Pricing Agreement on page 273

7.2.4.3 I4.3 - Develop Warranty and Support Agreement

Develop two warranty and support agreements. The first is between the information management organisation and the end users. It concerns the responsibility of the information management organisation to correct any bugs and the charges for minor enhancements or modifications to the system.
The second agreement is between the information system development team and the production and operations team. It concerns the responsibility for corrections and modifications to the system during specified time periods during development. The production and operations team assumes the responsibility for running regularly scheduled job streams during this process.

Warranty and Support Considerations

Minor enhancements and modifications

Minor enhancements and modifications are generally paid for by the user organisation that benefits most from the change. The cost should cover specification, design, construction, testing, installation and documentation.

Agreement between teams

The warranty agreement between the development and the production and operations teams should ensure that some part of the development team is on call for the production and operations team to resolve problems and to implement minor enhancements. The length of this warranty is usually a function of the size and complexity of the system.

Team responsibilities

The production and operations team assumes responsibility for all regularly scheduled jobs as a part of the initial production process. Development team personnel provide documentation and training for scheduling and troubleshooting for the jobs.

Support staff

The information management organisation must establish a support staff that can correct any bug quickly and completely. The support staff may be comprised of application programmers, systems programmers, database administrators and end user personnel for the duration of the Warranty and Support Agreement.

Contact point for problems

An end user, representing the user community, should have a single contact point for addressing problems with the system (i.e., documentation, software or hardware). It is intended that all user issues be addressed through the single end user contact point.

This step creates the following deliverables:

- **Warranty Agreement** - see section 7.3.4.3 ID4.3 - Warranty and Support Agreement on page 273

7.2.5 I5 - Support the System

This task is performed during the period immediately following the rollout of the implementable system.

7.2.5.1 I5.1 - Ensure Efficiency of System Performance

Evaluate the implementable module's computing resource consumption.

Optimise those implementable module's components which use an abnormal quantity of computing resources.

Evaluate the implementable module's ability to meet pre-defined scheduling requirements.

Optimise those implementable module's components which are hindering the system's performance.

Evaluate interference with/from other information systems or implementable modules.

Eliminate the causes of operational interference with/from the implementable module.

7.2.5.2 I5.2 - Ensure Usefulness of System Documentation

Evaluate the user's use of the operational documentation.

Correct the deficiencies in the operational documentation.

Evaluate the effectiveness of the implementable module's training program.

Evaluate accuracy, completeness and clarity of manual and automated procedures supporting the implementable module.

Enhance the procedural documentation supporting the implementable module

7.2.5.3 I5.3 - Ensure System's Operational Effectiveness

Evaluate the usability of the products of the implementable module in the operational environment.

Identify those conditions/situations outside the scope of the implementable module interfering with the effectiveness of the implementable module.

7.2.5.4 I5.4 - Generate Project Initiation Request(s) to Correct, Extend or Modify Operational Systems

This step creates the following deliverables:

- **Project Initiation Request** - see section 7.3.5 ID5 - Project Initiation Request on page 273

7.2.6 I6 - Respond to Emergency Situations

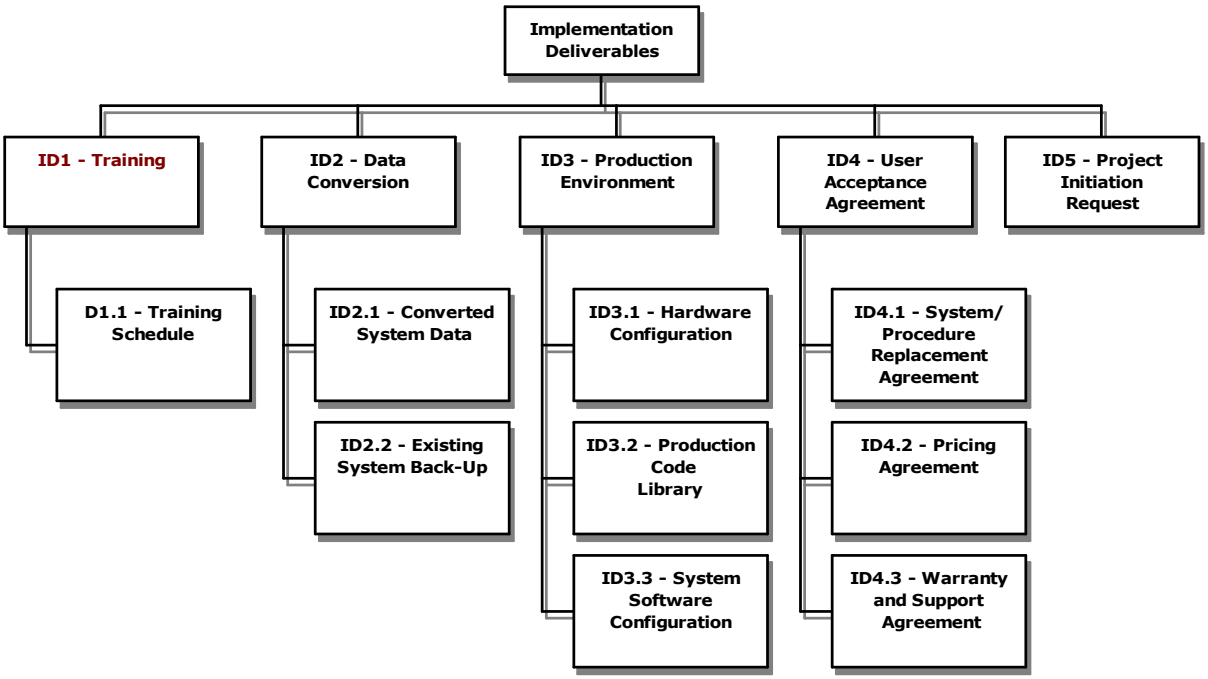
Follow the steps in the ADM in a highly compressed time-frame. A Project Initiation Request is completed or an activity log for a standing Project Initiation Request is updated to document each emergency situation.

This step creates the following deliverables:

- **Project Initiation Request** - see section 7.3.5 ID5 - Project Initiation Request on page 273

7.3 Implementation Phase Deliverables

Schematically, the structure of the deliverables in the Implementation phase is:



7.3.1 ID1 - Training

7.3.1.1 ID1.1 - Training Schedule

7.3.2 ID2 - Data Conversion

7.3.2.1 ID2.1 - Converted System Data

Data transformed from existing sources into formats accessible by the new system being placed into production operation. This data usually resides in production databases according to the data definitions contained in the modified data structure.

7.3.2.2 ID2.2 - Existing System Back-Up

The Existing System Back-up is a copy of all system files that will be modified or replaced by the new system at cutover. This copy includes the executable modules, the data structures and the data itself, and any other components of these applications that are required for them to operate. This copy is in a format such that it can be used to quickly restore the former applications to their condition prior to the cutover of the new application. This restoration may be necessary if the new application fails during or immediately after cutover.

7.3.3 ID3 - Production Environment

7.3.3.1 ID3.1 - Hardware Configuration

The physical equipment, the interconnections and communications between and among them, and their switch settings.

7.3.3.2 ID3.2 - Production Code Library

A location for all documentation and software relevant to a system in production. This includes the production load module and job control software libraries, containing versions of the application load modules necessary to operate the system.
The information library contains documentation relating to the production system.

Code Library Contents

The code library should contain the latest versions of design documentation; system documentation; production software release (including patches if they are released to users); program listings and job control software listings; output samples; integration, system and acceptance test results; issue, problem, enhancement and wish lists; and index to related project materials (e.g., working papers, methodologies, automated support tools).

Current Information

The information maintained in the information library must be about the latest versions in production. The documentation should match exactly with the production software in use. There may be enhancements and bugs that are corrected and put into production. These must be controlled to keep the documentation up to date. This can be accomplished by updating only through releases, periodic patches to the system, or special system enhancements that fall between major releases.

Software updates should not be released to the user without documentation. This is difficult to control, especially when a major bug is discovered and corrected. The project manager must determine the best way to maintain the documentation in the production code library.

7.3.3.3 ID3.3 - System Software Configuration

The parameter settings of the software that controls the operating environment.

7.3.4 ID4 - User Acceptance Agreement

A document setting out the terms under which users agree to accept a system into Production.

7.3.4.1 ID4.1 - System/Procedure Replacement Agreement

A user agreement and schedule for the replacement of previous procedures and systems. Consider the following areas when determining how best to convert from the old procedures to the new system.

Phased Versus Instantaneous Conversion

The user may want different locations or organisational units to start using the new system at different times. Although this limits risk during conversion, it demands a high ratio of programming and system support personnel to end users during the initial phases.

If the previous system had been manual (i.e., not automated), this method is convenient. If the previous system had been automated, however, the information-management personnel will have to support several incompatible versions of a system in the same business area. This is not recommended.

Phased Releases

If the new system is built and installed in phases or releases (each phase or release offering new functionality), it may be difficult during design and testing to confirm that the first release will be useful in production. The application of ADM principles will prevent this, but you should be prepared.

You may want to designate a rapid-reaction development group to provide support for capabilities. These are usually reports and displays that were not included in the first release. The group may also be required to support unknown uses of automated systems being replaced. These uses are commonly personal computing, information centres or other job streams submitted in computing centres where users submit their own work.

Conversion Schedule

If you schedule system installations with overlapping populations of end users, you will ensure minimal disruption when the users start using new procedures. Do not schedule new procedures from different development organisations in unanticipated or continuous bursts.

Work In Progress

It is important to cover the entities whose lifecycles began under the previous system and will end under the new system. Although testing should have confirmed that the system can support this case, user management may want to

schedule installation so that parts of work in progress, such as a particular order for a special customer, are either done completely with the old system or with the new system.

7.3.4.2 ID4.2 - Pricing Agreement

An optional agreement between users, the production and operations support staff, and the information management organisation about the system cost. One option of the pricing agreement is that there be "no end user charge."

7.3.4.3 ID4.3 - Warranty and Support Agreement

Agreement between users, the production and operations support staff, and the information management organisation about the responsibility for correcting deviations from the specifications and for developing minor enhancements or other minor modifications.

7.3.5 ID5 - Project Initiation Request

A document describing a business situation needing correcting or resolving.

8. Post Implementation Assessment Phase

8.1 Introduction

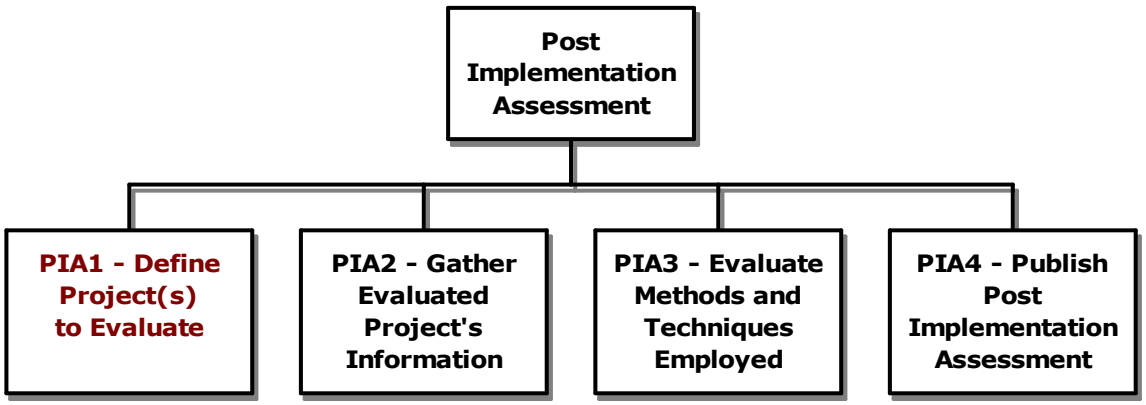
The IT project(s) which led up to and included the implementation of an operational system, a business process re-engineering project or a feasibility/assessment project are examined and evaluated to improve the ADM. The objective is to utilise the knowledge and experience acquired during the project(s) to correct flaws and secure improvements to the ADM. The PIA is not intended to evaluate the quality of the implemented system - only to assess the methods and techniques that were followed to develop and implement that system.

The scope of the phase is the group of IT development project(s), BPR or Feasibility/Assessment projects which had a common mission.

The phase input is the project team and end user experiences related to the newly implemented system.

The phase output is the assessment of the ADM components employed by the projects included in the PIA scope and recommended changes to the ADM.

Schematically, the structure of the steps in the Post Implementation Assessment phase is:



8.2 Steps in Post Implementation Assessment Phase

8.2.1 PIA1 - Define Project(s) to Evaluate

Identify projects to include in PIA.

This step creates the following deliverables:

- **Post Implementation Assessment** - see section 8.3.1 PIAD1 - Post Implementation Assessment on page 277

Define objectives of PIA.

This step updates the following deliverables:

- **Post Implementation Assessment** - see section 8.3.1 PIAD1 - Post Implementation Assessment on page 277

8.2.2 PIA2 - Gather Evaluated Project's Information

Identify scope of PIA project(s).

This step updates the following deliverables:

- **Post Implementation Assessment** - see section 8.3.1 PIAD1 - Post Implementation Assessment on page 277

Identify PIA project's participants.

This step updates the following deliverables:

- **Post Implementation Assessment** - see section 8.3.1 PIAD1 - Post Implementation Assessment on page 277

Identify users of PIA project's deliverables.

This step updates the following deliverables:

- **Post Implementation Assessment** - see section 8.3.1 PIAD1 - Post Implementation Assessment on page 277

8.2.3 PIA3 - Evaluate Methods and Techniques Employed in the Evaluated Project(s)

Locate the evaluated project's documentation.

Evaluate the evaluated project's documentation.

This step updates the following deliverables:

- **Post Implementation Assessment** - see section 8.3.1 PIAD1 - Post Implementation Assessment on page 277

Assess the evaluated project's deliverables.

Gather information from project participants. There are two general types of participants of interest to the assessment process. The technical team - analysts, modellers, designers and programmers are likely to provide information about the techniques and tools employed in the project. They will have formed opinions about their relative strengths and weaknesses as well as the obstacles encountered. The business persons will have observed the impact of some of the same techniques and tools on the project's deliverables and user community. They are likely to have opinions about the overall usefulness of some of those techniques. They will also have subjective opinions about the usability of the developed system.

This step updates the following deliverables:

- **Post Implementation Assessment** - see section 8.3.1 PIAD1 - Post Implementation Assessment on page 277

Assess the methods and techniques employed in the project(s).

This step updates the following deliverables:

- **Post Implementation Assessment** - see section 8.3.1 PIAD1 - Post Implementation Assessment on page 277

Determine Post Implementation Assessment conclusions.

This step updates the following deliverables:

- **Post Implementation Assessment** - see section 8.3.1 PIAD1 - Post Implementation Assessment on page 277

8.2.4 PIA4 - Publish Post Implementation Assessment

Identify recipients of Post Implementation Assessment. The recipients may include both business and information technology management.

This step updates the following deliverables:

- **Post Implementation Assessment** - see section 8.3.1 PIAD1 - Post Implementation Assessment on page 277

Counsel application development team on PIA findings.

Communicate PIA findings to the project's end users.

Formulate adjustments to ADM.

This step updates the following deliverables:

- **Post Implementation Assessment** - see section 8.3.1 PIAD1 - Post Implementation Assessment on page 277

Advise methodology experts of Post Implementation Assessment findings.

Publish Post Implementation Assessment.

8.3 Post Implementation Assessment Phase Deliverables

8.3.1 PIAD1 - Post Implementation Assessment

The PIA contains:

- Identification of the projects to "assess."
- Identification of the project's participants and objectives.
- Identification of the project's techniques and tools employed.
- Identification of the project's deliverables created.
- Assessment of the ADM methods and techniques employed in the project(s).
- PIA conclusions.
- Recommended changes to the ADM.
- Identification of the PIA recipients.

9. System Modification Phase

9.1 Introduction

This is the process of correcting flaws and enhancing the capability of an information system.

9.2 Steps in System Modification Phase

9.2.1 SM1 - Evaluate System

Both at regular intervals and an ad-hoc basis, benefits and costs are measured and compared with design objectives. Service levels are also measured and evaluated. This evaluation may lead to proposed changes or enhancements

9.2.2 SM2 - Assess Changes or Enhancement Requests

Requests may result from formal evaluation of the system, but more often they come from user, or information technology departments. The impact of each request must be assessed before the steps needed to carry out the changes can be defined.

9.2.3 SM3 - Analyse the Nature of the Change

The scope of the requested change must be analysed. The change may affect the structural aspects of the system (e.g., a change in the system software parameters related to performance, or a new release of the operating system),the design of the system (e.g., a change in the screen layout), or even the business model on which the system is based (e.g., a business change, such as a change in the logic of a process).

9.2.4 SM4 - Analyse the Impact of the Change

The impact of any requested change may be extensive and may even include effects on other systems. This task includes identifying the earliest point in the development path that is affected by the change, and then identifying the deliverables that are affected by tracing the task dependencies. With this information, developers can determine which tasks of each stage will have to be repeated and can estimate the cost of doing so. The quality of impact assessment is to a large extent determined by the availability of a repository that stores and relates development information.

9.2.5 SM5 - Execute the Change

Select appropriate tasks from the ADM to execute the "change." The impact analysis described above identifies the tasks of planning, analysis, design and construction that must be carried out in order to realise the change. These tasks are now executed in exactly the same way as they were originally carried out. Implementation related tasks often must be carried out as well.

10. RAD Requirements Planning Phase

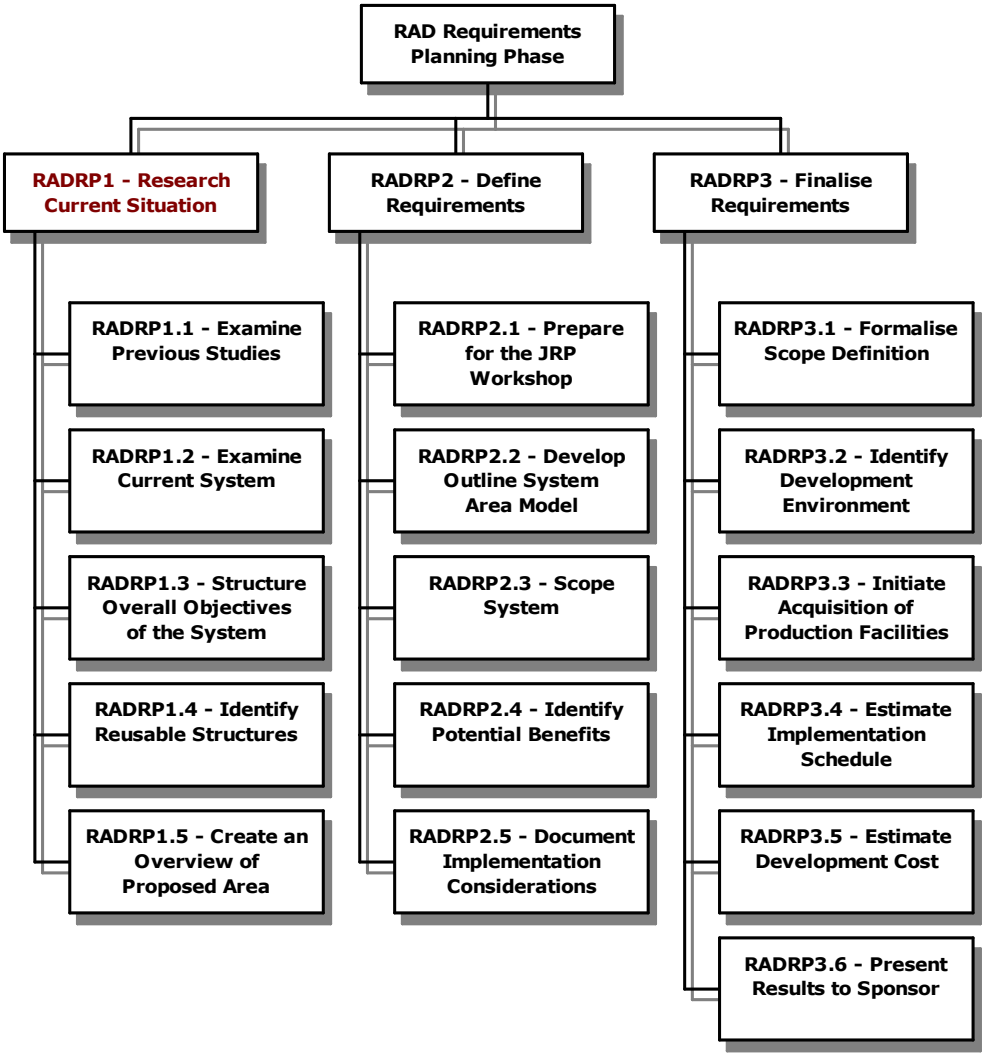
10.1 Introduction

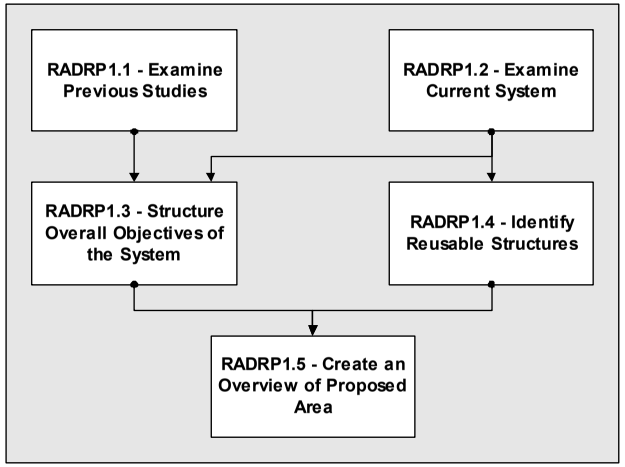
Requirements Planning is the first stage in the Rapid Application Design (RAD) Methodology Lifecycle. During this stage an outline of the system area and definition of the system scope are developed. Business executives, end users and IS professionals take part in workshops which progress through a structured set of steps, ideally with the results recorded in a CASE or some other modelling tool. Throughout the RAD Lifecycle all the information gathered is stored in the Information repository, ultimately to become a computerised design from which code can be generated. The workshops do not generate paperwork. The Requirements Planning stage typically takes one to four weeks to complete.

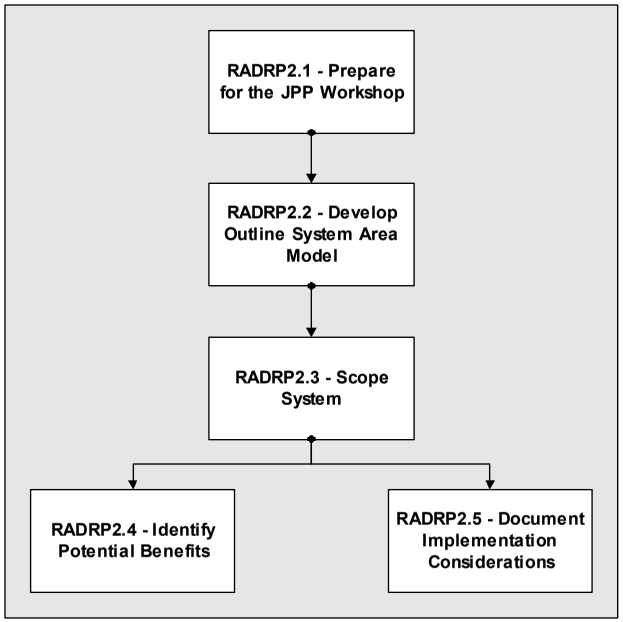
The scope of the phase is a vision of a stand-alone system.

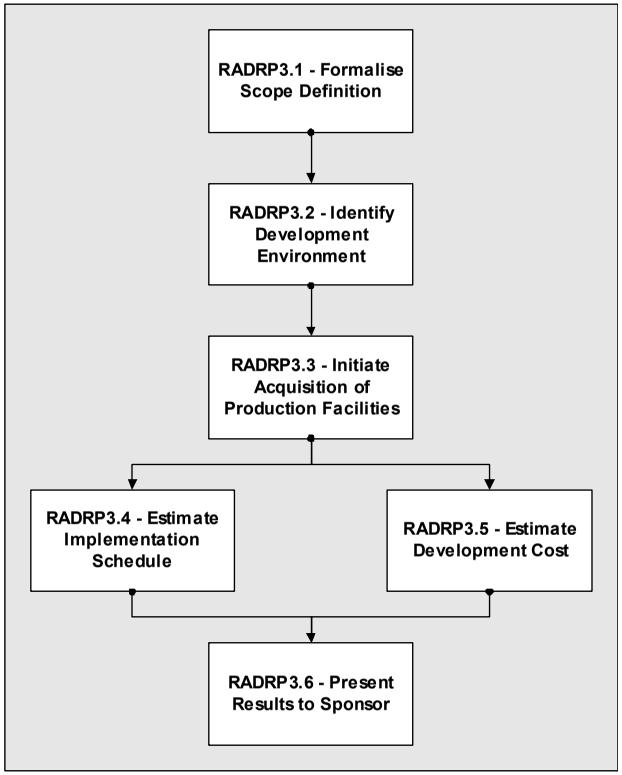
The output of the phase is a requirements definition statement and logical models to guide the user design for an information system.

Schematically, the structure of the steps in the RAD Requirements Planning phase is:


RAD Requirements Planning Phase
RADRP1 - Research Current Situation
RADRP2 - Define Requirements
RADRP3 - Finalise Requirements
RADRP1.1 - Examine Previous Studies
RADRP1.2 - Examine Current System
RADRP1.3 - Structure Overall Objectives of the System
RADRP1.4 - Identify Reusable Structures
RADRP1.5 - Create an Overview of Proposed Area
RADRP2.1 - Prepare for the JRP Workshop
RADRP2.2 - Develop Outline System Area Model
RADRP2.3 - Scope System
RADRP2.4 - Identify Potential Benefits
RADRP2.5 - Document Implementation Considerations
RADRP3.1 - Formalise Scope Definition
RADRP3.2 - Identify Development Environment
RADRP3.3 - Initiate Acquisition of Production Facilities
RADRP3.4 - Estimate Implementation Schedule
RADRP3.5 - Estimate Development Cost
RADRP3.6 - Present Results to Sponsor


RADRP1.1 - Examine Previous Studies
RADRP1.2 - Examine Current System
RADRP1.3 - Structure Overall Objectives of the System
RADRP1.4 - Identify Reusable Structures
RADRP1.5 - Create an Overview of Proposed Area


RADRP2.1 - Prepare for the JPP Workshop
RADRP2.2 - Develop Outline System Area Model
RADRP2.3 - Scope System
RADRP2.4 - Identify Potential Benefits
RADRP2.5 - Document Implementation Considerations


RADRP3.1 - Formalise Scope Definition
RADRP3.2 - Identify Development Environment
RADRP3.3 - Initiate Acquisition of Production Facilities
RADRP3.4 - Estimate Implementation Schedule
RADRP3.5 - Estimate Development Cost
RADRP3.6 - Present Results to Sponsor

10.2 Steps in RAD Requirements Planning Phase

10.2.1 RADRP1 - Research Current Situation

10.2.1.1 RADRP1.1 - Examine Previous Studies

If an enterprise model has been developed, the information stored in the information repository should be examined. Matrices that show business functions with locations, goals, problems and critical success factors should be extracted. The Enterprise Information Architecture and Business System Architecture should be studied; this will form the starting point for the Define Requirements task (which follows). If a BAA has been carried out in the proposed area, then all the relevant documentation from the repository should be extracted. This includes the entity relationship model and the process model. In this circumstance, the Requirements Planning and User Design stages should be combined.

10.2.1.2 RADRP1.2 - Examine Current System

The RAD workshop leader learns enough about the current system to become familiar with the terminology and buzzwords, examines system input and output, and takes a user guided tour of the working environment. All current system documentation is examined and relevant documents in the information repository are identified.

The RAD workshop leader should understand the user's feelings about the system. Do they like it? Is it easy to use? What changes would they like made? What additional functionality is required?

10.2.1.3 RADRP1.3 - Structure Overall Objectives of the System

The RAD workshop leader should list the overall objectives in terms of the strategic business opportunities, goals, problems and critical success factors that are relevant to the proposed system. Where an EM or BAA has taken place, there will be information available in the Information repository. Where neither study has taken place, this information must be obtained from the Sponsor and other user executives.

This step creates the following deliverables:

- **Initial Definition of Project Scope and Objectives** - see section 2.3.1.1 PID1.1 - Initial Definition Of Project Scope And Objectives on page 37

10.2.1.4 RADRP1.4 - Identify Reusable Structures

One of the goals of the advanced RAD environment is reusability. Screen templates, building blocks, application shells, entire applications, data models and documents can be designed for reusability. Any of these components that may be relevant to the application should be extracted from the repository. In addition, research should be carried out on similar systems that might offer guidance or ideas, both inside and outside the enterprise.

10.2.1.5 RADRP1.5 - Create an Overview of Proposed Area

As a result of the research carried out, a tentative overview of the system area is created in the information repository to be studied in the JRP and JAD workshops. This consists of a process decomposition, dependency diagram and an initial entity model. This task is unnecessary if a BAA has been performed.

This step creates the following deliverables:

- **Entity Type Definition** - see section 4.3.2.1 AD2.1 - Entity Type Definition on page 123
- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123
- **Process Definition** - see section 4.3.3.1 AD3.1 - Process Definition on page 125
- **Process Decomposition Diagram** - see section 4.3.3.3 AD3.3 - Process Decomposition Diagram on page 125
- **Process Dependency Diagram** - see section 4.3.3.4 AD3.4 - Process Dependency Diagram on page 126
- **Data Flow Diagram** - see section 4.3.3.7 AD3.7 - Logical Data Flow Diagram on page 126

10.2.2 RADRP2 - Define Requirements

10.2.2.1 RADRP2.1 - Prepare for the JRP Workshop

Preparation is a critical success factor for the JRP workshop. Refer to the User Workshop technique for specific guidelines about materials, participants, agenda and the workshop room.

Participants are essentially the managers of the business areas that will be directly affected by the operation of the proposed application. It may be desirable to have one or more SWAT team members present as observers to ensure clear communication and understanding of needs and priorities.

The RAD workshop leader prepares the agenda and materials for the workshop. The Sponsor holds a kick-off meeting. The meeting should be held several days before the workshop to give participants time to prepare for the workshop. The

Sponsor opens the kick-off meeting, and makes the workshop objectives clear to the planning requirements team. Participants receive information on the JRP workshop agenda, timetable and procedures. They also receive preparatory material to study and understand before the workshop.

At the start of the workshop, the RAD workshop leader reviews the agenda, purpose and objectives of the workshop. The strategic business opportunities of the system are described as well as any business assumptions that are to be made.

This step creates the following deliverables:

- **Planning Workshop Agenda** - see section 2.3.1.9 PID1.9 - Planning Workshop Agenda on page 38

10.2.2.2 RADRP2.2 - Develop Outline System Area Model

The draft diagrams describing the data and business activities that were prepared in the prior task or extracted from the results of prior planning efforts, are enhanced and corrected by the JRP workshop participants.

The relevant diagrams are the:

- Entity Relationship Diagram
- Process Decomposition Diagram
- Process Dependency Diagram
- Organisation Chart
- Business Function/Location Matrix

Verify, revise and enhance the diagrams

Each major business function that is affected by the system is discussed. First the accuracy of these diagrams and matrices is verified; then adjustments are made as necessary. Individual business functions are decomposed into processes. The entity types associated with the business processes are analysed in parallel with the process decomposition. The entity relationship diagrams relevant to the proposed system are expanded. The workshop participants revise and enhance the diagrams and create definitions of entity types and processes. This interaction is directed by the workshop leader.

This effort is focused on identifying business processes that will be supported by the proposed system. RAD focuses on building a system. Time should not be allowed to broadly expand the information architecture for the enterprise in general; these efforts must be concentrated on areas relevant to the proposed system.

Record the revised diagrams

The scribe enters the results of the group interaction in the CASE tool, polishes the information, and feeds the results back to the group as revised diagrams. A

data modelling expert may participate in the workshop as a visiting expert or may assist the scribe outside the workshop and assure the quality of the data modelling.

This step updates the following deliverables:

- **Entity Type Definition** - see section 4.3.2.1 AD2.1 - Entity Type Definition on page 123
- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123
- **Process Definition** - see section 4.3.3.1 AD3.1 - Process Definition on page 125
- **Process Decomposition Diagram**- see section 4.3.3.3 AD3.3 - Process Decomposition Diagram on page 125
- **Process Dependency Diagram** - see section 4.3.3.4 AD3.4 - Process Dependency Diagram on page 126
- **Data Flow Diagram** - see section 4.3.3.7 AD3.7 - Logical Data Flow Diagram on page 126
- **Process/Entity Type Matrix** - see section 4.3.4.2 AD4.2 - Process/Entity Type Matrix on page 127

10.2.2.3 RADRP2.3 - Scope System

The scope of the system is defined by the selection of the closely associated entity types and processes that support business functions to be automated by the new system.

The selection of the appropriate entity types and processes is made using Cluster Analysis techniques. These will also assist in identifying subsystems that could be developed separately. The processes are prioritised and those that can be implemented in the next version of the system are identified. Eliminate the processes which are of questionable value or may be difficult to build.

This step creates the following deliverables:

- **System Scope Definition** - see section 4.3.10.2 AD10.2 – System Scope on page 135
- **Project Plan** - see section 4.3.1.2 AD1.2 - Project Plan on page 122

10.2.2.4 RADRP2.4 - Identify Potential Benefits

JRP workshop participants identify quantitative and qualitative benefits associated with the implementation of the proposed system. A statement describing these benefits is prepared.

Benefits should be related to specific business processes selected for inclusion within the scope of the application. Benefits will usually be identified in parallel

with the selection of business functions to be supported. When possible, some basis for estimating the cost impact of each benefit should be provided.

This step creates the following deliverables:

- **System Benefits Definition** - see section 4.3.10.1 AD10.1 - System Benefits Definition on page 135

10.2.2.5 RADRP2.5 - Document Implementation Considerations

In the final stages of the workshop the participants identify and discuss cultural and technical considerations that could affect the development and implementation of the proposed system.

Typical Implementation Considerations

The following considerations should be discussed:

- Availability of users to participate in the development process
- Development standards and procedures
- Audit and security requirements
- Customer service objectives limiting implementation options
- Implementation
- Location and condition of existing data that must be converted
- An overview of the changes to existing system

Typical Cultural Changes

New systems can also cause dramatic cultural changes in an organisation. Some reorganisation may be desirable in order to take maximum advantage of the new system. Jobs may be displaced or new types of talent may be necessary. The staffing levels often need to be changed. These cultural factors need to be addressed as early as possible in the development lifecycle. It may take considerable time to accomplish the necessary changes and, if not addressed well, could cause the system to fail. Cultural changes may affect:

- Organisational structure
- Policies
- Business procedures
- Job content
- Skills requirements

Organisational changes

It is necessary to assign responsibility for bringing about the organisational changes. The nominated manager should review potential changes with top management. The benefits of the change must be understood by a level of management high enough to put the change into effect.

An outline plan should be created for an orderly transition to the new organisation. The actions required, the resources necessary and the people responsible should be listed. This set of actions must be meshed with the overall development and implementation plan. Consensus and approval should be obtained from the managers affected by the changes.

This step creates the following deliverables:

- **Implementation Considerations Definition** - see section 5.4.2.3 DD2.3 - Implementation Considerations Definition on page 211
- **Project Problems and Issues List** - see section 4.3.1.1 AD1.1 - Project Problems and Issues List on page 122

This step updates the following deliverables:

- **Project Plan** - see section 4.3.1.2 AD1.2 - Project Plan on page 122

10.2.3 RADRP3 - Finalise Requirements

10.2.3.1 RADRP3.1 - Formalise Scope Definition

The scope of the proposed system is formally described in the system scope definition. This documentation obtained from the information repository contains the process decomposition diagram, identifying the business functions to be addressed by the system, the entity relationship diagram and process dependency diagrams. Also included will be the process/entity matrix, which identifies the interactions of the data and processes. The documentation created during the JRP workshop will be completed and a final report obtained from the Information repository.
The data administrator reviews the scope definition to check that its contents make sense from a business perspective and that they fit into the overall enterprise information architecture.

This step creates the following deliverables:

- **Cost Benefit Analysis** - see section 4.3.7.2 AD7.2 - Cost-Benefit Analysis on page 130

This step updates the following deliverables:

- **System Scope Definition** - see section 4.3.10.2 AD10.2 – System Scope on page 135

10.2.3.2 RADRP3.2 - Identify Development Environment

The facilities necessary to construct and test the system software must be identified, including the physical environment, hardware and software.

Workstations

The facilities needed to construct the system are determined first. The number of workstations required depends on the size and number of "timebox teams." Every developer should have their own workstation equipped with the selected software

Equipment to Simulate the Production Environment

Equipment simulating the production environment must be available for testing purposes. The configuration of this equipment should be similar to the actual production configuration, to provide a reliable test. The capacity of this equipment usually is modest. If volume testing is performed, equipment is required that is similar in capacity to the actual production configuration, or equipment that can be reliably scaled up to that configuration.

Development Software

A shared encyclopaedia for all workstations is required. Often this encyclopaedia is held on a mainframe computer or a remote server to which all workstations can be connected. Other implementations use a PC or file server resident encyclopaedia linked to a network.

Workspace for Team Members

Workspace arrangements vary greatly between sites. To facilitate communications, the SWAT team(s) and construction assistance team should be located together, and preferably in the same building as the eventual users of the system. Rapid development requires some degree of isolation of the team members from day-to-day activities that could disrupt development efforts. There should be no unnecessary visitors or meetings, as it has been demonstrated repeatedly that a good work environment more than pays for itself in increased productivity.

This step updates the following deliverables:

- **Project Problems and Issues List** - see section 4.3.1.1 AD1.1 - Project Problems and Issues List on page 122

10.2.3.3 RADRP3.3 - Initiate Acquisition of Production Facilities

The existing hardware and systems software configuration may be inadequate to support the projected processing requirements of the system about to be built. This configuration includes central processors, data storage devices, communications facilities, workstations, operating systems, database management software and teleprocessing software.

Acquisition and installation of additional facilities of this nature often involves considerable delays. For this reason, an assessment of the adequacy of the current facilities to support the new application is made immediately upon starting the development of the software. If additional facilities are needed, arrangements for their acquisition are promptly initiated.

Assessment of the Adequacy of Current Facilities

Hardware platform requirements

Usually this requirement is expressed as the number of database transactions per unit time that will be imposed on the processing system. This assessment should focus on the one or two system procedures that are performed most frequently.

Data storage requirements

This requirement is expressed in bytes of data. Most systems have one or two major tables (files); the analysis should focus on these. The period of data retention significantly affects the storage requirements.

Number of workstations

New systems are often accessed by users not previously involved in automated data processing. These users must be equipped with new workstations.

Communications facilities

Systems involving distributed processing, distributed databases, and/or remotely located workstations require some type of communications facilities. This requirement often can be satisfied by existing communications links, but these should be reviewed to confirm their capacity and reliability in light of the new system's requirements.

System software

The new application may impose some unusual requirement on the operating system, database management system, or teleprocessing system. Often this involves upgrading currently installed software with newer versions.

Processing requirements of the proposed system

This assessment is constrained by the level of detail of its design at this stage in the development Lifecycle. The techniques used to estimate the processing requirements of the application should recognise this limitation. Extensive, detailed analysis is inappropriate.

Evaluating the Current Configuration Against Requirements

Evaluating the current configuration against the requirements of the new system is difficult. Other factors often must be considered, such as:

- age and status of the current facilities
- expected growth of processing demand associated with existing and other new applications
- general, long-range objectives of the enterprise's information processing department

Once a requirement to acquire additional hardware or system software is identified, arrangements should be immediately put in place for this acquisition. If acquisition delays threaten the planned system implementation schedule, the Sponsor should be informed, and adjustments to the schedule made as necessary.

This step creates the following deliverables:

- **Computing Environment** - see section 6.3.1.1 CD1.1 - Computing Environment on page 257
- **Network and Communications Links** - see section 6.3.1.2 CD1.2 - Network and Communication Links on page 257

10.2.3.4 RADRP3.4 - Estimate Implementation Schedule

A tentative implementation schedule for the system is prepared. This tentative schedule is based on preliminary assumptions about the implementation approach. The Rapid Construction stage of the development uses Timebox Management techniques. The timebox will be decided based on the size of the system, but rarely exceeds 12 weeks. For a large system, consideration should be given to the construction approach of either one step, parallel development of parts, or serial development of versions of increasing scope or functionality. This is a preliminary decision, for estimating purposes; a final decision will be made at the completion of the User Design stage.
With phased development, the initial version will implement the core functions of the system, while later versions will add additional functionality until the ultimate definition of the system has been implemented.

A tentative schedule for implementation is established based on the implementation approach, assumptions related to availability of system resources, user participation in development tasks, and management commitment to the project.

This step updates the following deliverables:

- **Project Plan** - see section 4.3.1.2 AD1.2 - Project Plan on page 122

10.2.3.5 RADRP3.5 - Estimate Development Cost

A function-point analysis of the proposed system is prepared, based on business functions that it will support. Using estimating guidelines developed from

experience with previous RAD projects, a general estimate of the total implementation cost is prepared.

Accurate estimation of implementation costs based on the general definition of the system is not possible. However, some basis for deciding if the system can be cost-justified is needed. The estimate of implementation cost is a general indication of the cost. The results of the cost estimation are used to revise the system benefits definition determined in the JRP workshop to give a complete cost benefit description for the system.

This step updates the following deliverables:

- **Cost Benefit Analysis** - see section 4.3.7.2 AD7.2 - Cost-Benefit Analysis on page 130

10.2.3.6 RADRP3.6 - Present Results to Sponsor

The system scope description, along with the preliminary cost benefit analysis and preliminary implementation schedule, are presented to the Sponsor to obtain approval to proceed with the User Design stage. The user coordinator is responsible for the presentation.

The specific format of this presentation will vary widely from one project to another. The presentation may be very short and informal, or eliminated entirely, if approval to proceed is unnecessary at this stage in the development process. The Sponsor should make a quick decision to give the go-ahead for the User Design stage, and release the funds for it.

This step updates the following deliverables:

- **System Scope Definition** - see section 4.3.10.2 AD10.2 – System Scope on page 135
- **Cost Benefit Analysis** - see section 4.3.7.2 AD7.2 - Cost-Benefit Analysis on page 130
- **Project Plan** - see section 4.3.1.2 AD1.2 - Project Plan on page 122

11. RAD User Design Phase

11.1 Introduction

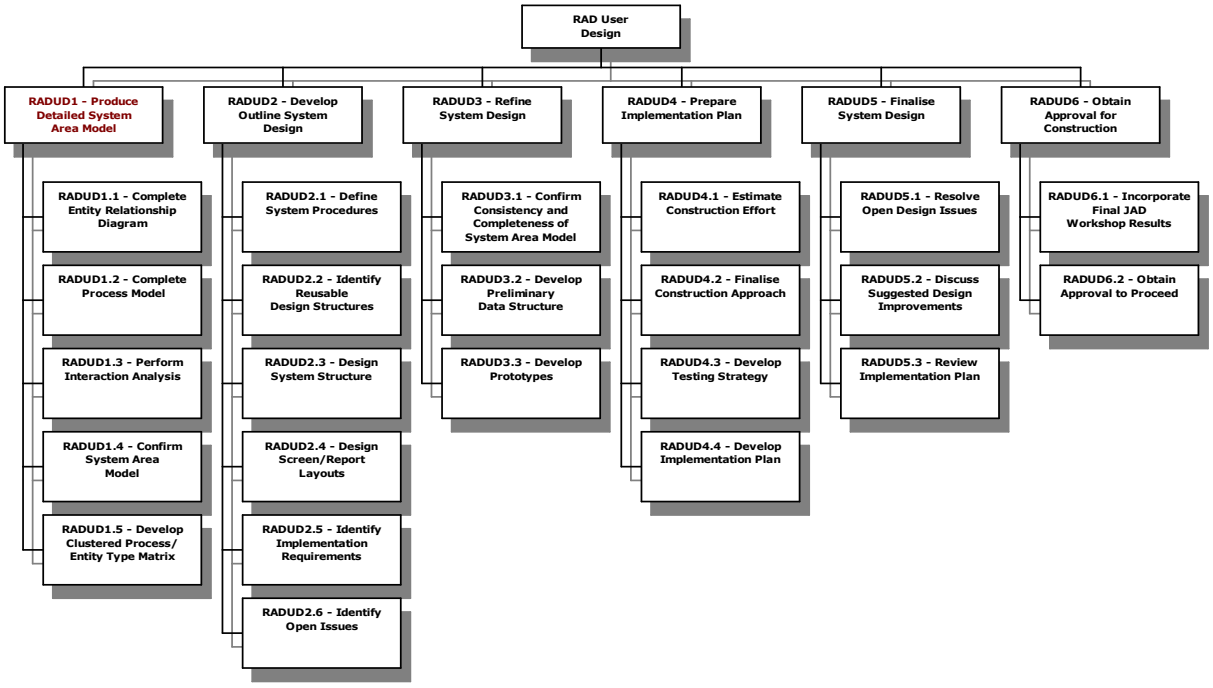
The User Design stage produces a detailed system area model, an outline system design and an implementation plan.

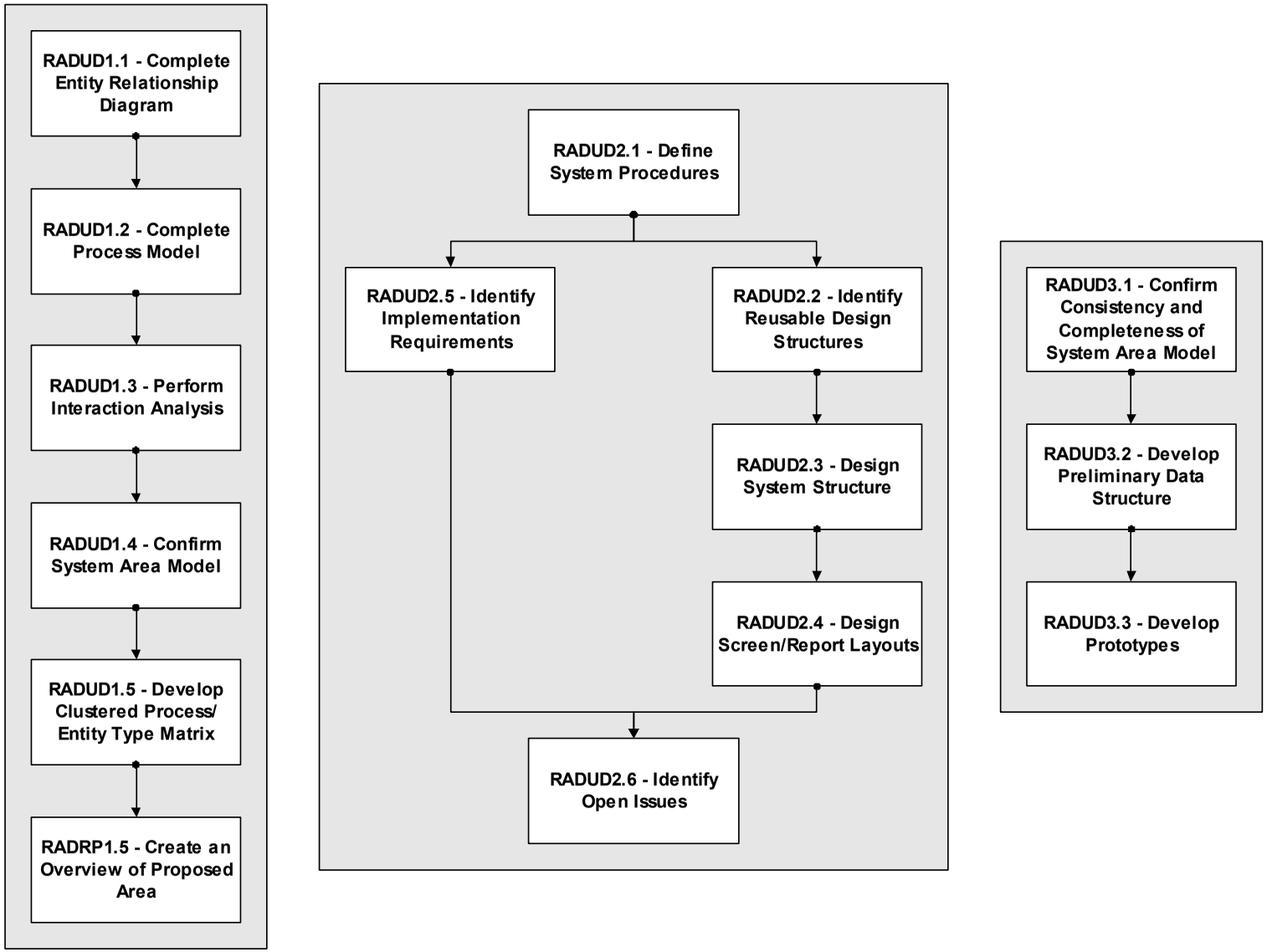
End users participating in JAD workshops perform the analysis and design tasks associated with this stage. These tasks are led by the RAD workshop leader and assisted by IS professionals. The results of the workshops are recorded in the CASE tool. Information in the repository developed in the JRP workshop or from a BAA project is used as input. If a BAA has been completed then the workshop focuses on the design of the system with a subsequent reduction in the time scale required for this stage. The User Design stage typically takes 3 to 5 weeks to complete.

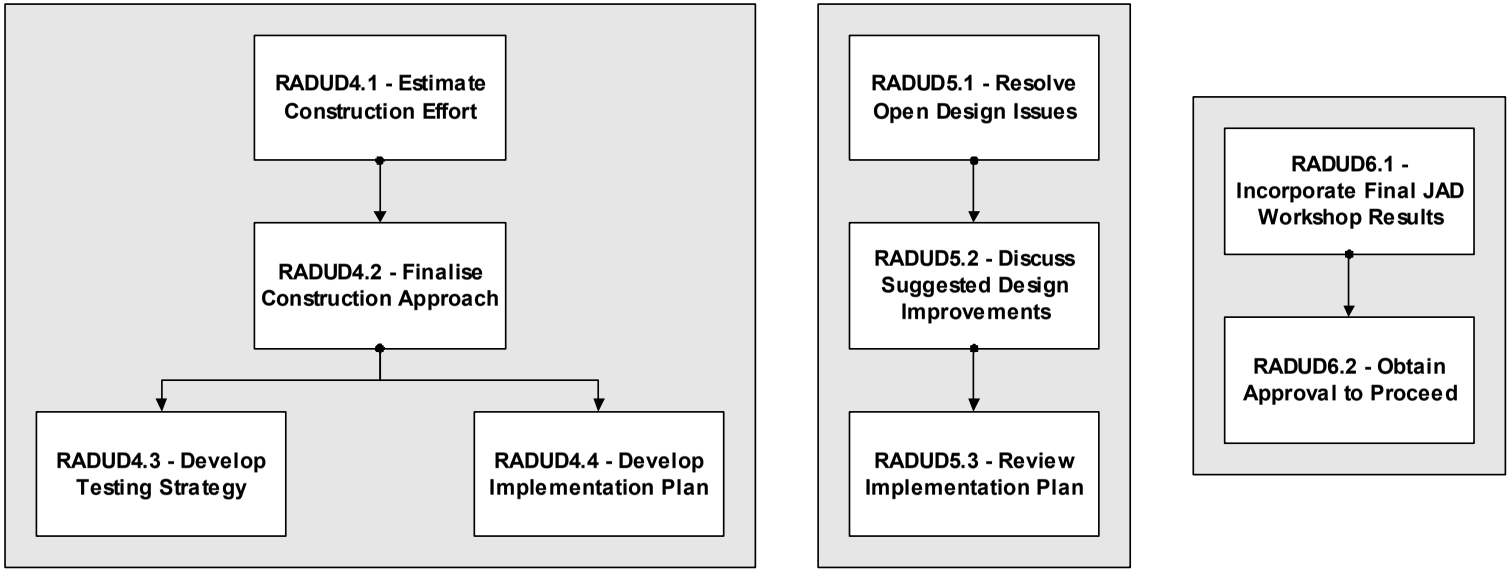
The scope of the phase is the concept of a stand-alone system expressed in terms of organisation units, business locations, business activities or entity types.

The phase input is a requirements definition statement and logical models to guide the user design for an information system.

Schematically, the structure of the steps in the RAD Requirements Planning phase is:




RADUD1.1 - Complete Entity Relationship Diagram
RADUD1.2 - Complete Process Model
RADUD1.3 - Perform Interaction Analysis
RADUD1.4 - Confirm System Area Model
RADUD1.5 - Develop Clustered Process/ Entity Type Matrix
RADRP1.5 - Create an Overview of Proposed Area
RADUD2.1 - Define System Procedures
RADUD2.5 - Identify Implementation Requirements
RADUD2.2 - Identify Reusable Design Structures
RADUD2.3 - Design System Structure
RADUD2.4 - Design Screen/Report Layouts
RADUD2.6 - Identify Open Issues
RADUD3.1 - Confirm Consistency and Completeness of System Area Model
RADUD3.2 - Develop Preliminary Data Structure
RADUD3.3 - Develop Prototypes


RADUD4.1 - Estimate Construction Effort
RADUD4.2 - Finalise Construction Approach
RADUD4.3 - Develop Testing Strategy
RADUD4.4 - Develop Implementation Plan
RADUD5.1 - Resolve Open Design Issues
RADUD5.2 - Discuss Suggested Design Improvements
RADUD5.3 - Review Implementation Plan
RADUD6.1 - Incorporate Final JAD Workshop Results
RADUD6.2 - Obtain Approval to Proceed

11.2 Steps in RAD User Design Phase

11.2.1 RADUD1 - Produce Detailed System Area Model

11.2.1.1 RADUD1.1 - Complete Entity Relationship Diagram

The JAD workshop participants complete the detailed data model as defined in the Requirements Planning stage. The development of the entity model may be performed in parallel with the development of the process model by the use of working groups within the JAD workshop.

Name and define attribute types

It is important that the attribute type is defined and named properly and that it is associated with an entity type. It is possible that new entity types may be identified when attempting to assign the attribute type to an entity type. Lastly, examine the resulting attribute types of each entity type and remove synonyms. Replace any foreign attribute types with relationship types.

Identify entity subtypes

The attribute types, relationship types and entity type definitions are examined to decide if it would be useful to define any entity subtypes. Wherever a subtype is proposed a classifying attribute type must also be defined. The relevant attribute types and relationship types are transferred from the entity type to the subtype.

Identify possible entity types for synthesis

The entity relationship model is examined to see whether any existing entity types could be synthesised into a single entity type. This can be a valuable way of rationalising the model in cases where two (or more) entity types have several corresponding relationship types and attribute types. The original entity types can then be defined as subtypes if they are left with special attribute types or relationship types of their own.

Define attribute type properties

The remaining properties of each attribute type are recorded; for derived attribute types the derivation algorithm should be recorded. Where it is known that attribute values are restricted to particular values, this information is recorded.

This step updates the following deliverables:

- **Entity Type Definition** - see section 4.3.2.1 AD2.1 - Entity Type Definition on page 123

- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123

11.2.1.2 RADUD1.2 - Complete Process Model

JAD workshop participants complete the process model, down to the elementary process level. The decomposition process focuses on the activities directly related to the proposed system.

The steps in this sub-task are:

1. The decomposition diagram is decomposed until the elementary process level is reached.
2. Process dependency diagrams or data flow diagrams are developed for each business event that initiates one or more elementary processes.
3. The definition and purpose of each elementary process and quantitative information such as frequency and time of usage are recorded.

This step updates the following deliverables:

- **Process Definition** - see section 4.3.3.1 AD3.1 - Process Definition on page 125
- **Process Decomposition Diagram** - see section 4.3.3.3 AD3.3 - Process Decomposition Diagram on page 125
- **Process Dependency Diagram**- see section 4.3.3.4 AD3.4 - Process Dependency Diagram on page 126
- **Data Flow Diagram** - see section 4.3.8.1 AD8.1 - Physical Data Flow Diagram on page 132
- **Business Event List** - see section 4.3.3.5 AD3.5 - Business Event List on page 126
- **Business Event Model** - see section 4.3.3.6 AD3.6 - Business Event Model on page 126

11.2.1.3 RADUD1.3 - Perform Interaction Analysis

The principal features of the elementary processes and process interactions with entity types, attribute types, and relationship types are examined and documented by the JAD workshop participants.

Entity Type Life Cycle Analysis

Entity type life cycle analysis is performed for those entity types that have complicated structures, such as many subtypes, or that are involved with many processes. For each of these entity types, the user design team determines the entity states that apply to it. Next, the elementary processes that cause each state change are identified. This may reveal missing elementary processes, inaccurate process definitions and entity types that are not well defined.

To complete the analysis, all elementary processes are reviewed to determine whether they involve the selected entity type. Also determined are those attribute types and relationship types whose values can identify the entity state. If none exist, then a status attribute type is defined.

This step updates the following deliverables:

- **Process/Entity Type Matrix** - see section 4.3.4.2 AD4.2 - Process/Entity Type Matrix on page 127
- **Entity State Diagram** - see section 4.3.4.7 AD4.7 - Entity State Diagram on page 128
- **Entity Type Definition** - see section 4.3.2.1 AD2.1 - Entity Type Definition on page 123
- **Process Definition** - see section 4.3.3.1 AD3.1 - Process Definition on page 125

This step creates the following deliverables:

- **Entity State Matrix** - see section 4.3.4.8 AD4.8 - Entity State Matrix on page 128

Process Logic Diagram Development

For each elementary process the user design team identifies the entity types to be created/modified/terminated by the process, the relationship types that are associated/disassociated by the process, the selection criteria for the entities, and existence checks for entity modification/termination or disassociation. As the actions are identified, they are numbered to indicate the logical sequence in which they may occur. The action number will show a feasible sequence of actions, though it is not necessarily the only possible sequence. The process logic diagram is drawn on top of (over) a subset of the entity relationship diagram.

This step creates the following deliverables:

- **Process Logic Diagram** - see section 4.3.4.4 AD4.4 - Process Logic Diagram on page 127

Process Action Diagram Development

The logic of each elementary process is then expressed in the form of a process action diagram. This is built by including all actions shown on the process logic diagram in the specified sequence. In addition, control statements are inserted in the action diagram to control the execution of conditional groups or repeating groups of actions,

This step creates the following deliverables:

- **Process Action Diagram** - see section 4.3.4.9 AD4.9 - Process Action Diagram on page 128

Attribute Action Identification

Attribute actions are identified and added to the process action diagram. Business algorithms are defined in the form of process action blocks. The information repository should be checked for definitions of business algorithms that can be used and reused in the proposed system.

This step creates the following deliverables:

- **Business Algorithm** - see section 4.3.2.4 AD2.4 - Business Algorithm on page 124

Exception Situation Consideration

Exception situations and their subsequent activities are considered and documented in the process action diagram.

This step updates the following deliverables:

- **Process Action Diagram** - see section 4.3.4.9 AD4.9 - Process Action Diagram on page 128

Action Diagram Review

The user design team reviews the action diagrams to ensure that the process logic conforms with the process definition and other aspects of the process models and that all entity types, relationship types and attribute types appearing within the process logic diagram and process action diagrams are incorporated in the entity relationship model.

This step updates the following deliverables:

- **Entity Relationship Diagram** - see section 4.3.2.3 AD2.3 - Entity Relationship Diagram on page 123
- **Entity Type Definition** - see section 4.3.2.1 AD2.1 - Entity Type Definition on page 123

11.2.1.4 RADUD1.4 - Confirm System Area Model

The process model and entity relationship model developed in the JAD workshop is reviewed by all the participants to check for consistency and completeness.

A listing of issues and items outstanding produced for the workshop participants to review and correct should be produced.

This step updates the following deliverables:

- **System Scope Definition** - see section 4.3.10.2 AD10.2 – System Scope on page 135

11.2.1.5 RADUD1.5 - Develop Clustered Process/Entity Type Matrix

The process/entity type matrix is refined with details of the actions of the elementary processes on the entity types in the form of create, read, update and delete (CRUD). This matrix is then clustered to show logical groupings of processes and entity types. If the system to be constructed is large and complex then parallel development is utilised. The clustered matrix is used to identify possible subsystems. If the system is to be developed by a single SWAT team then the groupings of processes and entity types are used to allocate parts of the system development to the individual SWAT team members.

This step updates the following deliverables:

- **Process/Entity Type Matrix** - see section 4.3.4.2 AD4.2 - Process/Entity Type Matrix on page 127

11.2.2 RADUD2 - Develop Outline System Design

11.2.2.1 RADUD2.1 - Define System Procedures

JAD workshop participants examine each process in turn to select the most appropriate technique or techniques to implement it, taking into account: time constraints for the process, volumes, type of user, design objectives of system, and technical context (e.g., hardware/software available).

One-to-One Mapping

Firstly, an attempt is made to map each process to a single procedure. One-to-one mapping is most desirable because:

- Each elementary process is a complete activity from the user's point of view, and hence the procedure will be meaningful to the user, and will leave the database in a consistent state
- A change in an elementary process (due to a business change) will only require a change to the single procedure

Alternative or Serial Procedures

Consideration is given to the use of alternative or serial procedures to implement the process. The details of the procedure, including its name, definition, techniques, constraints, and the elementary processes it supports are recorded in the Information repository.

- **Alternative Procedures**

More than one procedure may be required if the same process is performed in different processing environments, by users with different skill levels, or on different hardware or software platforms.

- **Serial Procedures**

Several processes may best be performed by one procedure, termed a compound procedure. This may be an appropriate mapping when multiple processes are always performed in series, several similar processes involve the same entity type, or procedures are generalised.

This step creates the following deliverables:

- **Procedure Definition** - see section 5.4.1.1 DD1.1 - Procedure Definition on page 209

11.2.2.2 RADUD2.2 - Identify Reusable Design Structures

The information repository is examined to identify any reusable structures that could be used in the design. Such reusable structures could be:

- Procedures
- Templates such as specimen screens
- Dialogs or transactions
- Building blocks such as algorithms, structures or knowledge of great complexity

Consideration should also be given to reusing entire applications that may be stored in the information repository; they may be modified to fit the requirements specified. The identified reusable structures are made available to the JAD workshop participants to incorporate into their system design.

11.2.2.3 RADUD2.3 - Design System Structure

Dialog structures for the automated, interactive procedures associated with the proposed system are developed by the JAD workshop participants. This dialog structure, which is documented using a dialog flow diagram, defines the flows between individual procedures and the nature of the user interaction with the system. At this stage it is only necessary to define the procedures and the major flows between them. This provides a basis for prototyping the screens and interactions between them. The detailed dialog flow is completed during the Rapid Construction stage.

Manual procedures that are directly related to the automated procedures of the application are also designed in this subtask, using similar techniques. Additionally, a data flow diagram for the complete system should be developed to show all on-line, batch and manual procedures, as well as the major data stores.

Alternately, the procedures can be represented with a structure chart allowing for more detail regarding data transmittal and procedure-calling elements.

This step creates the following deliverables:

- **Dialog Flow Definition** - see section 5.4.7.1 DD7.1 - Dialog Flow Definition on page 217
- **Dialog Flow Diagram** - see section 5.4.7.2 DD7.2 - Dialog Flow Diagram on page 217
- **Structure Chart** - see section 5.4.17.6 DD17.6 - Structure Chart on page 230

11.2.2.4 RADUD2.4 - Design Screen/Report Layouts

The JAD workshop participants develop rough layouts of the most important screens and reports. These screens and reports are associated with each automated, interactive application procedure that has been identified. The appearance of these screens and reports is a critical element of the user interface to the application.

Standards for the design of screen layouts that have been previously established within the enterprise are applied when appropriate.

This step creates the following deliverables:

- **Screen/Report Layout** - see section 5.4.8.1 DD8.1 - Screen/Report Layout on page 217

11.2.2.5 RADUD2.5 - Identify Implementation Requirements

Implementation activities that affect the end users are identified by the workshop participants. The activities to be considered should include:

- Changes required in user procedures
- Changes required in paperwork
- Phasing in the new system
- User training requirements
- Creation of user documentation

This step creates the following deliverables:

- **Implementation Considerations Definition** - see section 5.4.2.3 DD2.3 - Implementation Considerations Definition on page 211
- **Implementation Strategy** - see section 4.3.7.4 AD7.4 - Implementation Strategy on page 132

11.2.2.6 RADUD2.6 - Identify Open Issues

More significant cultural changes such as organisational, policy or management issues should have been identified, discussed, resolved and planned for earlier during JRP.

As the JAD workshop is completed, participants develop a list of issues that have not been resolved, and require further investigation before agreement on their solution is reached. These issues are listed in an open issues document, along with a brief summary of the actions required for their resolution.

This step updates the following deliverables:

- **Project Problems and Issues List** - see section 4.3.1.1 AD1.1 - Project Problems and Issues List on page 122

11.2.3 RADUD3 - Refine System Design

11.2.3.1 RADUD3.1 - Confirm Consistency and Completeness of System Area Model

The documentation of the system area models and outline design developed in the initial JAD workshop is analysed to confirm its technical correctness and consistency. This will have been an ongoing process throughout the project to ensure errors and inconsistencies are removed as quickly as possible.

As the review is carried out, issues not addressed by the workshop participants may be identified. If possible, the members of the construction team resolve the issues based on their own judgment and assessment of the intent of the workshop participants. If the issue cannot be resolved without the attention of the workshop participants, it is added to the list of open issues.

This step updates the following deliverables:

- **Project Problems and Issues List** - see section 4.3.1.1 AD1.1 - Project Problems and Issues List on page 122.

11.2.3.2 RADUD3.2 - Develop Preliminary Data Structure

The system architect develops a preliminary data structure based on the entity relationship model constructed during the first JAD workshop.

In addition the entity relationship model usage is summarised; this information is used to prioritise the entry points that will be supported.

The preliminary data structure is used as the basis for the database design that takes place at the beginning of the Rapid Construction stage.

This step creates the following deliverables:

- **Data Structure Specification** - see section 5.4.5.3 DD5.3 - Data Structure Specification on page 215
- **Data Structure Diagram** - see section 5.4.5.9 DD5.9 - Data Structure Diagram on page 217

11.2.3.3 RADUD3.3 - Develop Prototypes

The members of the construction team will develop prototypes of the screens and dialog flows that had been outlined in the workshop. A prototype is developed to show the users the screens and interactions between screens. The prototype does not usually contain processing logic.

A human factors expert examines the prototype and gives the developers help in achieving clarity, comprehension and consistency.

The prototype is made available to the user participants of the workshop for review. Based on their comments, enhancements to the screen designs and dialog flows can be made. Usability testing is also performed on prototypes.

This step updates the following deliverables

- **Dialog Flow Definition** - see section 5.4.7.1 DD7.1 - Dialog Flow Definition on page 217
- **Dialog Flow Diagram** - see section 5.4.7.2 DD7.2 - Dialog Flow Diagram on page 217
- **Screen/Report Layout** - see section 5.4.8.1 DD8.1 - Screen/Report Layout on page 217

This step creates the following deliverables:

- **Procedure Dependency Diagram** - see section 5.4.1.2 DD1.2 - Procedure Dependency Diagram on page 209

11.2.4 RADUD4 - Prepare Implementation Plan

11.2.4.1 RADUD4.1 - Estimate Construction Effort

Using the Function Point Calculation (see section 21 Appendix 9 – Function Point Analysis on page 452 for a brief introduction to this topic) technique or another technique, develop a quantitative measurement of the complexity of the proposed system as a means of estimating the construction effort.

Function Points

The Function Point Calculation technique calculates the number of function points for an application. The calculation is based on the number of screens, reports, interfaces, etc. that are associated with the application. The results of the system design, performed in the previous task, provide the information on which the calculation can be based.

The raw function point figure is usually adjusted to reflect the environment in which the application will be developed and will operate. The method for performing these adjustments is incorporated into the Function Point Calculation technique.

The calculated number of function points for the proposed system is an indication of its complexity and can be used to predict the effort necessary for its implementation. This prediction requires experience, since the effort-per-function point necessary to build any system varies widely.

Metrics and Timebox Estimation

The rapid construction effort for the system should also be estimated using metrics obtained from previous RAD projects. A combination of the results provides the best estimate of the effort required. If the time estimated to construct the system exceeds the timebox, then an item is added to the open issues list to review the system scope.

This step creates the following deliverables:

- **Construction Approach** - see section 5.4.21.3 DD21.3 - Construction Approach on page 236

11.2.4.2 RADUD4.2 - Finalise Construction Approach

One of the basic criteria used to determine the most appropriate construction approach is system complexity. One of several techniques for establishing complexity levels is Function Point Calculation.

For relatively simple applications (i.e., fewer than 1,000 function points), a one-step approach is recommended. The SWAT team and construction assistance team work together to develop the application. The construction activity of this group should not exceed three months duration.

If the system is more complex (i.e., over 1,000 function points), it may require more than three months of effort by a single construction team. In such situations, a different construction approach should be considered.
¤ Multiple Version implementation

The easiest approach to constructing a complicated application in a timebox is to develop it in multiple versions. The first version of the system will be limited in functionality so that development by a single SWAT team within a three-month

period is practical. After the first version is built and made operational, a second version is developed. This second version adds functionality onto the first version, but it, too, is built within the three-month period. Additional versions of the application are developed, if necessary, until the full functionality of the system is implemented as designed. However, practical considerations may limit the use of the multiple version implementation approach. Many applications cannot effectively operate without full functionality; subdividing them into versions is not possible. If the application is being installed at multiple locations, attempting to re-install new versions of an application may create unacceptable complications.

Parallel Development Approach

For these complex applications which are not amenable to versioning, a parallel development approach is applied. Several SWAT teams are established, each working in parallel on a portion of the application. Again, the objective is to limit the work by any one SWAT team to a three-month period, but parallel construction allows much larger applications to be built within that time-frame. However, parallel development creates coordination problems . Although the subsystems can be built and tested independently, they must fit together with absolute precision. To help coordinate the work of separate teams, the business model should be divided using cluster analysis into relatively autonomous, highly cohesive subdivisions. In a parallel development, the business model is termed a coordinating model.

This step updates the following deliverables:

- **Construction Approach** - see section 5.4.21.3 DD21.3 - Construction Approach on page 236

11.2.4.3 RADUD4.3 - Develop Testing Strategy

The extent of testing of the system that will be required during its construction is determined. This determination is based on many factors, including:

- The extent to which reusable design objects can be used in the system development
- The complexity of the procedures comprising the application
- The environment in which the application will be developed, including the development software to be employed
- Performance requirements imposed upon the application
- The potential consequences of discovering errors after the system is operational, including the difficulty of correcting them

Based on this evaluation, a general strategy for testing the system during development is developed. This strategy identifies the general types of tests that will be conducted, and describes a general testing scenario to be followed by all testing tasks.

- **Testing Strategy**- see section 5.4.20.8 DD20.8 - Testing Strategy on page 235

11.2.4.4 RADUD4.4 - Develop Implementation Plan

A plan is developed outlining the major activities that must be undertaken to achieve transition to the new system. A number of the tasks required for implementation are carried out during the Rapid Construction stage. Each of these components is identified and a plan prepared for their development. Consideration should be given to:

- Preparation for cultural changes, which may include changes in organisational structure, policies, business procedures, job content and skill requirements
- Design activities required for implementation, which are divided into those that affect the end users and those that require IT professional design without strong user involvement
- Conversion and production procedures must be identified and planned
- Documentation necessary for both implementation and subsequent operation and maintenance (Technical documentation is largely available from the Information repository. User documentation should be on-line with little or no external paperwork documentation.)
- Plan for user training which is developed in parallel with the construction of the system, including the determination of training needs and objectives, curriculum development, and design and production of training materials

This step creates the following deliverables:

- **Implementation Plan** - see section 5.4.2.4 DD2.4 - Implementation Plan on page 211

11.2.5 RADUD5 - Finalise System Design

11.2.5.1 RADUD5.1 - Resolve Open Design Issues

Unresolved design issues are discussed and resolved by the workshop participants. These issues were identified at the conclusion of the first JAD workshop or during the refinement process that occurred thereafter.

Each issue is completely described, and potential solutions are suggested. If necessary, one of the participants is designated to conduct further research and to report results back to the group.

As the second JAD workshop nears its conclusion, some design issues may be left unresolved. These are documented and left for resolution during the Rapid Construction stage.

This step updates the following deliverables

- **Project Problems and Issues List** - see section 4.3.1.1 AD1.1 - Project Problems and Issues List on page 122

11.2.5.2 RADUD5.2 - Discuss Suggested Design Improvements

Since the first JAD workshop, participants will have developed suggestions for improving the design of the system. These suggestions are based on experience in reviewing the prototypes, or upon further research into a particular design issue.

These suggestions are fully discussed in the JAD workshop. If adopted, the design is altered to reflect the changes.

Changes that involve significant modification of the original scope of the system, as established in the Requirements Planning stage, are discussed with the Sponsor before being approved.

This step updates the following deliverables

- **Construction Approach** - see section 5.4.21.3 DD21.3 - Construction Approach on page 236
- **Implementation Plan** - see section 5.4.2.4 DD2.4 - Implementation Plan on page 211
- **Dialog Flow Definition** - see section 5.4.7.1 DD7.1 - Dialog Flow Definition on page 217
- **Dialog Flow Diagram** - see section 5.4.7.2 DD7.2 - Dialog Flow Diagram on page 217

11.2.5.3 RADUD5.3 - Review Implementation Plan

The draft implementation plan is presented to the JAD workshop participants. The testing and implementation strategies on which the implementation plan is based are explained. Workshop participants discuss these issues and suggest improvements based on their experience in the enterprise.

These discussions are focused on issues that the workshop participants can contribute to, based on their experience. These include the availability of user personnel during the Rapid Construction and Implementation stages, the practicality of the implementation strategy, data conversion issues, etc.
The potential areas of risk also are discussed. Workshop participants are asked to suggest steps that could minimise these risks and to identify risk situations not already envisioned.

This step updates the following deliverables

- **Testing Strategy**- see section 5.4.20.8 DD20.8 - Testing Strategy on page 235
- **Implementation Plan** - see section 5.4.2.4 DD2.4 - Implementation Plan on page 211
- **Implementation Strategy** - see section 4.3.7.4 AD7.4 - Implementation Strategy on page 132

11.2.6 RADUD6 - Obtain Approval for Construction

11.2.6.1 RADUD6.1 - Incorporate Final JAD Workshop Results

The various documents describing the system design, the construction approach, the testing strategy and implementation plan are modified to reflect the discussions in the final JAD workshop (if those changes were not made immediately).

The documentation is then organised in a format suitable for presentation to the Sponsor.

This step updates the following deliverables

- **Structure Chart** - see section 5.4.17.6 DD17.6 - Structure Chart on page 230
- **Testing Strategy**- see section 5.4.20.8 DD20.8 - Testing Strategy on page 235
- **Screen/Report Layout** - see section 5.4.8.1 DD8.1 - Screen/Report Layout on page 217
- **Dialog Flow Diagram** - see section 5.4.7.2 DD7.2 - Dialog Flow Diagram on page 217
- **Data Structure Diagram** - see section 5.4.5.9 DD5.9 - Data Structure Diagram on page 217
- **Implementation Plan** - see section 5.4.2.4 DD2.4 - Implementation Plan on page 211
- **Construction Approach** - see section 5.4.21.3 DD21.3 - Construction Approach on page 236

11.2.6.2 RADUD6.2 - Obtain Approval to Proceed

The system design and implementation plan are presented to the Sponsor for approval to proceed with the Rapid Construction stage.

The content and format of this presentation varies considerably, depending on the specific circumstances. If the Sponsor has been involved in the design process, this subtask may be a formality. However, if the design and planning tasks have been conducted independently from the Sponsor, a more formal and complete presentation is necessary.

12. RAD Rapid Construction Phase

12.1 Introduction

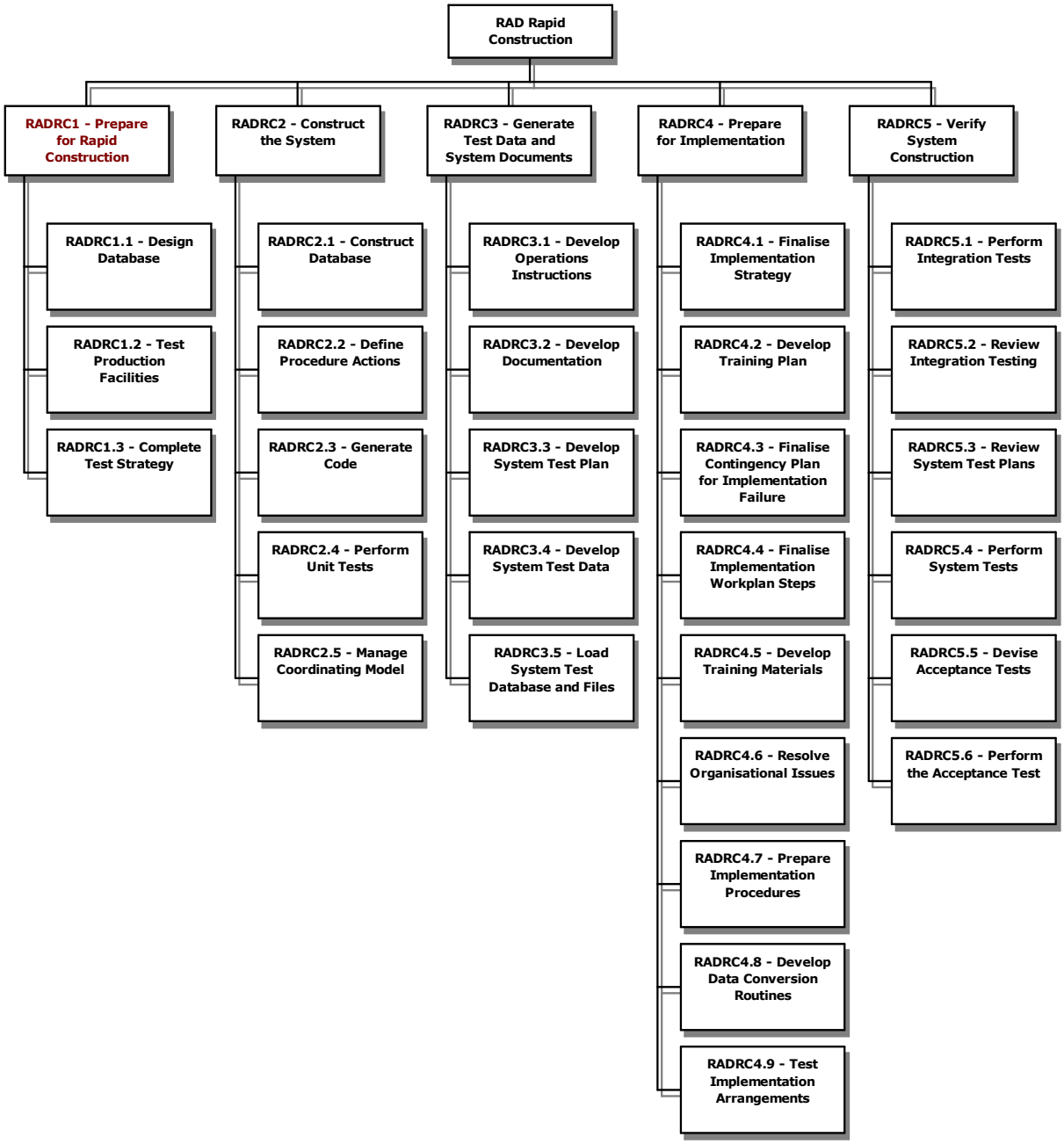
The design of the proposed system, initially described in the User Design stage, is completed in the Rapid Construction stage, and application software to implement that design is developed and tested. Tasks necessary in preparation for the transition of the system to production status are also performed.

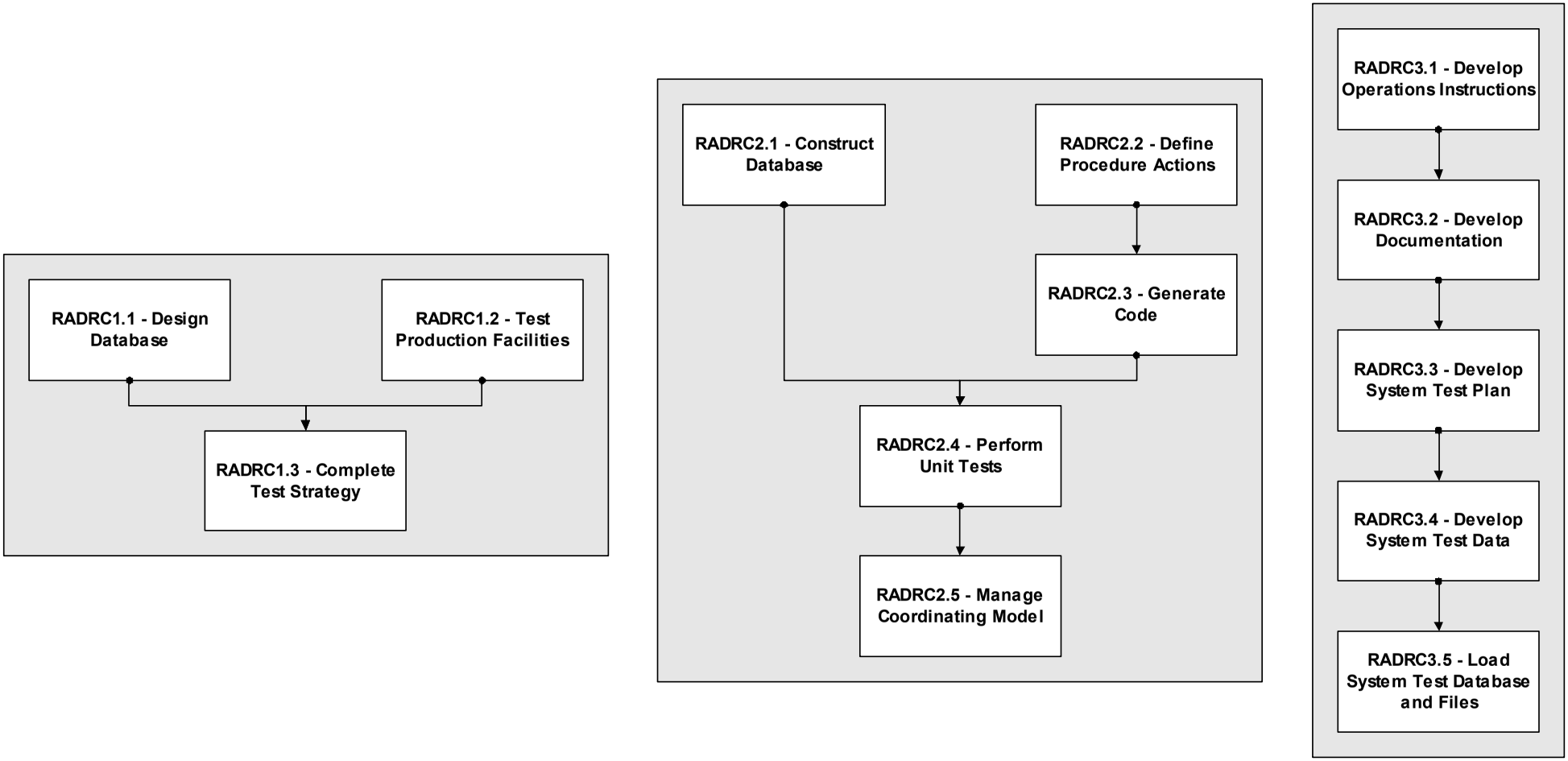
The scope of the phase is a system expressed in terms of its business and technology design.

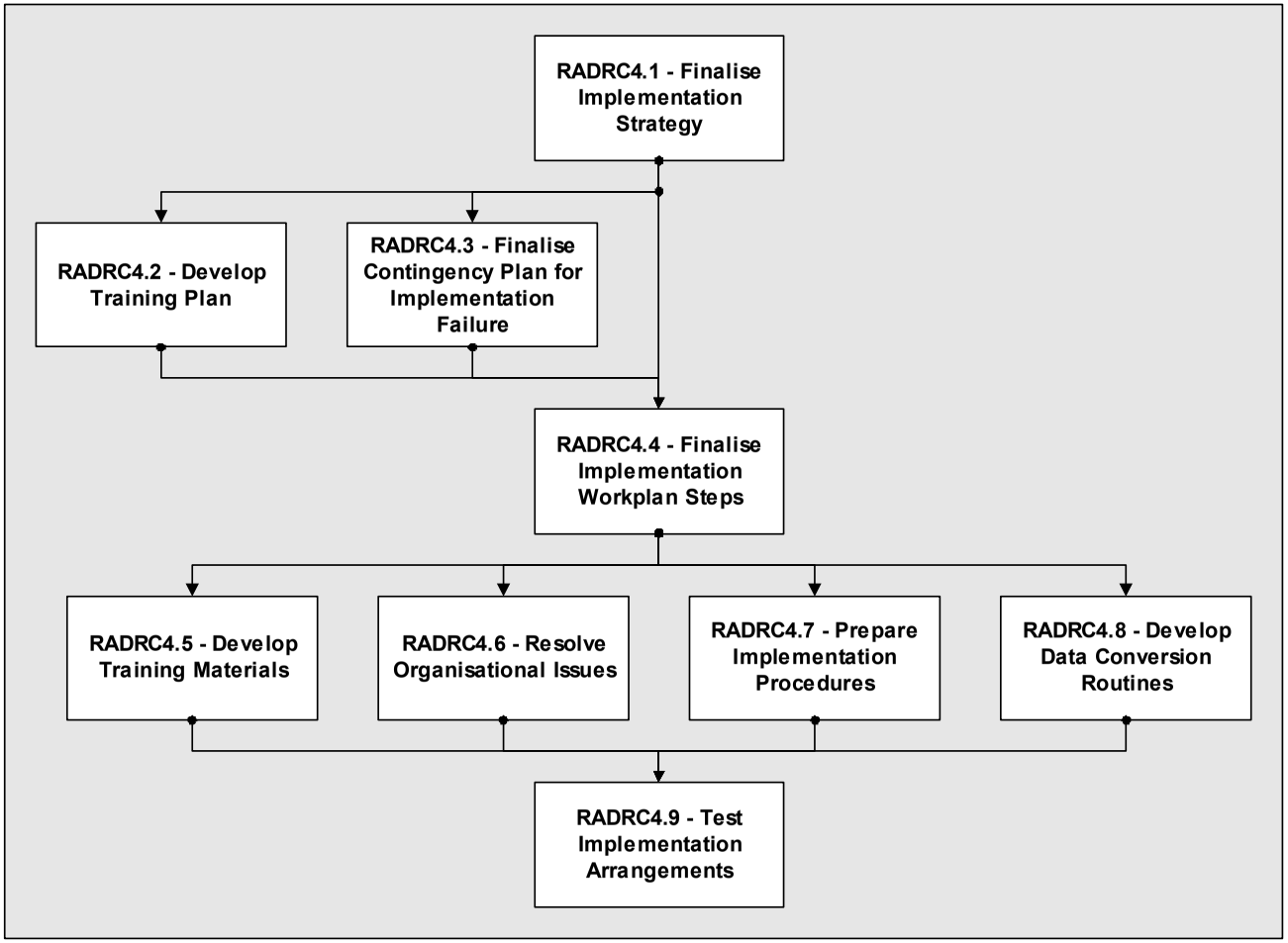
The phase input is the business and technology design of an information system expressed through structure charts, data structure diagrams, screen/report layouts and dialog diagrams.

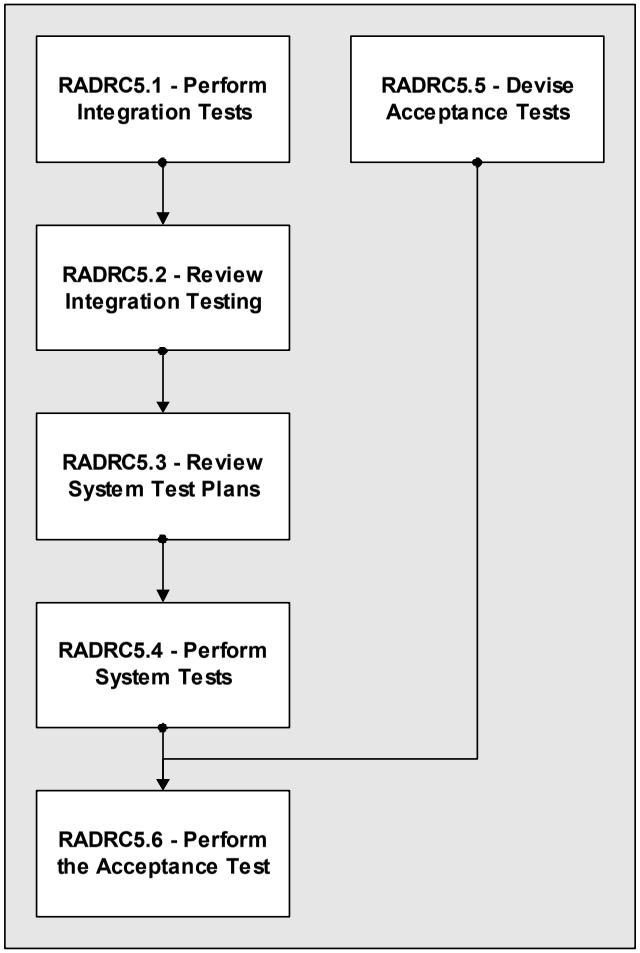
The phase output is an operational system with all documentation and training completed.

Schematically, the structure of the steps in the RAD Requirements Planning phase is:


RAD Rapid Construction
RADRC1 - Prepare for Rapid Construction
RADRC2 - Construct the System
RADRC3 - Generate Test Data and System Documents
RADRC4 - Prepare for Implementation
RADRC5 - Verify System Construction
RADRC1.1 - Design Database
RADRC1.2 - Test Production Facilities
RADRC1.3 - Complete Test Strategy
RADRC2.1 - Construct Database
RADRC2.2 - Define Procedure Actions
RADRC2.3 - Generate Code
RADRC2.4 - Perform Unit Tests
RADRC2.5 - Manage Coordinating Model
RADRC3.1 - Develop Operations Instructions
RADRC3.2 - Develop Documentation
RADRC3.3 - Develop System Test Plan
RADRC3.4 - Develop System Test Data
RADRC3.5 - Load System Test Database and Files
RADRC4.1 - Finalise Implementation Strategy
RADRC4.2 - Develop Training Plan
RADRC4.3 - Finalise Contingency Plan for Implementation Failure
RADRC4.4 - Finalise Implementation Workplan Steps
RADRC4.5 - Develop Training Materials
RADRC4.6 - Resolve Organisational Issues
RADRC4.7 - Prepare Implementation Procedures
RADRC4.8 - Develop Data Conversion Routines
RADRC4.9 - Test Implementation Arrangements
RADRC5.1 - Perform Integration Tests
RADRC5.2 - Review Integration Testing
RADRC5.3 - Review System Test Plans
RADRC5.4 - Perform System Tests
RADRC5.5 - Devise Acceptance Tests
RADRC5.6 - Perform the Acceptance Test


RADRC1.1 - Design Database
RADRC1.2 - Test Production Facilities
RADRC1.3 - Complete Test Strategy
RADRC2.1 - Construct Database
RADRC2.2 - Define Procedure Actions
RADRC2.3 - Generate Code
RADRC2.4 - Perform Unit Tests
RADRC2.5 - Manage Coordinating Model
RADRC3.1 - Develop Operations Instructions
RADRC3.2 - Develop Documentation
RADRC3.3 - Develop System Test Plan
RADRC3.4 - Develop System Test Data
RADRC3.5 - Load System Test Database and Files


RADRC4.1 - Finalise Implementation Strategy
RADRC4.2 - Develop Training Plan
RADRC4.3 - Finalise Contingency Plan for Implementation Failure
RADRC4.4 - Finalise Implementation Workplan Steps
RADRC4.5 - Develop Training Materials
RADRC4.6 - Resolve Organisational Issues
RADRC4.7 - Prepare Implementation Procedures
RADRC4.8 - Develop Data Conversion Routines
RADRC4.9 - Test Implementation Arrangements


RADRC5.1 - Perform Integration Tests
RADRC5.5 - Devise Acceptance Tests
RADRC5.2 - Review Integration Testing
RADRC5.3 - Review System Test Plans
RADRC5.4 - Perform System Tests
RADRC5.6 - Perform the Acceptance Test

12.2 Steps in RAD Rapid Construction Phase

12.2.1 RADRC1 - Prepare for Rapid Construction

12.2.1.1 RADRC1.1 - Design Database

The database administrator is responsible for designing the database using the Data Structure and Storage Design technique and making it available to the SWAT teams. Changes to the data structure definition are recorded in the shared encyclopaedia in readiness for database generation.

The following steps are performed:

1. Develop the preliminary data structure (consisting of a diagram and encyclopaedia entries) into a refined data structure based on the various design options provided by the database software to ensure satisfactory performance.
2. Define the data storage structure in terms of the datasets (physical files) that will be used to store the database. Define the database records that will be allocated to each dataset, as well as the physical characteristics of the datasets and their physical arrangement upon the storage medium.
3. Calculate the disk space required for the data taking into account the expected growth in the number of entities of each type.

This step creates the following deliverables:

- **Dataset Summary** - see section 5.4.18.4 DD18.4 - Dataset Summary on page 230
- **Production Data Structure** - see section 6.3.3.2 CD3.2 - Production Data Structure on page 257

This step updates the following deliverables

- **Data Structure Specification** - see section 5.4.5.3 DD5.3 - Data Structure Specification on page 215
- **Data Structure Diagram** - see section 5.4.5.9 DD5.9 - Data Structure Diagram on page 217
- **Implementation Team Assignment List** - see section 6.3.7.3 CD7.3 - Implementation Team Assignment List on page 258
- **System Construction Assignment List** - see section 5.4.21.4 DD21.4 - System Construction Assignment List on page 236

12.2.1.2 RADRC1.2 - Test Production Facilities

Equipment simulating the production environment must be available for testing purposes. To provide a reliable test, the configuration of this equipment should

be similar to the actual production configuration. The capacity of this equipment is usually modest. However, volume testing, if performed, requires equipment that is similar in capacity to the actual production configuration, or equipment that can be reliably scaled up to that configuration. If any of the production facilities are newly acquired, then complete testing is required to ensure that the actual production configuration can support the new system.

This step updates the following deliverables:

- **Facility Acquisition Plan** - see section 5.4.15.11 DD15.11 - Facility Acquisition Plan on page 227

This step creates the following deliverables:

- **Facility Requirements Assessment** - see section 5.4.15.12 DD15.12 - Facility Requirements Assessment on page 227
- **Implementation Team Assignment List** - see section 6.3.7.3 CD7.3 - Implementation Team Assignment List on page 258
- **System Construction Assignment List** - see section 5.4.21.4 DD21.4 - System Construction Assignment List on page 236

12.2.1.3 RADRC1.3 - Complete Test Strategy

A strategy for testing the system during development was prepared in the User Design stage, to provide a basis for estimating the cost of testing the system. This strategy is expanded into sufficient detail to guide test activities while the system is built.

A framework for testing the system is developed that defines the various test phases and describes general test scenarios. This framework ensures the completeness of the testing and minimises the effort to develop test data.

This step updates the following deliverables:

- **Testing Strategy**- see section 5.4.20.8 DD20.8 - Testing Strategy on page 235

12.2.2 RADRC2 - Construct the System

12.2.2.1 RADRC2.1 - Construct Database

The database definition statements are generated automatically or manually. The statements are processed by a database management systems utility to create the database objects and allocate space for them. The execution data used to create the initial production database is prepared by adjusting the data used for system testing.

This step creates the following deliverables:

- **Procedure Source Code** - see section 6.3.3.7 CD3.7 - Procedure Source Code on page 257

This step updates the following deliverables:

- **Production Data Structure** - see section 6.3.3.2 CD3.2 - Production Data Structure on page 257

12.2.2.2 RADRC2.2 - Define Procedure Actions

Examine the process to procedure mapping

When the procedures were defined in the User Design stage, the relationship between the procedure and process was identified. In the case of a one-to-one mapping the business logic expressed in the process action diagram is included directly in the procedure action diagram. In all other cases the mapping must be examined to determine how the logic is split over the procedures.

Add dialog actions

The next step is to add dialog actions. The action of a procedure may be different under different incoming flows. The decision as to which outgoing flow is to be followed is determined by setting an exit state within the procedure action diagram. Details of the flow are examined in order to control conditional actions. All the flows leaving the procedure are examined and a note made of the exit states. The conditions under which each flow is taken is determined and actions included in the procedure action diagram type set exit states.

Add layout and exception actions

Layout actions are then added. The contents of a layout are examined to determine conditional actions required based on the contents.

Lastly exception actions are added, including validation of input. An action or action group must be associated with each exception to ensure that all values are handled in a logical and consistent manner. `

This step creates the following deliverables:

- **Procedure Action Diagram** - see section 5.4.1.3 DD1.3 - Procedure Action Diagram on page 209

This step updates the following deliverables:

- **Screen/Report Layout** - see section 5.4.8.1 DD8.1 - Screen/Report Layout on page 217

12.2.2.3 RADRC2.3 - Generate Code

Procedure action diagrams or groups of procedure action diagrams are packaged together prior to code generation. As soon as the code has been generated and converted to executable form, the next subtask is performed.

This step creates the following deliverables:

- **Procedure Source Code** - see section 6.3.3.7 CD3.7 - Procedure Source Code on page 257
- **Executable Program Module** - see section 6.3.3.1 CD3.1 - Executable Program Module on page 257

This step updates the following deliverables:

- **Reusable Object Library** - see section 5.4.15.9 DD15.9 - Reusable Object Library on page 226

12.2.2.4 RADRC2.4 - Perform Unit Tests

The SWAT team member will immediately test the procedure once code generation has been completed. The CASE tool should have testing facilities to enable the unit testing to be performed. When errors are detected they will be immediately corrected by making the appropriate changes to the procedure action diagram or dialog flow diagram and then regenerating the code.

Additionally, usability testing should be performed by the users to ensure that the transactions under development are as easy as possible to use.

This step creates the following deliverables:

- **Test Results** - see section 6.3.4.2 CD4.2 - Test Results on page 258

This step updates the following deliverables:

- **Executable Program Module** - see section 6.3.3.1 CD3.1 - Executable Program Module on page 257
- **Procedure Source Code** - see section 6.3.3.7 CD3.7 - Procedure Source Code on page 257
- **Production Data Structure** - see section 6.3.3.2 CD3.2 - Production Data Structure on page 257
- **Reusable Object Library** - see section 5.4.15.9 DD15.9 - Reusable Object Library on page 226

12.2.2.5 RADRC2.5 - Manage Coordinating Model

If the system is being constructed by multiple SWAT teams, it is necessary to manage the coordinating model. Changes that must be made to the external

definitions of data have to be approved by the project manager. The changes are applied to the coordinating model by the repository manager who communicates the change to all the SWAT teams.

Shared Objects

Management of the coordination model is concerned primarily with identifying and tracking the definition and use of objects shared by two or more parallel development subsystems. The repository manager keeps track of the objects used by each SWAT team in order to determine what groups will be affected by changes to those objects. After changes have been approved by the project manager, the repository manager incorporates those objects into the coordinating model as shared objects and provides revised model subsets to each of the SWAT teams affected by the change.

Locking Changes

A change locking facility prevents simultaneous revisions to shared objects. This does not mean that whichever team gets the object first has the undisputed right to change that object. Changes that affect the other team should be agreed to by all affected teams before the changes are made.

Reusability of Objects

It is not necessary to manage objects that are used only within one subsystem. However, a related aspect of model management involves analysing objects to determine their reusability in the parallel subsystems and in subsequent projects. Thus, it is possible that additional shared objects will be identified in the course of analysing reusability.

This step updates the following deliverables:

- **Reusable Object Library** - see section 5.4.15.9 DD15.9 - Reusable Object Library on page 226

12.2.3 RADRC3 - Generate Test Data and System Documents

12.2.3.1 RADRC3.1 - Develop Operations Instructions

Systems are often executed on data processing hardware that is controlled by someone other than the users. In a mainframe or enterprise server environment an operations staff usually has responsibility for managing the operation of the systems running on the central processor. Similar responsibilities, often less organised, exist in departmental computer environments.

Operations Responsibilities

Operations personnel need instructions to perform their responsibilities related to the system. These responsibilities typically include:

- System start-up
- System shut-down
- Routine system data backup
- Communications management
- Error response
- Coordination with other systems
- Restart/recovery after system failure
- Security/access control

Instructions for Elementary Processing Support Functions

In addition to the written procedures for users, personnel responsible for the technical operation of the application must also be given definitive instructions defining how to perform certain elementary processing support functions. These instructions are also developed in this task.

This step creates the following deliverables:

- **Operations Manual** - see section 6.3.3.4 CD3.4 - Operations Manual on page 257

12.2.3.2 RADRC3.2 - Develop Documentation

Two categories of documentation are developed: technical and end-user.

Technical Documentation

Technical documentation resides in the information repository and is created as part of the analysis and design process.

User Documentation

User documentation should be on-line to the extent possible. There should be little or no external paperwork documentation. A particularly effective way of creating on-line documentation is with hyper-document software.

The user documentation team prepares the necessary documentation to describe the use of the system and any associated manual procedures. The documentation also includes HELP screens. Wherever possible, reusable document structures and templates are used to speed up the creation of documentation and to aid user familiarity. The need for help can be minimised if application screens are laid out well and include clear instructions.

Procedures Manual

Develop a procedures manual for any portion of the system procedures which involve manual, not automated, processing steps.

This step creates the following deliverables:

- **User Documentation** - see section 6.3.6.1 CD6.1 - User Documentation on page 258

This step updates the following deliverables:

- **Operations Manual** - see section 6.3.3.4 CD3.4 - Operations Manual on page 257

12.2.3.3 RADRC3.3 - Develop System Test Plan

The detailed test plan is developed. The test strategy is reviewed and a set of test scenarios and test conditions are developed. The test scenarios and conditions are determined by reviewing the system requirements and design specifications. The test scenarios documented in the plan must be designed to demonstrate that all documented functional, performance and interface requirements of the new system are satisfied. The test conditions specified in the plan are the basis for the creation of the test data.

This step updates the following deliverables:

- **Test Plan** - see section 5.4.20.9 DD20.9 - Test Plan on page 235

12.2.3.4 RADRC3.4 - Develop System Test Data

The test conditions described in the test plan are used to develop the test cases consisting of input data and anticipated results. A comparison of anticipated and actual test results prove whether or not the actual result is correct or identify if an error condition exists. Error conditions will normally require individual test cases to be developed. Occasionally more than one test case may be required to fully prove that the test condition operates correctly. Whenever possible, it is practical to verify several non-error test conditions in a single test case.

Purpose of the test data

The test data is used to provide the means for monitoring the operations of the load modules, interfaces, database and system procedures so that if the system successfully processes the test data provided, then the system is ready to go into full production. In addition, this task produces the documentation that accurately describes the operation of the new system. This documentation includes instructions describing the performance of any manual system procedures and the technical operation of the system software.

Test Case Documentation

For each system test case the following documentation should be prepared:

- Test condition(s) that the test case is verifying
- Initial database state
- Field values for input screens, documents and records
- Expected field values for output screens, reports and records
- Resulting database state

Test Cycle Documentation

In practice it is simpler to define the initial database state and the resulting database state for the entire test cycle, dm for each test case.

This step creates the following deliverables:

- **Test Cases and Test** Data - see section 6.3.4.1 CD4.1 - Test Cases and Test Data on page 258

This step updates the following deliverables:

- **Test Plan** - see section 5.4.20.9 DD20.9 - Test Plan on page 235

12.2.3.5 RADRC3.5 - Load System Test Database and Files

The test database contents are created and used for both system and acceptance testing. System testing should be conducted using entirely separate libraries from integration testing. Conditions for system testing should be the same as those for the production system (as far as it is practical).

The database definition control blocks and any record layout definitions are placed in the system test libraries. Next, the system test database is created using the procedure that will be used to create the initial production database. If the new database is an extension of an existing one, then the existing database must be unloaded and reloaded. In this case, a test database which is structured like the current production database is unloaded and then reloaded with the new database structure.

The system test database is populated using the programs that will be used to populate the production databases, such as conversion programs and on-line and batch update programs.

This step updates the following deliverables:

- **Test Cases and Test Data** - see section 6.3.4.1 CD4.1 - Test Cases and Test Data on page 258

12.2.4 RADRC4 - Prepare for Implementation

12.2.4.1 RADRC4.1 - Finalise Implementation Strategy

A general strategy for implementing the new application software in place of existing applications and procedures was previously developed through discussions with users and review of conversion requirements. This strategy is reviewed once more with users before proceeding further with implementation planning. Any new considerations based on further design progress are discussed. The strategy for implementation is finalised.

This step updates the following deliverables:

- **Implementation Plan** - see section 5.4.2.4 DD2.4 - Implementation Plan on page 211

12.2.4.2 RADRC4.2 - Develop Training Plan

A plan for conducting the user training necessary to support operation of the application in production is developed.

The training plan identifies the types of training that must be provided and the criteria for selecting the personnel who will receive it. A training curriculum is prepared consisting of a outline and the topics to be covered in the training session(s). The materials necessary to support the training are also described as are the qualifications of the instructor who will present each topic.

- **Training Plan** - see section 6.3.5.2 CD5.2 - Training Plan on page 258

12.2.4.3 RADRC4.3 - Finalise Contingency Plan for Implementation Failure

A plan for recovering from the failure of the new application after it is placed into production operation should be developed for all critical applications. This recovery usually involves restoring the systems and procedures previously in place back to their original status, prior to the initiation of the implementation effort.

The need for a contingency plan and the required level of detail to be included in it depend greatly on the circumstances surrounding the application being converted. If the application is simple and its operation is not critical to the functioning of the organisation, then a contingency plan is probably unnecessary. If the application is complex and has a significant probability of failure, or if the application is essential to the day-to-day functioning of the organisation, then a contingency plan is recommended.

This step updates the following deliverables:

- **Implementation Plan** - see section 5.4.2.4 DD2.4 - Implementation Plan on page 211

12.2.4.4 RADRC4.4 - Finalise Implementation Workplan Steps

A detailed work plan for performing the implementation is finalised. This work plan lists the specific steps necessary to place the application in production status. The work plan reflects the general implementation strategy defined previously.

The work plan is based on the preliminary version developed in the User Design stage. Estimates of the resources required to complete each step are listed on the work plan. Planned start and completion dates for each work step are also listed. A milestone chart identifying key events in the overall process is presented. The detailed implementation worksteps may also be recorded in other forms.

This step updates the following deliverables:

- **Implementation Plan** - see section 5.4.2.4 DD2.4 - Implementation Plan on page 211

12.2.4.5 RADRC4.5 - Develop Training Materials

Materials necessary to support training of the end-users of the application during the Implementation stage are prepared. This activity generally involves development of presentation slides, instruction manuals, wall charts, etc. that will be used in the training sessions. It also involves obtaining copies of relevant documents, particularly sections of the user documentation, for distribution to the users during the training sessions.

This step creates the following deliverables:

- **Training Material** - see section 6.3.5.1 CD5.1 - Training Material on page 258

12.2.4.6 RADRC4.6 - Resolve Organisational Issues

During the Requirements Planning and User Design stages of the application development effort various organisational issues were identified and steps were initiated to resolve them. During the Rapid Construction stage any new organisational issues resulting from design changes or implementation arrangements are identified and resolved. Resolve any open organisational issues that must be addressed prior to Implementation.

This step updates the following deliverables:

- **Project Problems and Issues List** - see section 4.3.1.1 AD1.1 - Project Problems and Issues List on page 122

12.2.4.7 RADRC4.7 - Prepare Implementation Procedures

Detailed procedures are required to support the execution of the implementation work plan. These procedures define the detailed steps necessary to perform the various tasks that user personnel are asked to execute. Many of these procedures involve data preparation tasks that must be performed manually. Other procedures cover operational tasks, such as reconfiguration or relocation of equipment, establishment of communications arrangements, and backup of critical data. Procedures necessary to implement the contingency plan for restoration of the old systems and procedures if the new application fails are also developed.

Implementation procedures are usually developed in printed form. These procedures must be explained to the users who are expected to invoke them.

This step creates the following deliverables:

- **Implementation Procedure** - see section 6.3.7.1 CD7.1 - Implementation Procedure on page 258

12.2.4.8 RADRC4.8 - Develop Data Conversion Routines

Implementation of many applications involves automated processing of existing data in a format suitable for loading into the application's data structures. This automated processing often involves development of routines designed for this specific purpose.

The data conversion routines are designed, coded and tested.

This step creates the following deliverables:

- **Data Conversion Routine** - see section 6.3.7.2 CD7.2 - Data Conversion Routine on page 258

12.2.4.9 RADRC4.9 - Test Implementation Arrangements

A test of the implementation plan should be conducted prior to initiating the Implementation stage. In this test all steps necessary to perform the implementation are performed in the same sequence as will occur in the actual implementation event.

The purpose of this test is to uncover problems before the actual implementation process is initiated. Users who will actually be performing the implementation tasks should participate in the test- this helps to train them in the process and

provide an opportunity to clear up any misunderstandings that they might have. All computer routines are executed to verify their correct operation. Attention should be paid to the time required to perform each implementation task and the planned schedule for the implementation verified.

This step updates the following deliverables:

- **Implementation Plan** - see section 5.4.2.4 DD2.4 - Implementation Plan on page 211
- **Implementation Procedure** - see section 6.3.7.1 CD7.1 - Implementation Procedure on page 258

12.2.5 RADRC5 - Verify System Construction

12.2.5.1 RADRC5.1 - Perform Integration Tests

The integration tests are performed according to the integration test plan as soon as the specified group of modules or transactions have been unit tested. The test cycle is executed and the results are compared with the anticipated results. Any discrepancies between the results must be investigated. Correction of the error may require modifications to the procedure action diagrams, dialog flow diagram, screen/report layouts or to the database definition. With the exception of changes to the database which are carried out by the database administrator, the remainder are made by the SWAT team member and the cycle of design, generate, test is performed once more.

This step creates the following deliverables:

- **Test Results** - see section 6.3.4.2 CD4.2 - Test Results on page 258
- **Working Tested System** - see section 6.3.4.3 CD4.2 - Working Tested System on page 258

This step updates the following deliverables:

- **Executable Program Module** - see section 6.3.3.1 CD3.1 - Executable Program Module on page 257
- **Production Data Structure** - see section 6.3.3.2 CD3.2 - Production Data Structure on page 257
- **Dialog Flow Diagram** - see section 5.4.7.2 DD7.2 - Dialog Flow Diagram on page 217
- **Procedure Action Diagram** - see section 5.4.1.3 DD1.3 - Procedure Action Diagram on page 209
- **Screen/Report Layout** - see section 5.4.8.1 DD8.1 - Screen/Report Layout on page 217
- **Data Structure Diagram** - see section 5.4.5.9 DD5.9 - Data Structure Diagram on page 217
- **Procedure Source Code** - see section 6.3.3.7 CD3.7 - Procedure Source Code on page 257

12.2.5.2 RADRC5.2 - Review Integration Testing

The results of the integration testing are inspected to ensure it is complete and has been properly conducted. Integration tests are not repeated during system testing.

12.2.5.3 RADRC5.3 - Review System Test Plans

A review of the system test plan is performed prior to commencing system testing. This ensures that any changes made to the system during the Construct the System task (as a result of user requirement changes) and during integration testing are incorporated into the system test plan.

This step updates the following deliverables:

- **System Test Plan** - see section 5.4.20.1 DD20.1 - System Test Plan on page 231
- **System Test Cycle Dependency Diagram** - see section 5.4.20.2 DD20.2 - System Test Cycle Dependency Diagram on page 234

12.2.5.4 RADRC5.4 - Perform System Tests

The system tests are performed according to the test cycles specified in the system test plan. For each test case the output is compared to the expected results. All differences are investigated by the construction assistance team and the SWAT team, and if necessary, changes are made to the system and the relative part or parts of it are regenerated and tested again. This process is repeated until there are no discrepancies between actual system test output and expected output, or until any remaining discrepancies are understood and approved by the users. All test results, including all iterations and the reasons for them, are fully documented. Additionally, usability testing is performed to make sure that the system meets the usability criteria required by the users.

This step creates the following deliverables:

- **Test Results** - see section 6.3.4.2 CD4.2 - Test Results on page 258

This step updates the following deliverables:

- **Working Tested System** - see section 6.3.4.3 CD4.2 - Working Tested System on page 258
- **System Test Plan** - see section 5.4.20.1 DD20.1 - System Test Plan on page 231

12.2.5.5 RADRC5.5 - Devise Acceptance Tests

A series of test conditions are tested for each test cycle during acceptance testing. For each test condition, a test case is prepared consisting of input data and expected results that will prove whether or not the test condition is correctly executed by the system.

This step updates the following deliverables:

- **Acceptance Test Cycle Dependency Diagram**- see section 5.4.20.4 DD20.4 - Acceptance Test Cycle Dependency Diagram on page 234

This step creates the following deliverables:

- **Acceptance Test Requirement** - see section 5.4.20.3 DD20.3 - Acceptance Test Requirement on page 234

12.2.5.6 RADRC5.6 - Perform the Acceptance Test

Acceptance testing is performed by parties designated by the business project manager and actual results are checked against the expected results. Any errors will be corrected by the SWAT team. It is quite likely that errors detected will be of an interpretive nature and a decision must be made at this point as to whether the changes can be effected in the timebox, or whether the changes will be made for the next release of the system.

This step creates the following deliverables:

- **Accepted System** - see section 6.3.4.4 CD4.3 - Accepted System on page 258

13. Appendix 1 – Terms

This appendix lists terms used throughout this document.

13.1 A

A	See Auditor
AC	See Application Development Methodology Council
Acceptance Testing	See Testing.
Accountable	Subject to giving an accounting for their actions (answerable).
Activity	Generic term referring to any action or procedure performed by a company or business.
ADM	See Application Development Methodology.
Analysis Stage	The stage of the ADM in which is performed a high level and broad analysis of the processes occurring within a business activity and the data required to support that activity. The analysis is at higher level and covers a broader scope than the detailed analysis performed in the design stage. The analysis stage follows the project initiation stage and precedes the design stage.
Application	The set of business processes that capture and maintain business data and provide business information. These processes may be either manual or automated.
Application Development Methodology	A methodology of tasks, deliverables and techniques.
Approval Responsibility	The role of the party who must formally approve the contents of the deliverable before the deliverable can become baselined and the development process can proceed. Accountability for obtaining the approval of the designated party rests with the party having primary responsibility for that deliverable. Once approved, any further changes require the use of the change management process. Issues not resolved are to be escalated through the designated approval party's management chain. Unresolved issues arising from a deliverable will stop the development process.
Attribute Type	An occurrence of an attribute type.
Attribute Type	A fact, having a value, about an entity type.
Auditor	An organisation unit or individual that performs a control function for the organisation.

13.2 B

BAA	See Business Area Analysis.
Baseline	The content of a deliverable has been reviewed and accepted by approving parties. It cannot be changed without appropriate management approval and acceptance by the approving parties.
BCA	See Business Contract Administrator
BE	See Business Expert
BPM	See Business Project Manager
BPR	See Business Process Re-Engineering.
BSD	See Business System Design.
Bug	A software error which is characterised by a deviation or omission from specified functionality.
Business Activity	A generic term referring to an action performed or executed by the business and which produces output meaningful to the business. .
Business Area	A set of related business activities and the data entities that those activities create or update.
	A business area is usually segmented into multiple design areas when transitioning from the analysis stage to the design stage.
Business Area Analysis	The stage of the Application Development Methodology that defines the business' requirements.
Business Area Analysis Project	A project analysing the business processes and events within a proscribed set of business activities and the data created or updated within that set of business activities.
Business Area Model	The set of models reflecting business activities, data, business locations and organisation units that represent a business area.
Business Contract Administrator	A party or organisation unit that provides legal consultation, manages and/or enforces an organisation contractual obligation.
Business Data	A set of facts that are meaningful to the business.
Business Event	An instance (or happening) in the life of the business when an external party interacts with the business and as a result of that interaction causes a change in the current state of the business. Anticipated business events result in business response(s). One or more business activities are executed to generate that business response.
Business Event Model	A series of models pertinent to a business area that depicts its business events and the set of business processes producing the responses to those events.
Business Expert	An individual with in-depth knowledge of an organisation unit’s overall business processes and data and who is able to identify the business needs of a particular organisational unit.
Business Function	A high level summary of ongoing processes that are taking place to perpetuate the business. Each business function is decomposed into multiple lower level business functions or

	business activities.
Business Location	A geographic area, county, city, town or site where the organisation does business.
Business Model	A technology independent business area model.
Business Process Re-Engineering	A project wherein the current methods of conducting a set of business activities is evaluated and revised with the intent of increasing the efficiency of the business activities.
Business Project Manager	A person representing the business who has decision-making authority for the user community. They are responsible for assignment of users to the project team and to develop user guides, office procedures and user training. Further, they identify users to be interviewed, users to participate in review walkthroughs and inspections. They also will identify users who will participate in testing, development of test scripts, test cases and test data. Additionally, they will resolve user conflicts, be a focal point for change control, prioritise user requests and keep the user community informed of project status.
Business Organisation	See **Organisation Unit**.
Business Response	The business deliverable or output of a set of business activities that satisfies the business event that triggered the set of business activities.
Business System Design	The phase of the design stage of the Application Development Methodology addressing the external interfaces of system being designed.

13.3 C

CCM	See **Code Configuration Manager.**
Code Configuration Manager	A person or organisation unit responsible for maintaining the library of code "objects" for a system and migrating/installing/configuring new and/or revised versions to appropriate environments.
Code Library	A repository of executable modules and/or objects pertaining to the University's systems and databases.
Computer File	Business data stored in/on a computer readable media.
Computer System	An automated set of tasks that performs one and only one business activity or some implementable portion of a business activity.
Concur Responsibility	The role wherein a party must formally endorse or concur with the approval of the deliverable in order for the deliverable to be baselined and the development process to proceed. Responsibility for gaining the concurrence of the designated concurrence rests with the party having primary responsibility for that deliverable. Verification of the fulfilment of a deliverable may be either verbal or in writing. It is the party having primary

	responsibility for the deliverable to determine which mode is appropriate.
Contingency Plan	A set of directives and procedures to be followed if implementation is not successful. It specifically refers to the contingency during the test to production turnover, not the contingencies that might occur during an operational situation.
Control Function	Any organisation unit performing a business control function such as auditing, production control or quality assurance.
Construction Project	A set of processes and tasks that transforms an information system design into an implementable module.
Construction Stage	The stage of the ADM wherein is created the information system, the files and databases which it requires, the implementation plans and the operational documentation and in which the operational training is performed. The construction stage follows the design stage and precedes the implementation stage.
Consultation Responsibility	The role of the party that may be required to provide assistance to complete the deliverable.
Conversion Plan	A set of directives and procedures that addresses the change of a process or set of data from one mode or format to another.
Critical Success Factor	Critical success factors are those few key areas in which satisfactory results will ensure competitive performance for the organisation. CSFs support the attainment of organisation Goals.
CSF	See **Critical Success Factor**.

13.4 D

DAD	See **Data Administrator**.
Data	Representation of concepts, facts or instructions in a formalised manner. Data are the raw material of information.
Data Acquisition System	An information system that collects and validates data and stores that data in a sharable data structure.
Data Administrator	An organisation unit or individual which develops and maintains an unambiguous and consistent set of definitions of data, activities and their relationships to facilitate common understanding and to provide a consistent framework for information systems within the enterprise.
Data Application System	An information system that extracts data from a sharable data structure, manipulates it in a prescribed manner and produces information in any form.
Data Capture	The activity of collecting data from an information source and storing that data in a sharable data structure to enable its access at a later time.
Data Conversion	The activity of changing the structure, format, location or media of data. The activity is frequently associated with the

	transfer of data from one or more existing operational systems to a new information system during its implementation.
Data Entity	A class of persons, places, things, events or concepts about which the organisation wishes to keep data.
Data Life Cycle	The continuum of the planning, acquisition, employment, maintenance and disposition of data.
Data Management Tools	Software and hardware resources and techniques for capturing, storing and controlling data as well as for delivering information.
Data Model	A representation of the business data captured by and required to support the organisation business processes.
Data Steward	An organisation unit or individual that is responsible for the accuracy, timeliness, access and security of a segment of the organisation's business data.
Data Stewardship	The role of establishing the definition of a data object, establishing data validation rules and establishing data access authority.
Data Validation	The activity of validating data as to it reasonableness.
Database	A set of logically structured data that enables direct access to individual members using multiple criteria.
Database Administrator	The organisation unit or individual responsible for designing and maintaining the physical database(s) required by an information technology project and develops database rollback and recovery strategies.
DBA	See **Database Administrator**.
DC	See **Documentation Specialist**.
Deliverable	The products of the Application Development Methodology tasks performed. Deliverables represent tangible products of the system development process.
Design Area	The scope of a project whose objective is to create the design of an information system. The design area is usually formed as a subset of a business area. That is, a business area is usually segmented into multiple design areas when transitioning from the analysis stage to the design stage.
Design Stage	The stage of the ADM in which the design of the information system, the files and databases that it requires, implementation and training plans and information system documentation is formulated. This stage typically occurs after the analysis stage and prior to the construction stage.
Design Project	A work effort whose objective is to produce the design of an information system.
Detailed Business Area Analysis	The second (and more detailed) phase of the Business Area Analysis stage of the ADM.
Detailed Business Model	An outline business model which also incorporates one or more of the following models: entity state transition diagrams, process logic diagrams, process involves attribute

	associations, process action diagrams, attribute type domains, attribute type properties, attribute type ranges and physical data flow diagrams.
Disaster Recovery Plan	A set of directives and procedures which identifies the actions to be taken in the event of an unplanned disruption to the automated and manual operations (processes) of the business.
Documentation Specialist	A person who writes, reviews, maintains and releases all product documentation; including user guides, reference manuals, detailed system specifications; and designs and defines documentation standards.
DS	See **Data Steward**.

13.5 E

EM	See Enterprise Model
EMS	See Enterprise Modelling Stage.
Emergency	A situation under which the end user is unable to carry on their mission critical business.
Emergency Fix	A software modification that returns a system to its original working condition.
End User	They provide the primary input to and receive the output from an information system.
Enhancement	A modification to an existing system that is intended to provide an additional functionality or improve quality.
Enterprise Model	A model of the enterprise's business activities, data, business locations, organisation units, existing systems and long term directions.
Enterprise Modelling Stage	The stage of the ADM in which the enterprise defines its strategic plan for those information system(s) that it might develop.
Entity	An occurrence of an entity type.
Entity Type	A class of persons, places, things, events or concepts of importance to the business.
Entry Point	The point of entry in a set of program code statements.
Environmental Software	Computer software other than that which directly captures and maintains business data and provides the business information. It is that software which establishes and maintains the software envelope in which the application software can operate. Common components of environmental software include operating systems, language compilers and utilities.
Environmental Software Support	An organisation unit or individual that provides and maintains that software which establishes and maintains the software envelope in which the application software operates.
EU	See End User
External Specifications	Content and format definitions of information vehicles associated with an information system that are visible and

	directly usable by human beings. Examples include PC screens, forms and reports. Specifically excluded are the contents of computer memory or storage devices.

13.6 F

Field	The smallest addressable portion of a computer file defined to contain data.
File	A set of logically structured data.

13.7 I

IM	See Implementable Module.
Implementable Module	A set of manual and/or automated procedures that operate as a unit to produce business information and/or perform a meaningful business task. An automated implementable module consists of code statements. An implementable module represents a business activity or subset thereof and interfaces or integrates with other implementable modules.
Implementation	To move all components of a system from a test environment to a production environment.
Implementation Stage	The stage of the ADM in which the products created in the construction stage are implemented for operational use.
Information	Data in a meaningful association or context.
Information Library	A repository of information about the data, activities and technology components of the organisation's systems and databases.
Information Model	A representation of the business information produced by and required to support the organisation’s business processes.
Information Resource Management	A set of management approaches that enable organisations to manage information as a shared resource.
Information Resources Planner	An organisation unit or individual that plans the information systems technology environment and recommends the business areas and information systems needed by the organisation.
Information Resources Strategy	A long-term direction, specifically related to information resources, which supports the organisation strategies.
Information System	A grouping of one or more implementable modules that comprises an information system.
Information System	Architecture A structure composed of business data and business processes that identifies individual business systems that can be developed with the knowledge that they can be readily integrated and can share data.
Information	A plan showing the allocation of information systems

System Plan	resources over the forthcoming three to five year time horizon. It describes the development and supporting projects and integrates the priorities of organisation management and the strategies of Information Resources.
Information System Project	A work effort whose objective is to produce an information system.
Information Technology	This refers to the application or utilisation of technology in a system; not an organisation unit.
Information Technology Steering Committee	A decision making body of individuals accountable for the selection and prioritisation of the organisation's information technology projects.
Information User	The individual or group which utilises information to perform their assigned business tasks.
Infrastructure Project	Projects directed at implementing technology, methodology, training, procedures or access systems over a broad section of the organisation and are, therefore, beyond the scope of a single business area.
Input Responsibility	The role of the party that must provide necessary information (either written or verbal) to complete the deliverable.
Integration Testing	See **Testing**.
IR	See **Information Resources**.
IRP	See **Information Resources Planner**.
IS	See **Information System**.
IT	See **Information Technology**.
Iteration	The process of repeating a set of activities till the products of the activities are satisfactory. The set of repeated activities can be limited or substantial in scope.
ITS	See **ITS**.

13.8 J

JAD	See **Joint Application Development**.
Joint Application Development	The use of facilitated sessions involving subject matter experts to gather information and make business decisions concerning an information system project.
Joint Requirements Planning	The use of facilitated sessions involving subject matter experts to plan future sessions concerning an information system project.
JRP	See **Joint Requirements Planning**.

13.9 L

Legal and Contracts	An organisation unit that either provides legal consultation or manages the establishment of organisation contractual

	obligations.
Level of Effort	A quantity of human time associated with the performance of a specified activity.
Library	A repository of information or program code relating to information systems and databases.
Linkage	A connection between two related records by which records may be accessed, but of which modules need have no knowledge.
Location	See **Business Location**.
Logical Business Activity	See **Business Activity**.
Logical Business Data	Conceptual representation of business data.
Logical Data Structure	A representation of business data logically structured in the data definition language and constraints of a particular database management system.

13.10 M

Mechanism	The implementation in technology of a business process.
Migration Plan	A set of directives and procedures providing for the movement or duplication of automated processes and/or data from one location or place to another without changing its form or function. Sometimes referred to as a Turnover Plan.
MM	See **Model Manager**.
Model Manager	A person or organisation responsible for managing the organisation's library of models.

13.11 N

Needs Analysis	Preliminary definition of the business situation needing correcting or resolving.
Needs Analysis Report	A collection of information about the nature and extent of a perceived business need. The information includes the description and objectives of the perceived business need as well as identification of the business activities, data entities, locations and organisation units involved in the need.

13.12 O

O	See **Operations**.
OBAA	See **Outline Business Area Analysis**.
Office of Record	The organisation department that is custodian for the organisation's official version of a set of information.

Operational Control Procedures	Procedures which are intended to ensure that a system is being operated in an expected and approved manner.
Operational System	An information system that processes business data and creates information on which the organisation's business decisions are made.
Operations	An organisation unit responsible for operating an information system, communications network and for hardware and communications facilities maintenance.
Organisation Unit	A person or group of people within the organisation structure assigned to perform or manage specific business processes.
Outline Business Area Analysis	The initial (and more generalised) phase of the Business Area Analysis stage of the ADM.
Outline Business Model	The set of models representing a business area which typically includes a process decomposition diagram, entity-relationship diagram, business event model, either logical data flow diagrams or process dependency diagrams, process involves entity type association and design areas.

13.13 P

PD	See **Project Director**.
Physical Data Structure	The framework or skeleton of a data storage facility that is not populated with data.
Physical Database	A data storage facility populated with business data. It is usually associate with a computer.
Pilot	A pilot is a system that is not yet fully operational. Whereas a prototype is used when the functions and detailed design of a system are not fully understood, in a pilot system the functions and design are thought to be understood, but the system is cut over in limited form so that experience can be gained with it before the full system is implemented. As a result of pilot operation, modifications are usually made before the system is fully deployed.
Policy	A stated course or method of organisation action to guide and determine present and future decisions.
Post Implementation Assessment Stage	The stage of the ADM that follows the Implementation Stage, supports the implementable module during its immediate post implementation period and responds to emergencies affecting the operational system during this period.
Post Implementation Assessment	A project normally conducted 90 to 180 days after the completion of a system development project. The purpose of which is to review the effectiveness of the system development process employed, to determine whether the operational system achieved the system development project's intended benefits and whether the operational system is efficient.

Primary Responsibility	The role of the party that has primary responsibility for consolidating all information and producing and publishing the deliverable. It includes coordination of all activities associated with the development and approval of the deliverable.
Principle	A comprehensive and fundamental doctrine.
Problem Description	Section of the project initiation request form used to describe the business situation needing correcting or resolving.
Procedure	A technology dependent activity normally expressed as a series of action steps followed in a definite order.
Process	A technology independent business activity.
Process Dependency Diagram	A model relating the dependencies, conditions, plurality and optionality among the processes within a theme.
Process Specifications	The definitions of an information system's information vehicles and business activities.
Processing Operations	An organisation unit responsible for operating an information system as well as for maintaining its operability.
Program Specifications	Structured guide identifying the actions to be performed or accomplished by a program.
Project	Defined work effort to accomplish a specific task.
Project Boundary	The outer limits of a project's scope.
Project Director	A person who resolves issues between the Business Project Manager and Technical Project Manager. They have decision authority for budget issues and resource issues beyond the scope of the project managers. Further, they act as a liaison between the project and the rest of the organisation community and resolve political issues beyond the scope of the project managers. The role of mediation is assumed in each task where it is not specified.
Project Initiation Request	Request to create new business information or functionality, increase the cost effectiveness and efficiency of existing business activities or to correct a deficiency in an existing system, file or database.
Project Initiator	An individual or group responsible for initiating a project request who may or may not become the project sponsor.
Project Initiation Stage	The stage of the ADM in which individual implementable module construction projects, design projects and business area analysis projects are formulated and their pursuit approved.
Project Leader	An individual having primary responsibility for the welfare of a project.
Project Plan	A plan identifying the deliverables and activities to be performed in a project. It includes the project's resource requirements, estimated costs, timeline and project schedule.
Project Scope	The business activities, business data, organisation units and business locations that represent the subjects or territory to be addressed by a project.

Project Scope Document	A document that defines the objectives, scope, time-line and resources of the project.
Project Sponsor	An individual or group of individuals representing organisation faculty or administrative management that endorses the scope and objectives of a project as well as representing that project in organisation councils.
Project Steering Committee	A group of individuals having collective accountability for overseeing the welfare of a project.
Prototype	A prototype of the system will be produced to whatever level of detail is required. At a minimum, this will consist of an agreed-upon sequence of dialogs for the system, in whole or in part. This can be one of: • **Application Concept Prototype** An application concept prototype is a system developed to test and refine the underlying premise of an application. • **Application Package Prototype** This is a "canned" application tried out with a small group of users before its full-scale implementation. • **Calculations and Logic Prototype** These are examples of the computations for complex logic or calculations in an application. • **Data Entry Prototype** This is a shell routine or system to check the speed and accuracy of data entry, and to test validity and integrity checks. • **Data System Prototype** This is a small-scale database to test database contents and organisation. • **Dialog Prototype** This is a simulation of the interplay of user actions and the intended system response. • **Partial-System Prototype** A partial-system prototype emulates a facet of an application. • **Reporting System Prototype** This is a mock-up of reports provided to users. • **Screen Prototype** A screen prototype is a working model of the screens and their interaction that simulates the operation of the system once it is ultimately developed. The prototype lacks logic to process or display data. It can, however, demonstrate movements from one screen to another, simulating navigation through a real version of the system.

	• **System Prototype** A system prototype is an executable version of the system procedures. The prototype is based on the source code developed to implement the system, which is itself derived from the design specifications describing the procedure.

13.14 Q

QA	See **Quality Assurance/Quality Control**.
Quality Assurance/ Quality Control	A function responsible for defining standards, guidelines and maintaining a glossary of terms to facilitate common understanding among anyone who has to interact with the project. Additionally, Quality Assurance ensures all deliverables meet or exceed standards set for the deliverable. Standards are usually based on the intended use, target audience and impact on the community of the deliverable.

13.15 R

RAD	See **Rapid Application Development**.
Rapid Application Development	A method of developing information systems that places a heavy reliance on group facilitation and operational prototyping. This method is most frequently associated with the development of small, stand-alone systems.
Record	The collection of fields representing a single occurrence of a record type. A record is a single occurrence of a record type.
Record Type	The physical incarnation of an entity type.
Relationship	An occurrence of a relationship type.
Relationship Type	A business based association between entity types.
Requirements	Definition of the business processes, events and data included within the scope of the information system.
Resource	Something available for use in achieving a goal or objective.
Responsible	Liable to be called to account as the primary party.
Review Responsibility	The role of the party who is to review and comment on the deliverable. After the review is complete, issues not resolved to a reviewer's satisfaction, are to be escalated through the reviewer's management chain. Such action will not necessarily stop the development process.
Risk	Anything that might adversely affect the success of a project. Examples of risks include length of project, costs associated with projects, technological and functional uncertainties, lack of technical experience, potential reorganisation issues and input of changing business activities.

Risk Analysis	The process of identifying and planning for risks associated with a project.

13.16 S

S	See **Sponsor**.
SA	See **System Architect**.
SAC	See **Security Administrator**.
SAD	See **Systems Administrator**.
SC	See **Steering Committee**.
Scope	Defines and limits the area of interest and level of detail of a work activity. Scope is usually stated in measurable terms such as data entities, business activities, business locations and organisation units.
SD	See System Developer.
Security Administrator	The organisation unit(s) accountable for implementing the security and protection of the information and technology resources of the organisation.
SEM	See Strategic Enterprise Modelling.
Specification	The precise definition of something that exists or which is to be created.
Sponsor	An individual or group representing the organisation's faculty and/or administrative management that endorses the scope and goals of a project and represents that project in management councils. The sponsor has primary responsibility for marshalling the resources needed to fund the project.
Stakeholder	A party who has a vested interest in the outcome of a study or project.
Standard	A model or measure of quantity or quality established by authority, custom or general consent to guide present and future behaviour.
Steering Committee	Members of the academic and/or administrative staff having collective accountability for overseeing the welfare of an information technology project.
Strategy Model	Representation of the organisation goals and critical success factors.
Strategic Enterprise Modelling	A method of determining the enterprise's business functions, information needs, business locations, organisation units and business goals with the intent of establishing an information system architecture, data architecture and a development plan for enterprise's strategic information systems.
Stress Testing	See **Testing**.
Subject Area	A bounded set of data entities logically grouped by a common subject.
Support Responsibility	The role of the party that must be involved in the creation of the deliverable to ensure that its requirements and responsibilities are met. It may involve supplying significant

	input or providing a control function.
SWAT	Skilled Workers with Advanced Tools.
SWOT	Strengths, Weaknesses, Opportunities and Threats.
System	One or more processes or set of actions that may be either manual or automated.
System Activity	A technology dependent procedure.
System Architect	An individual responsible for the overall design of the system and for the successful integration of sub-systems. The System Architect provides technical leadership, plans and estimates, technical interfaces external to project; and manages overall test strategies. This individual also resolves technical issue/conflicts, and ensure adherence to standards.
System Developer	The individual(s) responsible for the modelling, analysis, programming, testing and/or implementation of an information system. (AKA: Modeller, Developer, Application Developer, or Programmer/Analyst)
System Life Cycle	The continuum of the planning for and creation, employment, maintenance and disposition of a system.
System Testing	See **Testing**.
Systems Administrator	An organisation unit or individual which provides and maintains that hardware and software which establishes and maintains the environment in which the application software operates.

13.17 T

TD	See **Technical Design**.
Technical Design	The phases of the design stage of the ADM which addresses technology issues and internal design issues.
Technical Project Manager	An individual having primary technical responsibility for the welfare of an information system project.
Technology	Refers to hardware, operating and other control software, methods and communications.
Technology Architecture	A representation of technology components (computing, data communications and environmental software) configured logically to support an enterprise.
Technology Support Contact	A user department's representative who has been given responsibility as the first point of contact for computing related problems or questions. This person has the responsibility to communicate all unsolved problems and unanswered questions to appropriate personnel for resolution. An individual who installs and maintains all software, hardware and the networks for end-clients.
Test Scenario	Specification of test data and the expected results of the test.
Testing	This consists of one or more of the following types of tests: • **Acceptance Testing** The evaluation of all facets of the

	implementable module by the end user's staff. The objective is to place the implementable module under situations and conditions unforeseen by the development team as well as to test the completeness and clarity of the operational training and documentation. • **Benchmark Testing** Tests performed to determine whether the time-related and volume-related business requirements of an entire system have been met. • **Integration Testing** Testing the integration of all components of an implementable module to ensure that all modules and all interfaces of new and existing modules are functioning properly. • **Regression Testing** Testing of a business system as a whole to ensure that recently made revisions have not adversely affected the functioning of the system as it operated prior to the introduction of the revisions. • **Stress Testing** Evaluation of the measured performance of all the software, hardware and communications facilities within an implementable module under maximum data volumes and extreme transaction frequencies as well as other demanding conditions. • **System Testing** Testing of a business system as a whole to ensure that the entire system including all interfaces to existing systems operates satisfactorily. • **Unit Testing** The testing of individual segments of code to ensure that errors can be detected.
Theme	A business event coupled with those business processes that it triggers along with the business response(s) produced by those processes that satisfies the event.
Timeline	A series of identified tasks, their dependencies and their durations. This information is not associated with the calendar.
TPM	See Technical Project Manager
Training Plan	A set of directives and procedures which identifies the organisational positions requiring training, the skills for which training is required and the training vehicle which will be employed to effect the training.
Training Specialist	An organisation unit or individual that prepares and/or teaches a full curriculum of technical courses to the user community. They manage all aspects of courseware development process and work with management to insure proper course rollout and translation/localisation of course material.

TS	See **Training Specialist**.
TSC	See **Technology Support Contact**.

13.18 U

Control Function Group	One or more persons which have restrictive influence or authority over a project's execution but which are not involved in providing information to the project or in performing the project.
EIM	See **Enterprise Information Model**.
Enterprise Functional Model	A representation of the business processes performed by the enterprise.
Goal	Specific targets that are intended to be reached in a specific time.
Enterprise Information Model	Collection of the enterprise Business and Data models.
Enterprise Strategy	A business direction, long-term in nature, toward which the enterprise intends to move.
User	A person or organisation that is supplying input to and/or receiving output from an information system.

13.19 W

Walkthrough	A guided assessment of the product of a development activity by a group of people with the intent of discovering errors or omissions in the product. Evidence of the walkthrough's findings is generally demonstrated with a participant "sign-off" process.

14. Appendix 2 - Entity Relationship Modelling

14.1 Introduction

This section describes the definition of entity types and their relationships. It defines entity and relationship concepts, their documentation and various guidelines and rules concerning the development of an entity relationship model.

Entity relationship modelling enables us to describe data and its inherent structure. The entity relationship model is represented as a diagram, known as the entity relationship diagram. This diagram is used to show our understanding of data and is used in later stages of the Application Development Methodology as a basis for data structure design.

14.2 Concepts

14.2.1 Activity

A business action or series of actions whose goal is the attainment of a business goal. An activity may be a function, process or procedure.

Two types of business activities are recognised in the methodology - functions and processes. Analysis of business functions and processes is concerned with understanding what needs to be done in an enterprise or what activities must be accomplished. It is not concerned with how these activities take place, as expressed by the technology used or the persons currently responsible.

Despite their similarities, it is useful to distinguish between functions and processes.

Inputs

The inputs to an activity may be tangible objects (e.g., materials and resources), information, or a mixture of the two. Sometimes the information describes the tangible objects (e.g., a delivery note describes the items delivered).

Work

Work is performed on inputs in order to produce the outputs. For example:

- Metal blank turned on a lathe to produce a candlestick
- Orders processed to produce a delivery schedule

Outputs

The outputs from an activity are also tangible objects, information, or a mixture of both. As with inputs, output information may describe material output.

14.2.2 Attribute Type

This is a descriptor of an entity type. Attribute types are the names of the things the business must store about entity types.

For example, the entity type EMPLOYEE may be described by attribute types Employee Name, Employee Number, Birth Date, Sex, Height and Eye Colour. An individual entity of the type EMPLOYEE may be described as J. Frost, 3142, 5 April 1959, Male, 72, Blue. This description consists of values of the above attributes.

Entity types, therefore, are described by attribute types, and entities are described by attribute values.

Attribute Type Source

The categorisation of an attribute type according to whether its values are basic, derived or designed.

Attribute Type Source Categories

It is useful for the verification and selection of attributes, and for later program and data design, to record the source category of an attribute. There are three possible source categories:

- **Basic Attribute**: An attribute type whose values cannot be calculated. Its values must be entered into the database (e.g., Employee Name, Birth Date)

- **Derived Attribute**: An attribute type whose values can be calculated from attribute values available in the database (e.g., Invoice Value, Account Balance, Pension Contribution)

- **Designed Attribute**: An attribute type whose value is calculated to serve a particular purpose and has no specific meaning to the enterprise (e.g., Abbreviated Customer Name, Employee Number)

Designed attributes types are not business objects and so are generally best avoided during Business Area Analysis. You may need them occasionally, when there is no other way of identifying the entities of an entity type (e.g., Employee Number), or when they are imposed by an external authority (e.g., Postal Code/Zip Code).

Fixed Attribute Type

An attribute whose values, once established for any given entity, remain unchanged for the life of that entity.

Composite Attribute Type

A named collection of attribute types of one entity type in which each group of values may be treated as if it were a single value.

An attribute type may consist of a meaningful collection of other attribute types. For example, the attributes Birth Month, Birth Day and Birth Year comprise the attribute type Birth Date, which is meaningful to the business and may be handled as a single attribute type. Birth Date is therefore described as a composite attribute type.

Optional Attribute Type

An attribute type that need not have a value for every entity of the type it describes.

When an entity need not have a value for a particular attribute, that attribute is optional. For example, the entity type LOCATION has an attribute of Telephone Number, but not every LOCATION has a telephone. Any such LOCATION will have no value for the Telephone Number attribute.

The optionality of an attribute may depend on a defined condition (e.g., BOOK has a Price only if it is still in print).

Optionality may depend on the existence of a relationship membership (e.g., EMPLOYEE has Pension Contribution only if EMPLOYEE participates in a PENSION SCHEME). The conditions and condition logic for attributes are basically the same as for relationships. They are a type of predicate condition and have predicate condition logic.

Transient Attribute Type

An attribute type whose values may change several times during the life of the entity that each value describes.

14.2.3 Cardinality

This is a business rule indicating the number of times a particular object or activity may occur

There are a number of different types of cardinality:

- **Cardinality of a Dependency** - The number of executions of each process that may occur before or after each execution of the other process.
- **Cardinality of a Subprocess** - The number of times a subprocess is executed during each execution of the process of which it forms a part.

- **Cardinality of a Relationship Type** - The number of pairings in which an entity may participate in the relationship membership.

There are three forms of cardinality:

1. **One-to-one (1:1):** For example, PERSON married to PERSON, where a PERSON may be married to one and only one other PERSON and vice versa.

2. **One-to-many (1:M):** For example, BOOK printed as EDITION, where each BOOK may be printed in several EDITIONs, but each EDITION is a printing of a single BOOK.

3. **Many-to-many (M:N)**: For example, AUTHOR writes BOOK, where any AUTHOR can write many BOOKs, and any BOOK can be written by many AUTHORs.

Given any grouping under a one-to-many relationship, there will be one entity paired with one or more entities of the same or another type. For the entities with a "one" membership (e.g., ORDER placed by one CUSTOMER), there is a single, identifiable, related entity (e.g., the customer who placed the order). For entities with a "many" membership, however, there are no easily identifiable related entities.

Consider the relationship "CUSTOMER may place one or more ORDERs." From the viewpoint of an order, the customer who placed it is easily distinguished because there is only one of them. It is not so simple to distinguish a particular ORDER for a given CUSTOMER, however. You need to use some method for distinguishing any one ORDER from all other ORDERs for that CUSTOMER, because there may be many orders. Similarly, if ORDER has a one-to-many relationship to ORDER ITEM, for each ORDER ITEM there is a single ORDER and a single CUSTOMER. For each CUSTOMER, however, there is no one uniquely identifiable ORDER or ORDER ITEM.

There can be a number of Cardinality restrictions:

- **Fixed Cardinality** - A type of cardinality condition in which a cardinality of a dependency, relationship or subprocess is always the same number. A relationship may have fixed cardinality (e.g., BUDGET consists of twelve PERIODs, or CHILD is born to two PARENTs). With fixed cardinality, the number of entities participating in each grouping is known for at least one of the entity types.

- **Cardinality Condition** - A type of predicate condition that places constraints on the cardinality of a dependency, relationship or subprocess.

14.2.4 Condition

This is a rule expressed in terms of predicates or constants that describes an aspect of the business that causes the business to decide to do some activity or to stop some activity.

Process conditions test the states of the business at the time of execution and change the business process depending on the results of the test.

Procedure conditions test the states of the attributes or fields or the actions of the operator at the time of execution and change the computer procedure depending on the result.

Condition Logic - The condition logic is an expression of a predicate condition that may be expressed in the form "If P then Q".

Conditional Statement - A unit of activity in an action diagram wherein a condition is expressed that controls a subsequent process, procedure or step execution.

Selection Condition - A rule expressed in terms of predicates and constants, used to select one or more entities of a given type for involvement during the execution of a process or a procedure step.

Selection conditions are used to describe which entity occurrences are to be acted on during an execution of a process or procedure. They consist of the criteria that are needed to select the required entities unambiguously.

Selection can occur by two methods:

1. Selection by the existence of a grouping under a named relationship
2. Selection by the range of values of a named attribute

That is, entities can be selected because they participate in a given relationship membership, and entities can be selected because some or all of their attribute values meet given criteria.

It is possible to combine both methods in a single selection condition. It is also possible to act on all entities of an entity type.

Conditional Membership - The membership of an entity in a pairing where that membership depends upon some predicate values. For example:

> READ existing customer WITH name = "Smith"
> READ account WHICH is owned by existing customer

Execution Condition - A rule expressed in terms of predicates or constants that states when an activity (process or action group) may or may not be executed. Execution conditions are used to determine the combination of actions that will enable each execution of a process or procedure. Not all actions occur every time a process or procedure occurs; execution conditions specify the circumstances in which an action does occur. For example:

> IF existing customer status IS EQUAL TO "priority" USE priority dispatch

Execution Preconditions and Post-conditions - Optionality, cardinality and exclusivity can all be affected by conditions that detail constraints on the execution of a process (i.e., preconditions) or on the possible outcomes of execution. Optionality conditions specify when a dependency may enable execution of a process, but its absence may not prevent execution.

Conditional Subprocess - A process whose execution depends on predicate values established by prior processes.

What Conditions Are and Are Not - Conditions are expressed in terms of entity states or time events. These conditions constrain the execution of the process and are defined for the process, but they are not part of the logic of the business process. They are part of the control logic, and can be documented within the process description. To define the conditions accurately, you must have analysed the entity type life cycles and have been through some iteration between dependency and life cycle analysis.

Cardinality Conditions - Cardinality conditions determine the number of executions of a dependent process or the number of times a preceding process may execute for each execution of a dependent process.

Exclusivity Conditions - Exclusivity conditions determine which one of a number of possible alternative dependencies may either enable a process or be established by a process.

Compound Conditions - Simple conditions can be combined into a single, compound condition that defines all constraints upon executions or outcomes of a process. The constraints may be the presence or absence of dependencies associated with the process. They can be combined using logical operations. For example, the precondition of PAY SUPPLIER could be WHEN month ends AND received supplier invoice OR part-paid supplier invoice AND IF scheduled payment

Preconditions and Results - The set of conditions that constrain the execution of a process is known as its precondition; the set that constrains the outcome is known as its result.

14.2.5 Decomposition

This is the breakdown of the activities of an enterprise into progressively increasing detail.

The principal technique for describing the functions of an enterprise is function decomposition. An activity can be decomposed when two or more other activities can be defined that together are equivalent to it.

For example, the process TAKE ORDER may be decomposed into the following processes:

- RECEIVE ORDER

- ALLOCATE STOCK OF BOOKS
- CHECK CUSTOMER CREDIT

14.2.6 Domain

This is the collection of values from which each of the values of one or more attributes or fields must be taken.

Domains to Group Attributes

An attribute describes only one entity type and has only one meaning; Employee Start Date is not the same attribute as Seminar Presentation Start Date and Employee Start Date is distinct from Employee Birth Date. These are, however, all dates. They share the same formats, and take their values from the set of all valid dates.

Sharing a domain has a further implication for attributes that can be compared or that may participate in the same computations. For example, it is permissible to add money values together, but not to add money to weight values.
Domains are useful for classifying similar attributes so that modules developed by separate teams can share data easily. The time spent in considering them will usually pay off in an increased understanding of attributes and a reduction in the complexity of the business description and significantly foster system integration.

Hierarchy of Domains

It may be useful to develop a hierarchy of domains so that conversion of values is evenly applied among all applications. For instance a domain of volume may include "cubic centimetre" and "gallon" and the algorithm used to convert from one to the other. A second example, important to international business is the domain "money" which contains Dollars, Euros and other currency.

14.2.7 Elementary Process

This is the smallest unit of business activity of meaning to a user. When it is complete, the elementary process leaves the business area in a self-consistent state. That is, the business person has come to closure on the process and all the business information is in a static and complete condition.

Criteria for Elementary Processes

To limit the decomposition, find a level of business activity that has meaning to the business as an activity, has enough detail to be useful, and is not so detailed that an execution of it would not imply that something useful and complete had been done. The elementary process is a business job, usually done by one person that cannot be stopped in the middle without missing some information needed for the next elementary process. For example, to make a telephone sale,

the elementary process must gather both what the customer wants and where the customer needs the material delivered.
It is always possible to find elementary processes, but their content and level are dependent on the business area context. What is elementary in the context of one enterprise, or even one business area, could be too high level or too low level in the context of another enterprise or business area. For example, in some enterprises, TAKE ORDER may be elementary, whereas in other environments, ALLOCATE STOCK OF BOOKS and CHECK CUSTOMER CREDIT may be elementary.

Self-Consistent Business Area

A process that leaves the business area in a self-consistent state is one whose output is complete and meaningful in itself. For example, a process may be defined that calculates the value of an order by calculating and summing the value of its items. As long as the values of individual items are not required by other processes, a process CALCULATE ITEM VALUE does not leave the business area in a self-consistent state. It provides partial or intermediate results that are of no interest to the user until they are all calculated and summed to give the final order value.

Level

Self-consistency alone, however, is not enough to define an elementary process. Any process at a level higher than elementary must also leave the business area self-consistent. For example, a process TAKE ORDER might have ALLOCATE STOCK OF BOOKS and CHECK CUSTOMER CREDIT as elementary processes. The process TAKE ORDER would, therefore, not be elementary even though it leaves the business area in a self-consistent state. It is possible that a user would see TAKE ORDER as a complete, self-contained unit with no useful subprocesses. In this case, the process TAKE ORDER would be elementary. Note that the level is defined in the context of the business user.

Implementation Factors

The identification of elementary processes may involve considering how processes have been implemented, because the user's perception is in terms of current procedures that support the process. Such implementation aspects should be set aside, and the business requirements should concentrate on defining the elementary process. The BAA should define the business as it should be, not as it is.

Unit of Work

When you define an elementary process, you are saying that there is a user activity that must be completely carried out for the business area to be satisfied. This often corresponds to a unit of work within an organisation. It is a task, often assigned to a single person, with measurable outputs and one that achieves some recognisable benefits for the enterprise.

Although an elementary process ideally corresponds to a single unit of work, it contains no consideration of how it will be achieved. The completion of the process could take an appreciable period of time. One aim of Business System Design will be to minimise this time. During the execution of the process, the business area may be inconsistent (i.e., certain business rules may be broken), and some entities may not be in recognisable states. When the process is completed, all business rules must be satisfied and all entities must be in recognised states.

14.2.8 Entity

This is a fundamental item of relevance to the enterprise, about which the enterprise needs to keep information.

Entity and Entity Type - Most businesses have suppliers, deal with products, and sell them to customers. There are thus likely to be data about individual suppliers (e.g., Poly Leisure Products of Hong Kong). Poly Leisure Products is an entity; the collection of all suppliers is the SUPPLIER entity type. The collection of all customers is the CUSTOMER entity type. A sample customer entity may be Harrods Department Store. Similarly, there is a PRODUCT entity type.

Entities that are Tangible Objects - Following are examples of entities that are tangible objects:

- ORDER, of which one entity might be a request on 6 November for 50 footballs from Poly Leisure Products
- VEHICLE, with an entity Ford Model T, Registration Number 135 AXU
- ROOM, with entities kitchen at 14 Conway Road, boardroom at James Martin & Co.

Entities that are Activities or Events of Interest to the Business - Following are examples of entities that are activities or events:

- LECTURE ATTENDANCE (e.g., John Davis attends the lecture on Entity Relationship Modelling on 3 May 1997)
- EQUIPMENT MALFUNCTION (e.g., the lighting dimmer in lecture room 2 fails at 3 PM. on 4 June 1995)

Entities that are Intangible - Following are examples of entities that are intangible:

- GEOGRAPHICAL SALES AREA (e.g., Europe, Asia, Australia)
- MARKET SECTOR (e.g., consumer, electronics, instrumentation)
- CUSTOMER TYPE (e.g., distributor, retail store)
- BUSINESS ROLE (e.g., principal, guarantor, legal advisor, broker)

14.2.9 Entity Type

This is a collection of all the entities important to the business, to which a specific definition and common predicates apply.

Most businesses have suppliers, deal with products, and sell them to customers. There are thus likely to be data about individual suppliers (e.g., Poly Leisure Products of Hong Kong). Poly Leisure Products is an entity; the collection of all suppliers is the SUPPLIER entity type. The collection of all customers is the CUSTOMER entity type. A sample customer entity may be Harrods Department Store.

Tangible Entity Type - An entity type whose entities are physical objects or particular views of physical objects.

Conceptual Entity Type - An entity type whose entities are intangible objects without physical form.

Associative Entity Type - An entity type whose entities relate to two or more entities. Associative entity types are used to resolve many-to-many relationship types in an entity relationship diagram. For example, each entity of the entity type CLUB MEMBERSHIP relates one entity of CLUB with one entity of MEMBERSHIP.

Collective Entity Type - An entity type whose entities gather together or consist of entities from one or more other types. In some cases, a collective entity type may be helpful in improving the performance of a system.

Component Entity Type - An entity type whose entities form part of a larger collective entity type.

Overlapping Entity Types - Two or more entity types with entities in common. Overlapping entity types are to be avoided.

Isolated Entity Type - An entity type that does not participate in any relationship.

Transient Entity Type - An entity type whose entities are of only short-term interest to the enterprise.

14.2.10 Entity Type Horizon

This is a context within which an entity type is identifiable by a specific identifier, but beyond which the entity type is not uniquely identifiable.

Each entity type has a logical horizon. This is the collection of other entity types (possibly none) that can be uniquely selected by following chains of many-to-one relationships away from it.

Each entity type has a logical horizon. This is the collection of other entity types (possibly none) that can be uniquely selected by following chains of many to one

relationships away from it. For example, the logical horizon for an ATTENDANCE is:

COURSE, SESSION and ITEM will form an initial grouping, as will STUDENT and BOOKING, because of their mandatory association with the COURSE grouping, both directly to SESSION and indirectly through COURSE PRESENTATION, it is more strongly associated with the COURSE grouping than with the STUDENT grouping, to which it has but a single association. At a further level of summarisation, it is therefore merged into the COURSE grouping.

Entity Type Horizon for Unique Identification

For an ORDER ITEM, there is a "horizon of identifiability" which is called the entity type horizon. Within this horizon, the ORDER ITEM can uniquely identify singly related entity types, but anything outside of the horizon cannot be identified. For example, if a CUSTOMER makes many PAYMENTs, there is no way of distinguishing individual PAYMENTs from the viewpoint of an ORDER ITEM or even an ORDER. For each PAYMENT entity, there is a horizon that encompasses its CUSTOMER but that does not include any ORDERs.

You will find horizons useful when you analyse the structure of entity relationship models. For example, you may know that you want to be able to determine, uniquely, which ORDER a DELIVERY was for. So you know that ORDER must be within the horizon of a DELIVERY.

You can also consider the optionality of relationships when analysing horizons. For example, if the plural end of a one-to-many relationship is optional, you cannot assume that the single entity type is identifiable, because there may be no pairing.

14.2.11 External Object

This is a source or destination of information, materials or resources outside the immediate scope of analysis.

Because some producers or receivers of data are outside the system or business area under investigation, you cannot tell what procedures produce or accept the data, or what happens to those data. There must be procedures to or from which the data are flowing, but because you cannot formally identify them, you identify the producers or receivers as external objects.

Such external objects are typically named by using the name of an entity type that represents the procedures, people or organisations receiving or producing the data flow.

Imports and Exports

It is useful to describe not only the imports and exports, but where they come from and go to. Imports and exports link an activity in the business with its corresponding sources and destinations.

The external sources and destinations of information, materials or resources are represented as external objects (e.g., CUSTOMER, SUPPLIER, GOVERNMENT AGENCY). They carry out their own activities, which create or use the imports or exports through which they interact with your enterprise. They are therefore of interest to the business, and they usually also appear as entity types in the information model.

14.2.12 Function

This is the highest, most aggregated level of business activities that together completely support one aspect of the enterprise's mission considered in analysis. A function may consist of other functions or processes, both being business activities.

Examples of Possible Functions

- PURCHASING
- DISTRIBUTION
- SALES
- PERSONNEL
- RESEARCH

Component Activities Necessary to Ensure Stock is Available

A function called STORING has as its purpose to ensure that stock is available for delivery to customers. In order to do this, it must perform several of the following activities.

- Receive Deliveries from Suppliers
- Store Books
- Issue Books for Delivery to Customers
- Monitor Stock Levels
- Request Publishing of Replacement Stock

Characteristics of the Component Activities

For each of the component activities above, questions of the following type may be asked:

- How often does it take place?
- What are the inputs and outputs for each occurrence of the activity?
- How long does an occurrence of the activity take?

Difference Between Function and Process

Applying the same questions to STORING as a whole is, however, meaningless. It is meaningless to think of an occurrence or execution of STORING, although this can be done for one of its components. This is what distinguishes functions from processes.

Difference Between Function and Organisational Unit

At first sight, functions may seem like organisational units, and there may, in some organisations, be a close correspondence. Although this correspondence is not coincidental, a function is not an organisational unit. A function is independent of present or future organisational structures, because it represents the "what" of the business as opposed to the organisational unit's "how."

Functions Found During the ISP

Identification of functions is a principal goal of Information Strategy Planning (ISP). A group of these functions forms part of the definition of the scope for the Business Area Analysis project. The project is not normally expected to look for new functions unless better information suggests that improvements should be made to the function decomposition.

14.2.13 Identifier

This is a predicate or set of predicates whose values distinguish each entity of a specific type from others of the same type.

Primary Identifier

The preferred means of distinguishing each entity of a type from others of the same type.

When alternative identifiers are found, designate the identifier likely to be the most useful as the primary identifier.

Ways of Distinguishing Entities

There must be at least one way of distinguishing entities of a type, or you do not have a valid entity type. It is, however, possible to have more than one way of identifying entities. Distinguish between entities of the same entity type by:

- An attribute whose value distinguishes the entities
- The existence of a relationship to another entity
- A combination of attributes and relationships

Single Attribute Value Identifier

Attribute values may be used to identify entities uniquely. A single value may suffice (e.g., "JM & Co. 1" identifies a vehicle). Alternative attribute values may also be employed. For example, if JM & Co. 1 is owned by James Martin & Co., it may also be identifiable by a unique asset number. An asset number as the identifying attribute of the entity type VEHICLE, however, is useful only for company-owned vehicles. Note also that asset number is a designed attribute.

Composite Identifier

Two or more predicated (attributes or relationships) whose values, taken in combination, uniquely identify the entities of one type.

When the values of more than one attribute are needed in combination, the identifier is composite. For example, SUPPLIER might be identified by Name and Location, as in James Martin & Co., Reston. VA.

It is possible to identify entities with relationship memberships or with a combination of relationship memberships and attributes. Relationship memberships are used when the identifying value will be found in a related entity type. In short, an entity can be identified by any combination of predicates.

For example, an ORDER ITEM may be identified by its relationship with a particular order and Item Number (an attribute of ORDER ITEM). Item Numbers may be duplicated across a set of ORDER ITEMS for different ORDERs, so the occurrences must be distinguished by finding the ORDER that they belong to. Order Number is associated with an ORDER ITEM by virtue of the relationship between the two entity types. It is in part the existence of a relationship membership that identifies the ORDER ITEM.

An entity may be identified purely by relationships. For example, the entity type PRODUCT STOCK is identified uniquely by its relationship with the PRODUCT and WAREHOUSE LOCATION entity types.

14.2.14 Location

This is a physical place at which activities of interest to the enterprise may be performed or hardware may be installed.

A location is a physical place at which activities of interest to the enterprise may be performed by the enterprise itself or by external organisational units. For example, a warehouse would be of interest, either if owned by the enterprise or if used by the enterprise but managed by a specialist supplier.

A location is not always equivalent to a physical address (e.g., headquarters and a regional sales office may share a postal address).

A location need not even be fixed geographically (e.g., a ship or a salesperson's car, especially given adequate radio links, may be an acceptable location).

A location is not an organisational unit, although it may sometimes share a name with a unit.

Location Type

A classification of locations based on the similarity of their role or purpose and of their level of activity.

There may be many locations that perform the same role (e.g., sales offices, factories or warehouses). Location type groups the locations that perform similar roles and that share broadly similar frequencies of activities and volumes of data.

Location number is the number of locations of each type in which activities of the enterprise are performed. It, together with any anticipated growth factor, is documented as a property of the location type.

14.2.15 Normalisation

This is the decomposition of a data structure, to remove any implicit dependencies between objects within the structure.

Normalisation is a staged procedure that reduces any given collection of attributes to several, more basic collections called "normalised relations." A fully normalised relation is a group of attributes in which the existence of values for the non-identifying attributes is entirely dependent upon the existence of values for the identifying attributes of the group.

Normalisation is used within correctness checking to help decide whether an attribute has been assigned to the correct entity type, and whether further entity types are needed in the model.

Normal Form

The results of the stages of normalisation are attribute groups in some normal form: First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), Fourth Normal Form (4NF) and Fifth Normal Form (5NF).

Each normal form serves to remove ambiguities from the database. In an unnormalised collection of attributes, the values can be organised in repeating groups (e.g., multi-valued attributes) with many dependencies between them. In a 5NF collection of attributes, you will have separated out all multi-valued attributes and eliminated all but the most direct dependencies and provided the most flexible access to unambiguous data.

First Normal Form

1NF achieves the separating of multi-valued attributes by constructing a distinct separate collection of entity types to contain the multiple values. In 1NF, no entity type has repeating groups.

Second Normal Form

2NF achieves the separation of entity types that are not dependent on the complete identifier of the group. In 2NF, no entity types are only partially dependent on their identifiers.

Third Normal Form

3NF achieves the separation of entity types that are dependent on attributes other than the complete identifier. In 3NF, no entity types are indirectly dependent on their identifiers. A well designed entity relationship diagram is a third normal form schema.

Fourth Normal Form

4NF achieves the separation of entity types that are conditionally dependent on other attributes. In 4NF, every entity type is directly dependent on its identifier under all circumstances. Using entity subtypes in an entity relationship diagram places the schema into the fourth normal form.

Fifth Normal Form

5NF achieves the separation of large composite identifiers into independent tables, each with only a pair of identifying attributes. This can achieve a reduction in the level of redundancy among the total set of attribute values, but because it adds little to the meaningfulness of the model, it is generally not employed.

14.2.16 Optionality

This is a type of predicate condition that places constraints on whether a dependency, relationship or attribute must exist.

Optional Attribute Type

An attribute type that need not have a value for every entity of the type it describes.

Optional Relationship Membership

Membership of an entity type in a relationship, such that entities of the type may exist without participating in a pairing under the relationship.

Optional Dependency

A process dependency under which execution of one of the participating processes may execute without resulting in or following from the execution of the other.

14.2.17 Pairing

This is where two entities of one or two types associated by virtue of a defined relationship.

Relationship, Pairing and Grouping

Entities are the occurrences of an entity type, but what is an occurrence of a relationship? It can be thought of as an association between two entities. This is known as a pairing. For example, "Hamlet written by Shakespeare" is an occurrence of the relationship "written by."

When you want to consider an AUTHOR and all the BOOKs the author has written, use the term grouping. Given joint authorship, a grouping could consist of a book and all the authors who helped write it; conversely, there would be a grouping for each author and all the books the author had written. In effect, a grouping is a set of pairings.

Pairing Action

An action on one or more pairings of a given relationship by an elementary process following an action on an entity.

Pairing Membership

The participation of an entity in a pairing.

14.2.18 Partitioning

This is a basis for subdividing the entities of one type into subtypes.

Partitionings

The reason for subtyping is called partitioning. For instance, employee may be partitioned because the pay calculations for salaried and hourly employees require different processes and/or different data. The entity type that is subtyped uses the classifying predicate to identify which portion of a partition (which subtype of the partition) the particular entity uses.. Business entity types may be subtyped because there is reason to deal with some entities differently than others.

The subdividing of EMPLOYEEs into either ADMINISTRATORs, EDITORIAL STAFFERs or PRINT ROOM STAFFERs is a partitioning of the entity type. The name of the classifying attribute indicates the purpose of the partitioning, which often has the same name. For the above example, the EMPLOYEE entity type has a "Job Type" partitioning consisting of ADMINISTRATOR, EDITORIAL STAFFER and PRINT ROOM STAFFER subtypes. The classification of entities into one of the subtypes is achieved through the Job Type attribute of EMPLOYEE.

Partitioning Name

A partitioning name is a way of referring to and labelling a partitioning.

Multiple Partitionings

If you decide that the distinction between potential, current and former employee is relevant, you can define another partitioning for EMPLOYEE so that there are

multiple partitionings of EMPLOYEE. Each EMPLOYEE may be a member of both partitionings (e.g., a POTENTIAL EMPLOYEE and EDITORIAL STAFFER, a POTENTIAL EMPLOYEE and an ADMINISTRATOR). You need a classifying attribute (e.g., Employee Status) for the new partitioning.

Hierarchy of Partitionings

It is possible to define a hierarchy of partitionings. For example, you might wish to partition administrators further into accountants and salespersons. Thus, ACCOUNTANT is a subtype of ADMINISTRATOR, which, in turn, is a subtype of EMPLOYEE.

Conditions

The membership of an entity in one of the subtypes of a partitioning is determined by a condition that is a property of the partitioning. This condition usually involves the classifying predicate, but it could be more complex, perhaps involving other attributes or relationships. Any attribute or relationship of an entity type can be used to classify entities into its subtypes.

For example EMPLOYEEs can be classified into POTENTIAL, CURRENT and FORMER, depending solely on the classifying attribute "Employee Status." Whether a CURRENT EMPLOYEE is also a TRAINEE, PERMANENT or FREELANCE, however, might be determined by Employee Status, Service Length and Qualifications, together with membership as a plural entity in an "EMPLOYEE managed by EMPLOYEE" grouping.

For example: subdivisions of EMPLOYEE

An entity subtype is a subdivision or further specialisation of an entity type. For example, the entity type EMPLOYEE might be subdivided in several ways to aid in analysis of this entity type. These might include:

- ADMINISTRATOR, EDITORIAL STAFFER and PRINT ROOM STAFFER
- POTENTIAL, CURRENT and FORMER EMPLOYEE
- TRAINEE, PERMANENT and FREELANCE EMPLOYEE
- MALE, FEMALE EMPLOYEE

Optional Partitioning (i.e., Optional Subtype Classification)

A partitioning for which some entities of the entity type may not be members of any of its subtypes.

You may find that VICE PRESIDENT is a useful subtype of EMPLOYEE. Some EMPLOYEEs are VICE PRESIDENTs; most are not. Therefore, membership of this partitioning, which happens to have only one useful subtype, is optional. In the case of ADMINISTRATORs, EDITORIAL STAFFERs and PRINT ROOM STAFFERs, an EMPLOYEE has to be classified into one (and only one) of the subtypes. In this case, membership of the partitioning is mandatory or fully enumerated.

An optional classification is the equivalent of defining an "other" subtype in a partitioning. For example, you could define a mandatory classification for EMPLOYEE to be VICE PRESIDENT or OTHER EMPLOYEE that indicates that an employee must be either a vice president or another type of employee that you have not specified.

When participation in a partitioning is optional, you describe the partitioning as not fully enumerated. This implies that the population of entities existing for the entity type is not completely covered by the subtypes defined within the partitioning (i.e., some lie outside and are not counted within it).

Transient Partitioning

A partitioning for which the membership of an entity in a subtype may change several times during the entity's life.

14.2.19 Predicate

A predicate is an attribute type or a relationship membership. A predicate value is an attribute value or a pairing.

Predicate and Predicate Value

You will need to know many things about discovered entity types. Some of these things will be other entity types (e.g., SALEs to each CUSTOMER). You describe this by defining the other entity types and associating them by relationships. The other things you need to know about entity types are particular to the entity type and describe it (e.g., a Customer Name and Address). These are the attributes that describe an entity type. Attributes and relationships have a similar effect on entity types; they also have similar properties. Because of this, you can use the generic term "predicate" to describe them, and "predicate value" to describe their occurrences.

Predicate Condition

A condition that affects the values of predicates and their properties. You often need to define conditions that affect the values of predicates and their properties. These are called predicate conditions. There are a number of different types of predicate conditions.

Conditional Membership

A common cause of optionality is conditional membership, where the participation of an entity in a pairing depends upon predicate values (e.g., BOOK can be used for TEACHING if the condition "Book Type = textbook" is true).

Condition Logic

The logic "Book Type = textbook" is known as the condition logic for the condition. The attribute values in the condition logic need not belong to the entity

types affected by the condition. Also, the condition logic may include participation in a relationship membership (e.g., BOOK sent to DISTRIBUTOR if the condition logic "book is published" is true). A condition can be expressed through a number of varying condition logic statements, but it can still have the same meaning (e.g., the condition logic "A nor B" is the same as the logic "NOT (A or B)," both of which can apply to a single condition).

Integrity Conditions

Membership conditions may be compound. For example, the previous condition might be made more restrictive by adding the condition that the "distributor type is not a book club." Membership conditions are one way to ensure the integrity of your models and are thus one type of integrity condition. Integrity conditions in their turn are one type of predicate condition.

Cardinality Conditions

It is also possible to define conditions to ensure the integrity of cardinality of a relationship membership where a maximum or minimum cardinality can be defined. Cardinality conditions are rarely used.

14.2.20 Procedure

This is a method by which one or more elementary processes may be carried out.

A procedure is a method by which a specific process may be carried out. It defines how the activity is carried out as well as what the activity is. The method used to perform a process is referred to as a technique (e.g., on-line technique, batch technique, clerical technique).

Procedures in Current Systems Analysis

A procedure in current systems analysis is a named package of computer software or manual activities. The lowest level of concern in BAA Procedure Analysis of Current Systems is the one actually named in procedure manuals, system documentation or user discussions. There may be abstractions of procedures, such as subsystems or systems.

Procedure Definition

Having identified the procedures required to implement the elementary processes, you define each procedure in turn. This definition translates the business logic of a process into a specific method of supporting the process. This provides a starting point for the subsequent detailed procedure design.

Classification of Procedures

A policy is adopted in which the division into modules and the choice of languages follow the same approach for each type of program function. This

creates consistency throughout the system, providing easier programmer interchange, easier debugging and higher productivity.

Procedures or subsets of procedures can generally be described as having one of the following functions:

- Query
- On-line update
- Batch update
- Extract
- Report

Procedure Condition

A rule expressed in terms of logical data constructs, layouts and constants that constrains the behaviour of the business system.

Procedure Decomposition

The breakdown of the designed procedures of a current system into progressively increasing detail. Identify systems, decompose the systems into subsystems and the subsystems into procedures.

14.2.21 Process

This is a defined business activity, executions of which may be identified in terms of the input and output of entities of specific types, or in terms of data about entities of specific types.

Processes are the specific activities of the enterprise. The process is described in terms of the entity types involved, such as CUSTOMER, ORDER or PRODUCT. The definition draws attention to the distinction between a process and an execution of it.

Execution of a Process

An execution of a process is concerned with the specific entity occurrences involved (e.g., Acme Company, Order 106359, 2-cm Screw) and the values of associated attributes (e.g., Quantity Ordered and Unit Price). Each execution of a process is uniquely identified by the states of the entities involved, the values of their attributes and the time of execution. Furthermore, the outcome of an execution of a process or its completion criteria are well known (e.g., an ORDER is either accepted or rejected).

The same cannot be said for a broader business activity, such as SELLING, where the idea of an execution is meaningless. An execution of SELLING might involve taking an ORDER, setting a salesperson's revenue objectives, or analysing last year's sales. In other words, an execution could mean different things at different times. It is therefore not useful to speak of an execution of SELLING. SELLING can instead be defined as a function.

Examples of Processes

- SCHEDULE DELIVERY
- PAY AUTHOR
- INVOICE CUSTOMER
- PAY EMPLOYEE
- CHECK CUSTOMER CREDIT
- COMMISSION BOOK

Processes Describe WHAT, Not HOW

Because processes describe what the business does and not how it is done, it is not necessary to consider:

- What techniques are used
- When it is done
- Where it is done

In practice, many techniques may be employable for implementing a process. A set of steps for implementing a process with a chosen technique is called a procedure. Procedures are not described during Business Area Analysis; instead, they are designed during the Business System Design phase of the Analysis Stage, because they are system activities rather than business activities.

Scope of Process Analysis During BAA

The identification of processes is a principal goal for a Business Area Analysis project. Functions in the project scope are analysed to reveal every process.

14.2.22 Relationship Type

This is a reason of relevance to the enterprise why entities from one or from two entity types are considered as a logical unit. A relationship type is often referred to simply as a relationship.

Relationship and Relationship Membership

Relationship membership is the participation of an entity type in a relationship. The associations between its entity types are as important to a business as the entities, themselves. These are analysed as relationship types. The occurrence of a pairing is a relevant association between two entities. For example, Ian Palmer "writes" Database Management, where "writes" associates an AUTHOR entity type with a BOOK entity type. The relationship type is named with the verb (e.g., "writes") that states the reason why entities of the two types may be associated.

A relationship type can be seen from the point of view of each of the two participating entity types. The name is different, depending on which entity type is being considered.

For example, the "writes" relationship is seen from:

- The Author's viewpoint as: AUTHOR "writes" BOOK (active name)
- The Book's viewpoint as: BOOK "is written by" AUTHOR (passive name)

These two views are called the relationship memberships. Each relationship pairing always has two memberships.

Relationship Viewpoints

In the above example, you have taken the viewpoint of each entity type and considered whether it is an active or passive participant in the relationship. Each of these viewpoints is a relationship membership, and each relationship pairing has two of them. A relationship type often comes about because there is some process that associates the entity types (e.g., WRITE BOOK). If there is such a process, the name of the relationship type will come from it. In this case, the active participant is the entity type that carries out the process (e.g., it is the AUTHOR who writes the BOOK). Similarly, the passive participant is the entity type that is the object of the process (e.g., the BOOK that is written).

Relationship Action or Pairing Action

A type of action on one or more pairings of a given relationship by an elementary process following an action on an entity.

Relationships Reflect Business Rules

Relationship Cardinality

The number of times an entity may participate in a membership pairing with another entity or entities. For instance a person may be the "parent of" zero, one or many other persons.

Membership or Relationship Optionality

Membership of an entity type in a relationship, such that entities of the type may exist without participating in a pairing under the relationship.

When describing a business, determine whether or not an entity of a given type always participates in a pairing of a given relationship. For example, an EDITION is always of a BOOK and cannot exist unless it is associated with a BOOK. The relationship membership "EDITION is printing of BOOK" is therefore said to be mandatory.

You may, however, find that at some time there are BOOKs that have not yet been printed as an EDITION. The relationship membership "BOOK printed as EDITION" is therefore said to be optional.

Optionality of Each Relationship Membership

The state of being mandatory or optional is a property not of the whole relationship, but of each of the two relationship memberships. Optionality of a relationship considers whether entities of a given entity type can exist without participating in any pairing under that relationship.

Mandatory membership implies a cardinality of at least one. Optional membership implies that the cardinality may be zero.

Fully Optional Relationship Membership

A relationship under which entities of both entity types can exist without participating in some pairing.

When both relationship memberships are optional (e.g., PUBLISHER commissions BOOK, where PUBLISHERs exist who do not commission any BOOKs, and BOOKs are written that have not been commissioned by a PUBLISHER), the relationship is called fully optional.

Most relationships are partly optional (i.e., only one of the relationship memberships has the property of optional).

Transient Relationship Membership

The membership of an entity of a specific type that may change between groupings under a specific relationship several times during that entity's life.
Exclusive Relationships

Two or more relationship memberships in which each entity may participate in any one, but not more than one, grouping under these relationships.

Exclusive Relationships and Integrity Conditions

If, at any time, a pairing holds between two entity types, it occasionally excludes the possibility of one or both of the entity types having pairings with other entity types at that time. In other words, the possibility of pairing between two entity types may depend on the lack of a pairing between one or the other or both of them and some other entity types.

For example, a PAYMENT may pay for one or more BOOKs or one or more MACHINEs but not for a mixture of both. Thus, the "PAYMENT pays for BOOKs" relationship membership and the "PAYMENT pays for MACHINEs" relationship membership are mutually exclusive. We call these exclusive relationship memberships.

Integrity Conditions

Exclusive relationships must be implemented through data integrity conditions. Exclusivity can be affected by an integrity condition, which can be complex.

A simple condition for the above example may be:

> If Payment Method = cash, the PAYMENT must be for BOOKs (because the company will never pay for machinery by cash)

A more complex condition may be:

> If Payment Method = cash and PAYMENT is sent to AUTHOR, PAYMENT pays for one or more BOOKs

14.2.23 Reusability

This is the ability of an existing component to be incorporated or modified to fit into multiple models or applications.

Reusable Component

An object or defined collection of objects in any stage of the development life cycle that may be used in multiple applications. Reusable components may be of the following types:

- **Application**: an entire system that may be reused as is or modified to fit the user requirements.
- **Building Block**: a part of a system that may or may not be able to stand on its own, such as an algorithm, a structure, a library module, a help system or knowledge of great complexity.
- **Application Shell**: an outline of the generic components of an application, that may include comprehensive features common to applications, such as security, auditability, checkpoint-restart, protection from failures, human factoring, cooperative processing, etc., where the developer fills in the shell with the specific application details. The resultant product is less likely to have errors, instability, omissions and misunderstandings.
- **Template**: a pattern that can be copied to create a component with the same or similar characteristics as the template. Examples include structures to indicate choices such as dialog box, action bar, pull-down menus; transaction processing dialogs; data entry screens; HELP screens; error messages; environments, such as windowing.
- **Generic Data Model**: a generalised data model for an industry or line of business.
- **Data Structure**: a record structure or a portion of a database.
- **Communication Standards**: standard communication formats for exchanging information. Examples of standard formats include ANSI X.12, X.400 (electronic mail), ISO EDIFACT.
- **Documentation Standards**: standards for the kind and format of information to include in technical and user documentation. Examples include standard table of contents for each type of manual, diagramming standards, reusable templates and text for a "Getting Started" manual.

Building Block

This is an existing component that may be used to build a system.

Reusable Construct

This is a system component that may be used without change in more than one application. Common examples are library modules and help screens.

Generic Reusable Construct

A component that implements a feature or features common to most computer applications. Examples of generic constructs are start module, command processor, menu module, security module, work module, transaction processor, query module, user dialog module, error processor and stop module.

Application-Specific Reusable Construct

A component that implements a feature or features used by a particular kind of application and is applicable to a number of situations in the application.

Reusable Object

This is an object that may be useful in implementing more than one application.

Reusable Object Library

A repository specifically defined to contain reusable objects for an enterprise.

Reusable Structure

This is an application-specific or generic component or a skeleton that may be used in more than one application.

14.2.24 Subject Area

This is an area of interest to an enterprise, centred on a major resource, product or activity of the enterprise and about which the enterprise may wish to hold data.

A subject area is a major subdivision of the data objects (e.g., entity types and attribute types) that the business uses. It is defined initially during an Information Strategy Planning (ISP) study to aid in the finding of entity types, or as part of the definition of a Business Area Analysis (BAA) project. Once the entity types are found, the areas are then useful as broad groupings of entity types. It is best to keep the subject areas simple, so that they consist of only those entity types and interrelationships that pertain to a particular subject entity type.

An analysis project may involve several subject areas. For example, a sales analysis project may consider the Sales, Personnel, Customer and Product subject areas. As the analysis progresses, you will find that a subject area consists of a collection of entity types and the relationships between them.

Hierarchy of Subject Areas in the BAA

In Business Area Analysis, complexity is added, so it may be useful to form a hierarchy of subject areas either for presentation, or for the management of effort and the understanding of the model.

14.2.25 Subtype

A category of entities of an entity type that either:

- Has different attributes than the remaining entities
- Shares in different relationships than the remaining entities
- Shares different processes than the remaining entities

A collection of entities to which a specific definition and common predicates apply.

Subdivisions of the Entity Type EMPLOYEE

An entity subtype is a subdivision or further specialisation of an entity type. For example, the entity type EMPLOYEE might be subdivided into:

- ADMINISTRATOR, EDITORIAL STAFFER and PRINT ROOM STAFFER
- POTENTIAL, CURRENT and FORMER EMPLOYEE
- TRAINEE, PERMANENT and FREELANCE EMPLOYEE
- MALE, FEMALE EMPLOYEE

Making a Subset into a Subtype

If these are useful subsets of the population of EMPLOYEE entities, it is worth recording them as subtypes of EMPLOYEE, defining what is special about the subtypes and analysing their additional attributes and relationships.

For example, of all EMPLOYEEs, only EDITORIAL STAFFERs can edit and proofread BOOKs and commission AUTHORs, so there are additional relationships to the subtype EDITORIAL STAFFER. Only EDITORIAL STAFFERs have Subject Experience and relevant Qualifications. These are additional attributes of the subtype.

Note that EDITORIAL STAFFER still has all the attributes and relationships of EMPLOYEE.

Optional Subtype Classification

You may find that VICE PRESIDENT is a useful subtype of EMPLOYEE. Some EMPLOYEEs are VICE PRESIDENTs; most are not. Therefore, membership of this partitioning, which happens to have only one useful subtype, is optional. In the case of ADMINISTRATORs, EDITORIAL STAFFERs and PRINT ROOM STAFFERs, an EMPLOYEE has to be classified into one (and only one) of the subtypes. In this

case, membership of the partitioning is mandatory. Describe this classification as fully enumerated.

An optional classification is the equivalent of defining an "other" subtype in a partitioning. For example, you could define a mandatory classification for EMPLOYEE to be VICE PRESIDENT or OTHER EMPLOYEE (i.e., an employee must be either a vice president or another type of employee that you have not specified).

When participation in a partitioning is optional, describe the partitioning as not fully enumerated. This implies that the population of entities existing for the entity type is not completely covered by the subtypes defined within the partitioning (i.e., some lie outside and are not counted within it).

The parent entity type must contain an attribute or combination of attributes from which one can determine which subtype applies to each entity. These attributes are called 'classifying' attributes. For instance, if EMPLOYEEs are either 'hourly' or 'salaried' then an attribute type called 'employee type' that takes on a value of 'H' or 'S' would be a classifying attribute.

14.3 Rules

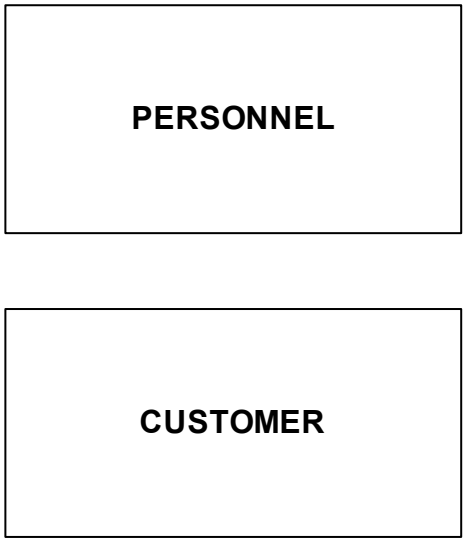
14.3.1 Diagram Conventions for Entity Types

An entity type is depicted by a rectangle known as the entity type box. This distinguishes it from processes and other activity object types that are represented as "cushions" (rectangles with rounded corners). The entity type box may be coloured or shaded to distinguish it from a subject area or design object.

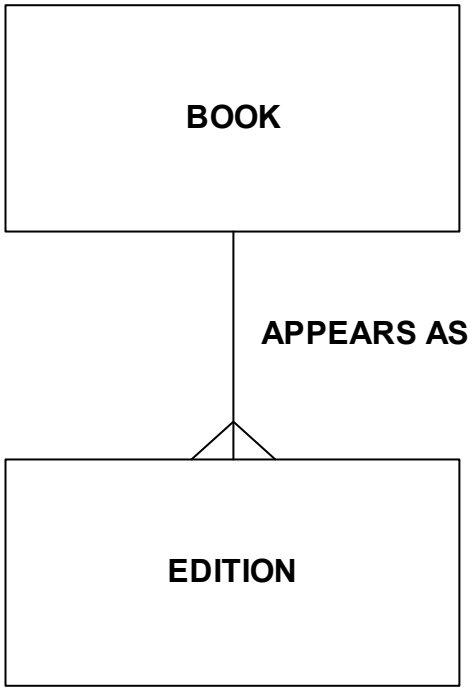
The rectangle always contains the entity type name. In addition, in manually produced diagrams, it may contain some attribute type names (in smaller letters), and entity partitionings and subtypes (in smaller rectangles).

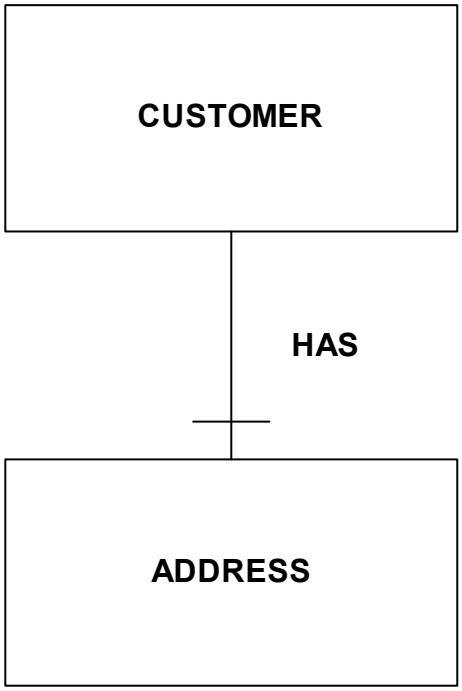
14.3.2 Diagram Conventions for Relationship Types

A relationship type is depicted by a line connecting the related entity types. One relationship membership name is included alongside the line. The other relationship membership name is documented but not shown along the line (normally one name gives the diagram sufficient meaning). The name is chosen to make the diagram readable.



Cardinality of one is symbolised by a bar - | - written across the relationship line. The symbol for cardinality of many is a crow's foot (in preference to an arrow, which connotes direction) adjacent to the plural entity. A one-to-one relationship has no crow's feet, a one-to-many (1:M) has a crow's foot at one end only, and a many-to-many (M:M) has a crow's foot at each end.



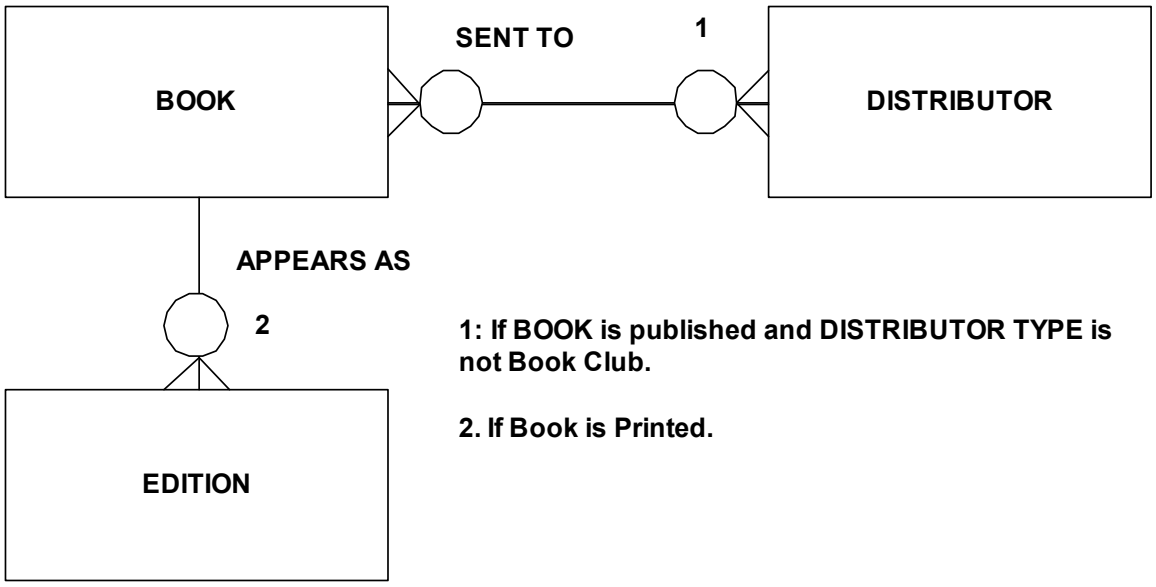


Optionality and Annotated Conditions

The symbol for optional relationship membership is a small circle across the relationship line next to the crow's foot or bar (adjacent to the entity type whose entities optionally participate in the relationship). Thus, a fully optional relationship has an optionality symbol at each end of the line, and a partly optional relationship has this symbol at one end only.

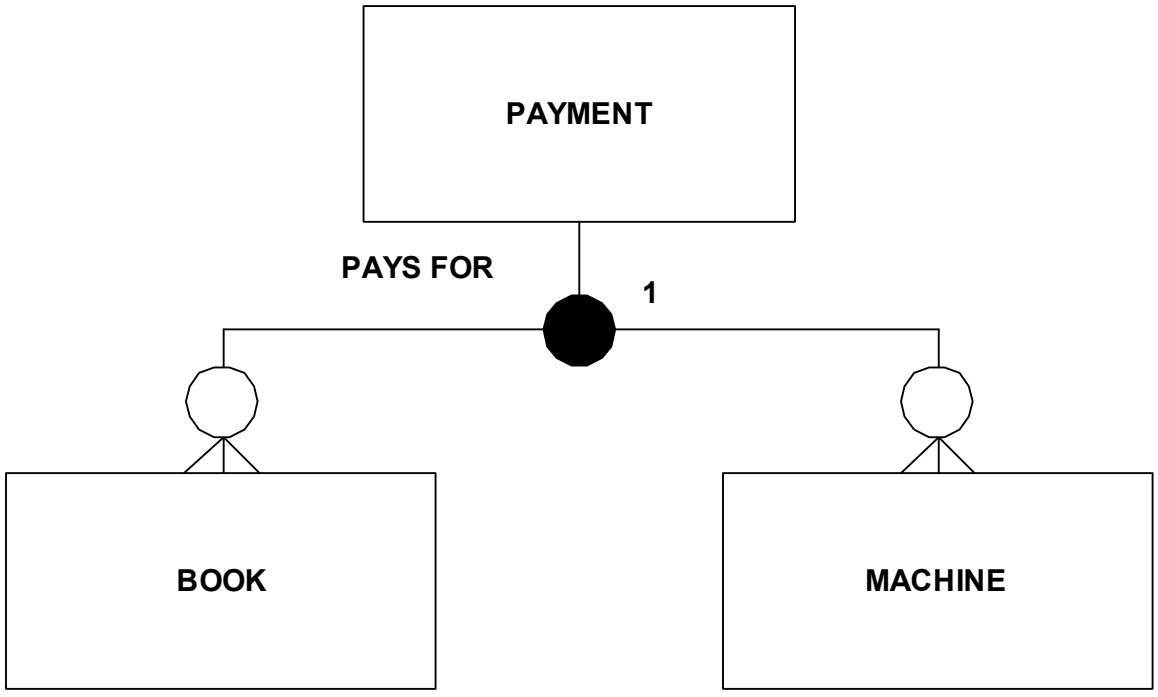
When relationship membership is conditional, the optionality circle may have, as a manual annotation, a condition number or, alternatively, the condition logic written alongside.

We do not describe cardinality conditions on the diagram except in exceptional circumstances where the condition is very important to the business. If it is shown, the logic is written beside the cardinality symbol. However the condition should always be fully documented in the relationship membership description,

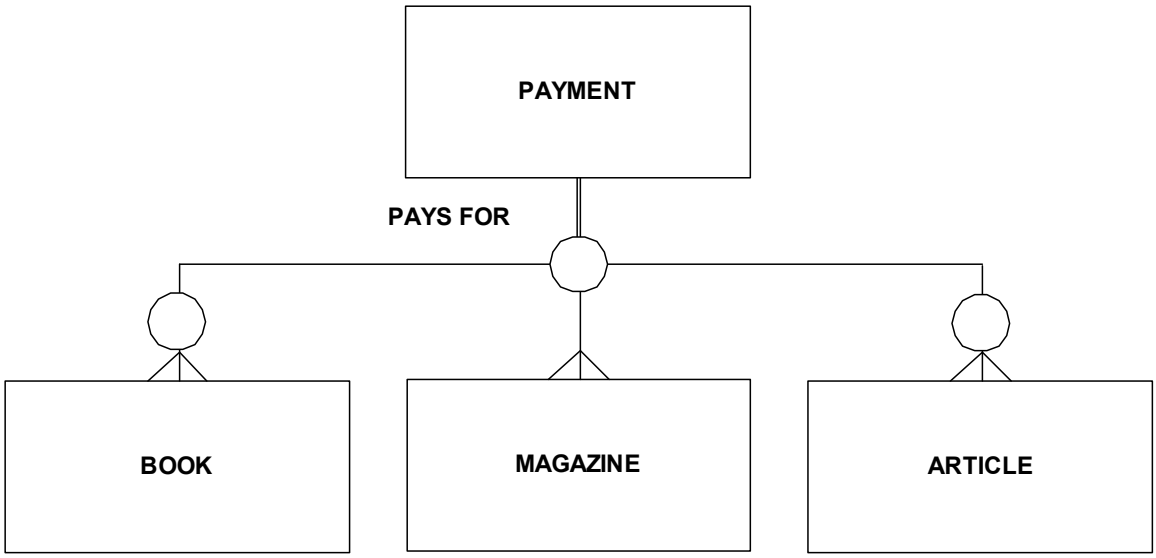


Exclusive and Inclusive Relationships with a Condition

The symbol for exclusive relationship membership is a circle intersecting all the relationship lines involved and adjacent to the entity type whose participation in the relationship is constrained. An exclusive relationship may have a determining condition in which case it may be indicated by an annotation beside the circle; inclusive relationships, if employed, are the same but without the circle.

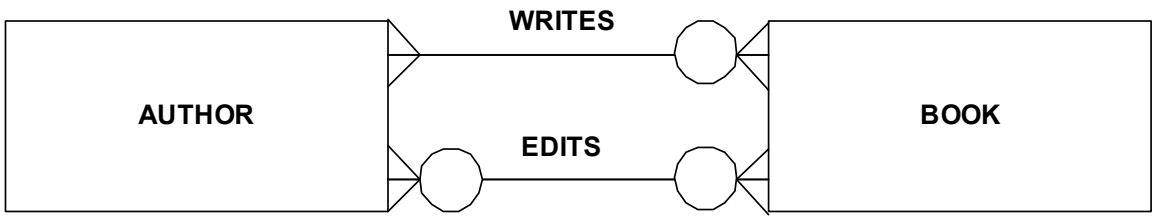


1: A PAYMENT is either for BOOKS or MACHINES.



Parallel Relationship Types

A parallel relationship is shown by it following a path beside the relationship it is parallel to. For example:



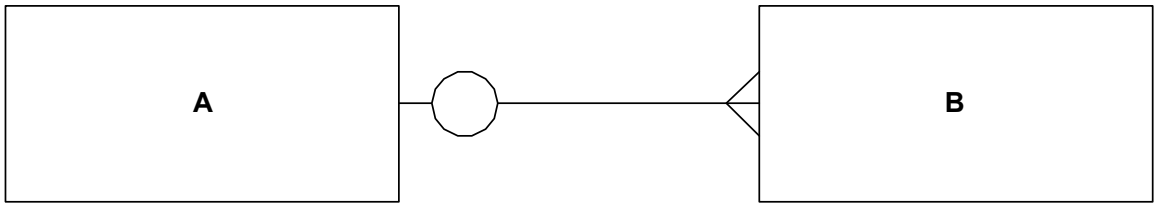
In summary, the diagram rules are:

ENTITY

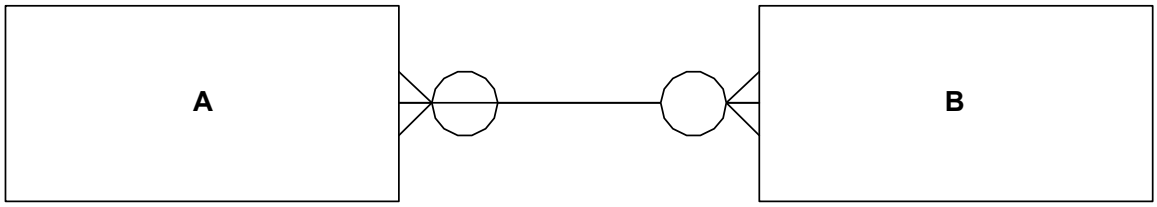
1:1 RELATIONSHIP

1:M RELATIONSHIP

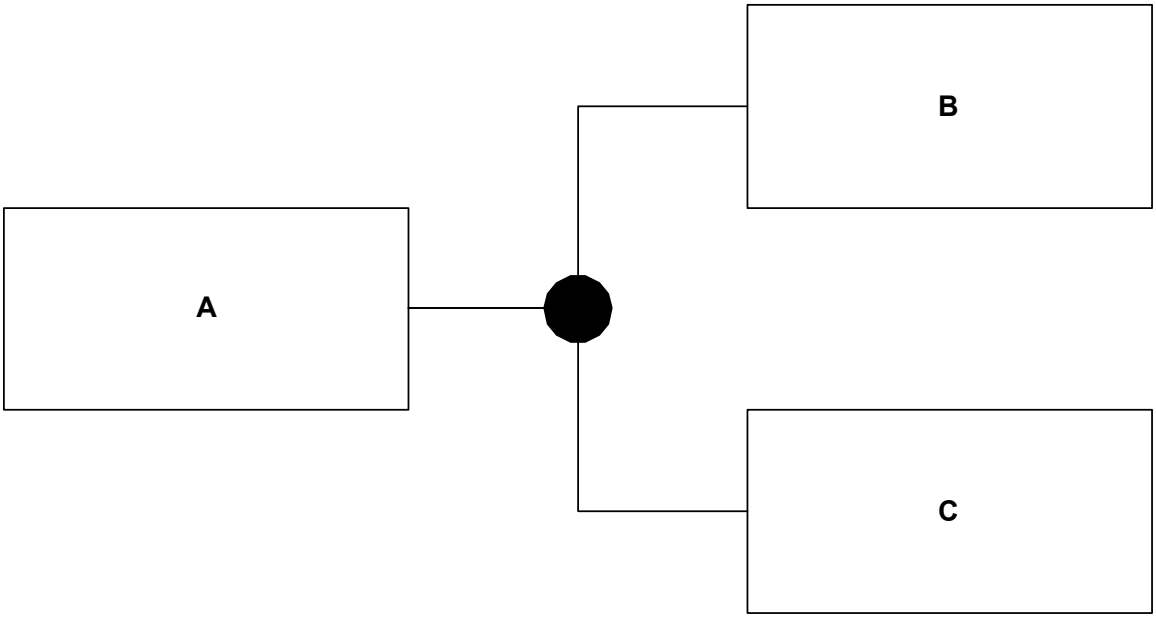
M:N RELATIONSHIP



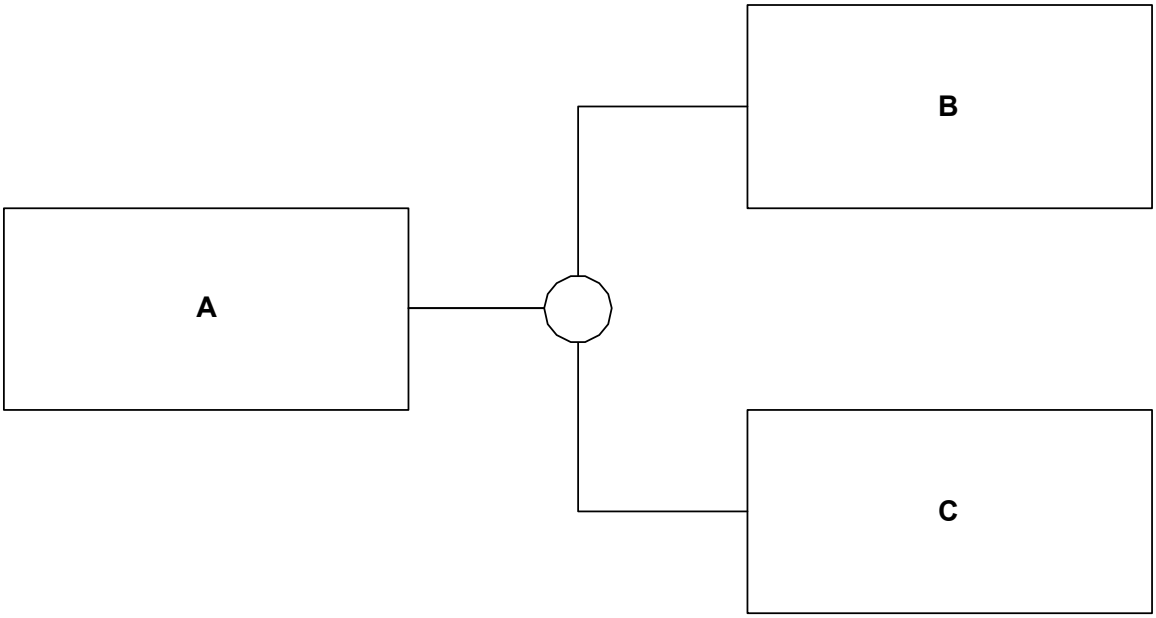
PARTLY OPTIONAL 1:M RELATIONSHIP



FULLY OPTIONAL 1:M RELATIONSHIP



EXCLUSIVE RELATIONSHIP



INCLUSIVE RELATIONSHIP

14.3.3 Entity Types

An entity type can be defined only if each of its entities is uniquely identifiable.

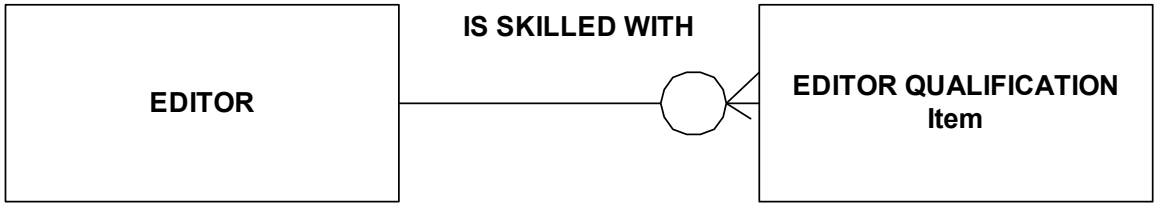
An entity type must have at least one relevant attribute type.

An entity type participates in relationship types (i.e., it is not isolated).

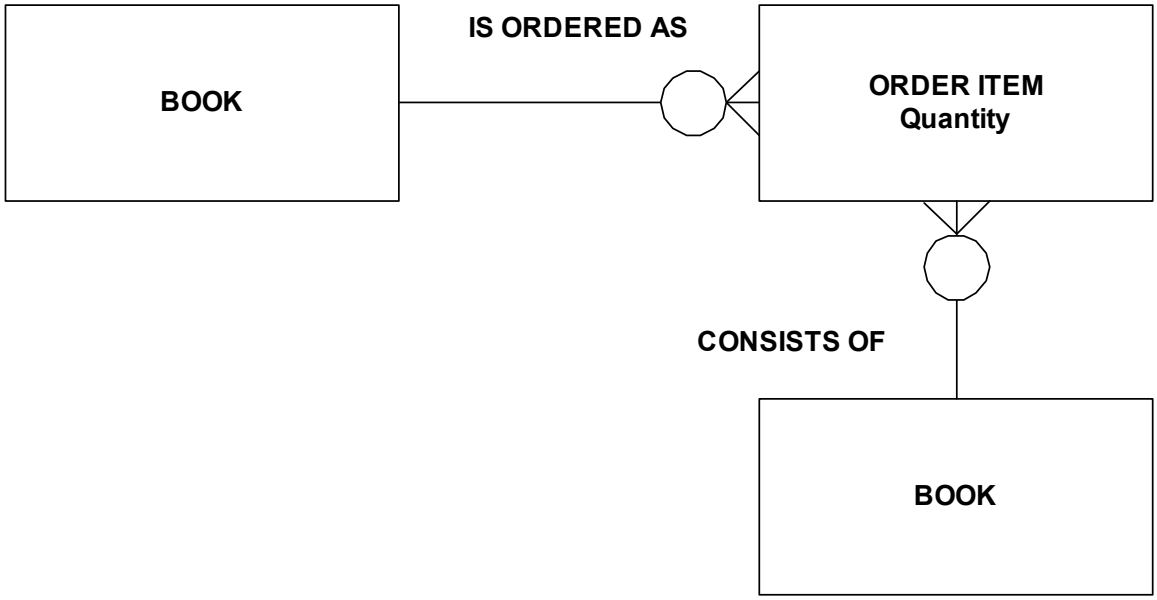
An entity type is classified as being in only one subject area.

An entity type should not normally be solitary (i.e., have only one entity).

An entity type with a "solitary" attribute type is usually not useful. If that attribute type is the only identifier, then it is describing itself. If there is one foreign identifier, assign the attribute type to the identifier's entity type. In this new position, the attribute type may now be multi-valued. For example, the entity type EDITOR QUALIFICATION has only one attribute type (i.e., Item, whose value might be master's degree) and a many-to-one relationship with EDITOR. To resolve this condition, the entity type EDITOR QUALIFICATION may be discarded and a multi-valued attribute type (i.e., Editor Qualification Item) is established for the entity type EDITOR.



When there is more than one entity type involved via a foreign identifier, as happens with an entity type associating other entity types, the attribute value cannot describe the other entities. The solitary attribute type and its entity type may then remain separate. For example, the entity type ORDER ITEM may have only one attribute type (i.e., Order Item Quantity), but it has two relevant foreign identifiers in the ORDER and BOOK entity types.



14.3.4 Rules About Relationship Types

- Relationships are defined only between one or two entity types, except where inclusive relationships have been explicitly depicted.

- If a relationship exists between entities of only one entity type (i.e., it is an involuted relationship), then the different pairings are with different entities.

- A relationship type does not have attributes. If any appear to have been discovered, then an entity type is defined for them.

- If an entity can exist without ever becoming involved in a particular membership, then that membership must be optional for its entity type.

- If an entity can ever be associated with more than one entity under a given relationship, then the cardinality of the latter entity type is many.

14.3.5 Rules About Entity Analysis Objects

- Each entity analysis object must be represented by only one analysis concept so that there is no overlap or redundancy. For example, "Address" may not be both an entity type and an attribute type of an entity type where it represents the same things or values.

- As understanding improves, an object may be reclassified. For example, the relationship of LECTURER to SUBJECT ("can teach") may reveal the attribute types "Experience Length" and "Skill Level" and may therefore be replaced by the entity type EXPERIENCE and two new relationship types.

- The name of one entity analysis object may not be used as the name of any other entity analysis object.

14.4 Entity Relationship Modelling Techniques

14.4.1 Entity Types

Entity Type Names

Use a singular noun that, if possible, is well known to the users.

Two or more names for the same entity type are often found in different parts of a business; these may be synonyms. If a choice of synonyms exists, use the least ambiguous. Keep this standard throughout the analysis, and document the synonyms.

Similar problems occur with homonyms (i.e., where the same name is used for two or more entity types). If this occurs, unambiguous names must be chosen for the entity types, and the homonyms must be documented as synonyms of the affected entity types.

Entity Type Definition

Define what the entities are, not how they are represented as data. Include:

- A definition of scope, or what the entities are, preferably using well-defined and recognisable terms
- A qualification, or method of distinguishing entities, using terms known from names or definitions of other business objects
- It is useful to describe any excluded factors in a definition
- It is extremely useful to give examples of entities of the type

Solitary Entity Types

If the definition implies that the entity type is solitary (i.e., there is only one entity of its type, such as the enterprise itself), the analysis may be incorrect. Ask if there will always be only one entity (e.g., FACTORY and COMPUTER CENTRE may initially be solitary, but there may be plans to build more).

If this entity is relevant, seek another entity type (e.g., ORGANISATIONAL UNIT) whose definition and relationships may be extended to encompass the entity.

14.4.2 Naming Relationship Types

Syntax: Entity Type VERB Entity Type

Use a verb relating the two entity types in a meaningful clause (e.g., CUSTOMER places ORDER). Use an active verb, one used in the name of the process that associates the entity types. Construct an inverse name relating the two entity types in reverse order (e.g., ORDER placed by CUSTOMER). The verb in the inverse name is normally passive. Relationship memberships should be named in the same way.

Name both relationship memberships because it is useful to be able to read a relationship from both memberships, both relationship memberships should be named.

Avoid time implications. Allow the relationship name to include past and present (e.g., PERSON sometimes is "employed by" ORGANISATIONAL UNIT). Specific time implications (e.g., PERSON "previously employed by" ORGANISATIONAL UNIT) are used only where selectivity is meaningful.

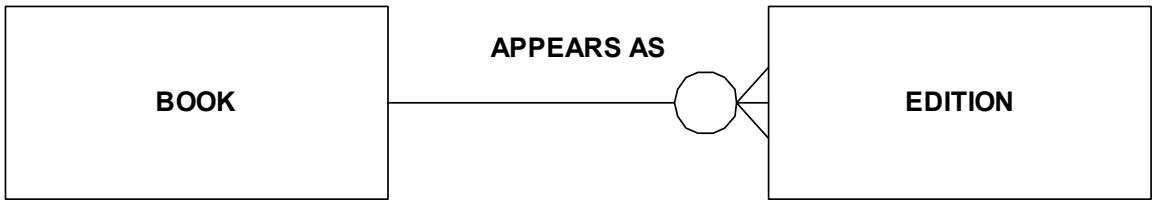
14.4.3 Reading a Relationship Type

A relationship type can be read in terms of either of its two relationship memberships. It can also be read to include the optionality and cardinality of the chosen membership. The optionality determines whether the relationship may or

must hold for a membership. The cardinality determines whether the relationship is to exactly one or to one or more entities for a membership.

Relationship Element Membership	**Description**
<Entity Type A>	This is the entity type from whose viewpoint the relationship membership is taken.
Always/Sometimes	This is determined by the optionality of the membership.
<Relationship Name>	This is the verb of relationship.
One/One or More	This is determined by the cardinality of the relationship.
<Entity Type B>	This is the other entity in the relationship.

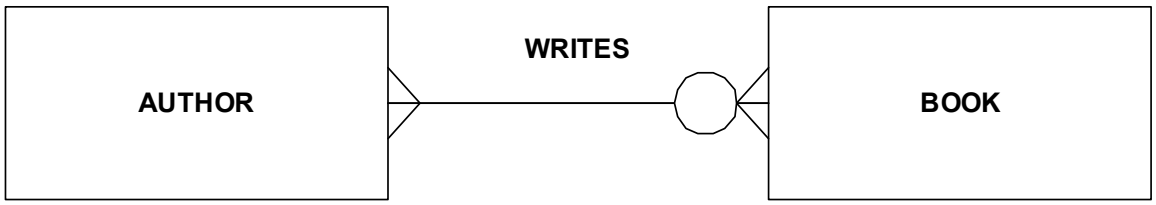
The following relationship:



can be read as:

- A BOOK sometime appears as one or more EDITIONs.
- An EDITION always is of one BOOK.

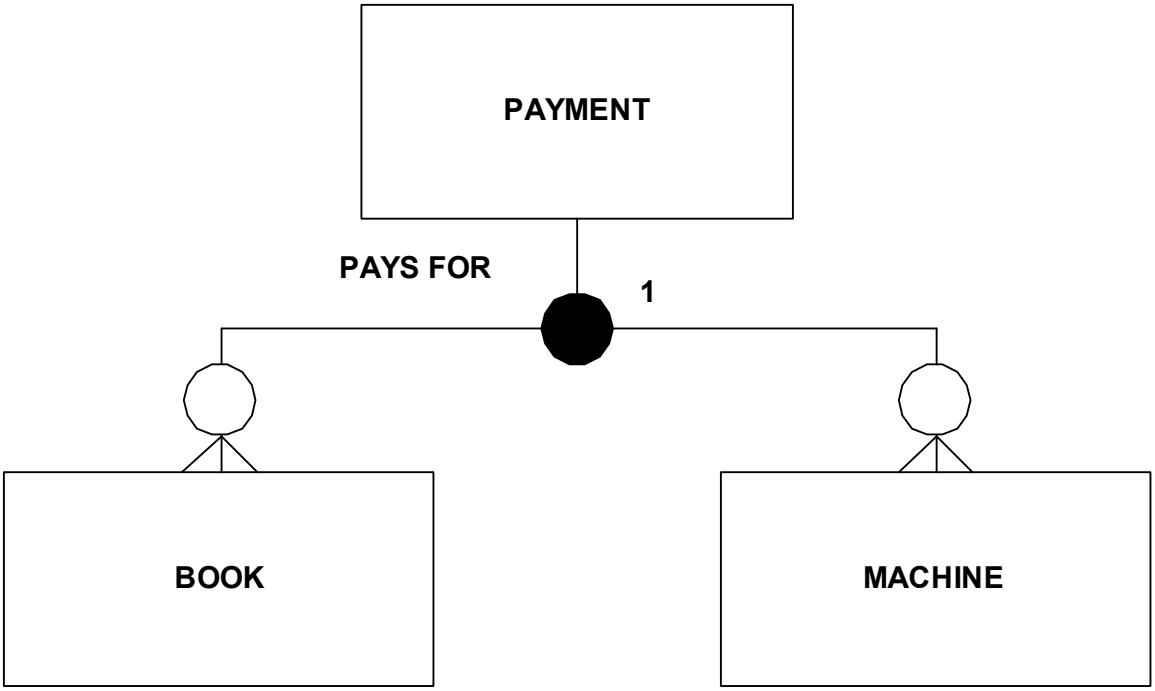
The following relationship:



can be read as:

- An AUTHOR sometimes writes one or more BOOK.
- A BOOK always is written by one or more AUTHORs.

The following relationship:



can be read as:

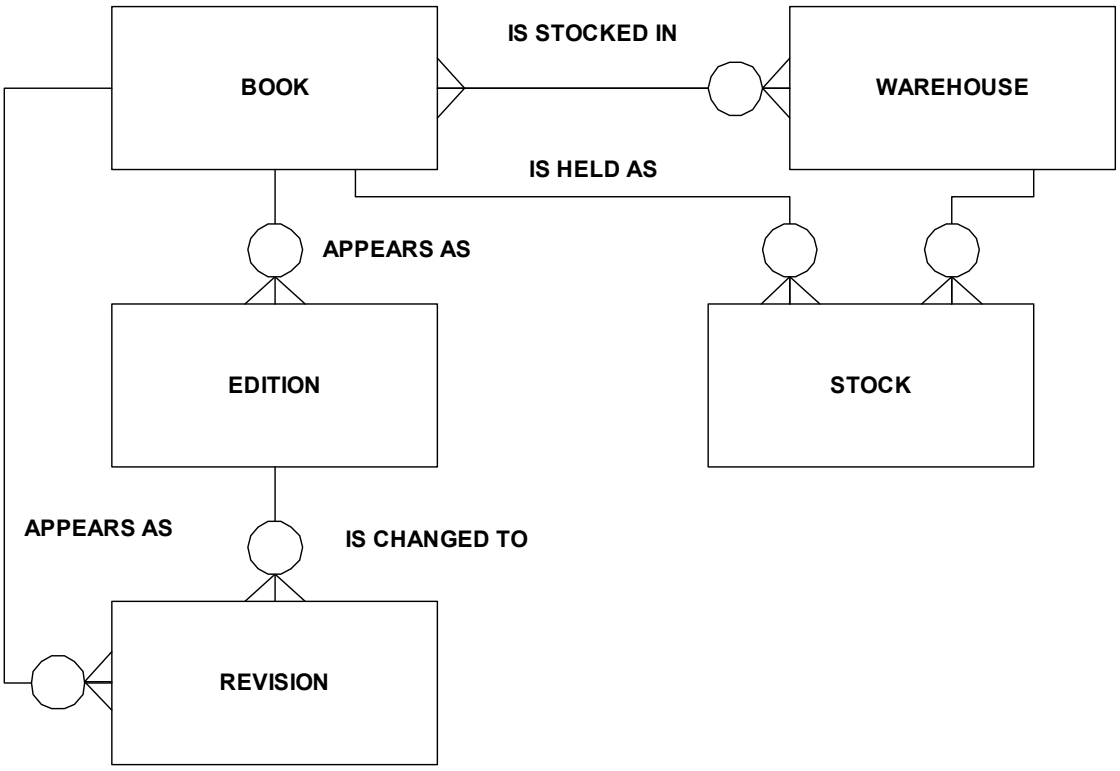
- Each PAYMENT always pays for one or more BOOKs or pays for one or more MACHINEs.

14.4.4 Redundant Relationships

It is possible to think of a relationship between a given entity type and almost every other entity type in a business area. Many of these will associate entity types that are also indirectly associated through other entity types. For example, BOOK stocked in WAREHOUSE is also represented by two other relationships: BOOK held as STOCK, and STOCK held in WAREHOUSE.

Each grouping of a specific book in warehouses can be derived from the BOOK held as STOCK groupings, because a book is stocked only in warehouses where it is allocated to be held as stock. Therefore, the relationship BOOK stocked in WAREHOUSE gives no additional information and is redundant.

A relationship is redundant only if it is so at all times. That is, all the pairings it implies can be derived from other relationships. When you decide whether or not a relationship is redundant, you must consider the optionality of the relationships and the defined meaning of each relationship.



In this example, REVISION is always of an EDITION, which is always of a BOOK. Because a REVISION must always be of a BOOK, the relationship between REVISION and BOOK is redundant. However, if a book could be revised separately from an edition's being printed (e.g., before the book is formally published), then the relationship between REVISION and EDITION is fully optional. Thus the relationship between REVISION and BOOK is no longer redundant because there will be occasions when there will be no relationship through an edition to a book.

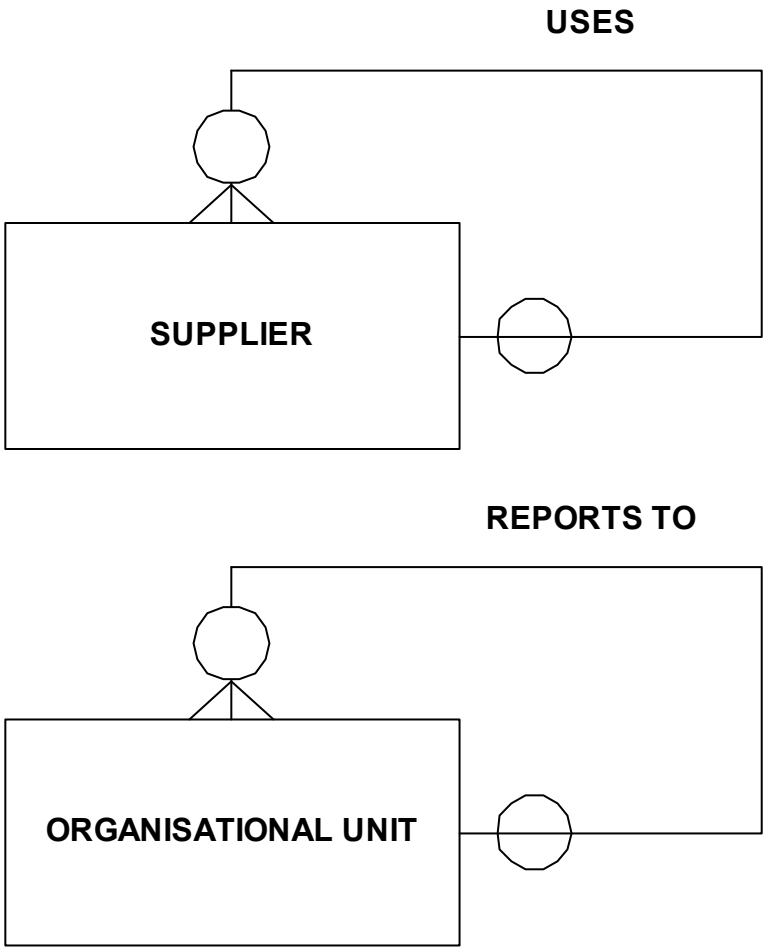
14.4.5 Types of Relationship Types

Fully mandatory one-to-one and many-to-many relationships almost never occur. Treat these cases with caution and suspicion. It is likely that at least one of the memberships is optional. Fully mandatory one-to-one relationships suggest that the two entity types are the same thing.

14.4.6 Modelling Organisational Structure

You often need to model the structure of the organisation you are analysing, and the module must be as resilient as possible to change. Organisations may change their structure frequently, however, so it is best to model with a generalisation. One way of achieving this is by having a single box represent the organisation, with an involuted relationship representing the structure. If the organisation is structured hierarchically, there will be a one-to-many involution; if it is structured as a matrix, there will be a many-to-many involution.

Other organisations are also modelled (e.g., customers with a head office and subsidiary companies). These are modelled with involutions, as well.



14.4.7 Refinements to the Entity Relationship Model

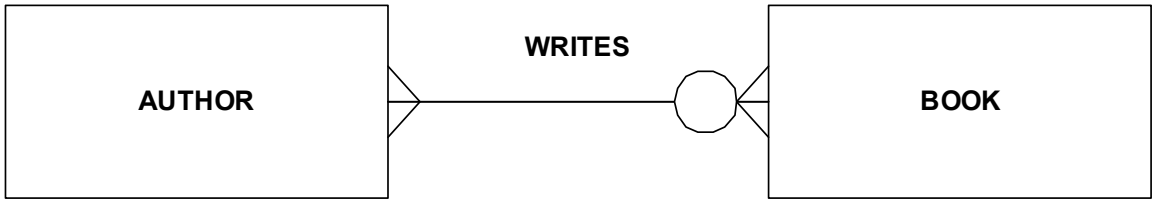
Isolated entity types, (i.e., entity types with no identified relationships) are very rare. If you find one, consider if there have been any relationships missed or if that entity type is really useful. Can it be useful if it has no firm association with any other information relevant to the enterprise?

Consider one-to-one relationships to see if the related entity types are the same thing. This is likely if you find a mandatory, one-to-one relationship. Similarly, if you find partly optional one-to-one relationships, consider whether the entity types are the same over time (e.g., PROSPECT becomes CUSTOMER in which case, treat PROSPECT as a state in the life of CUSTOMER) or if one is a subtype of the other. An alternative may be that over-time, the relationship is one-to-many or many-to-many.

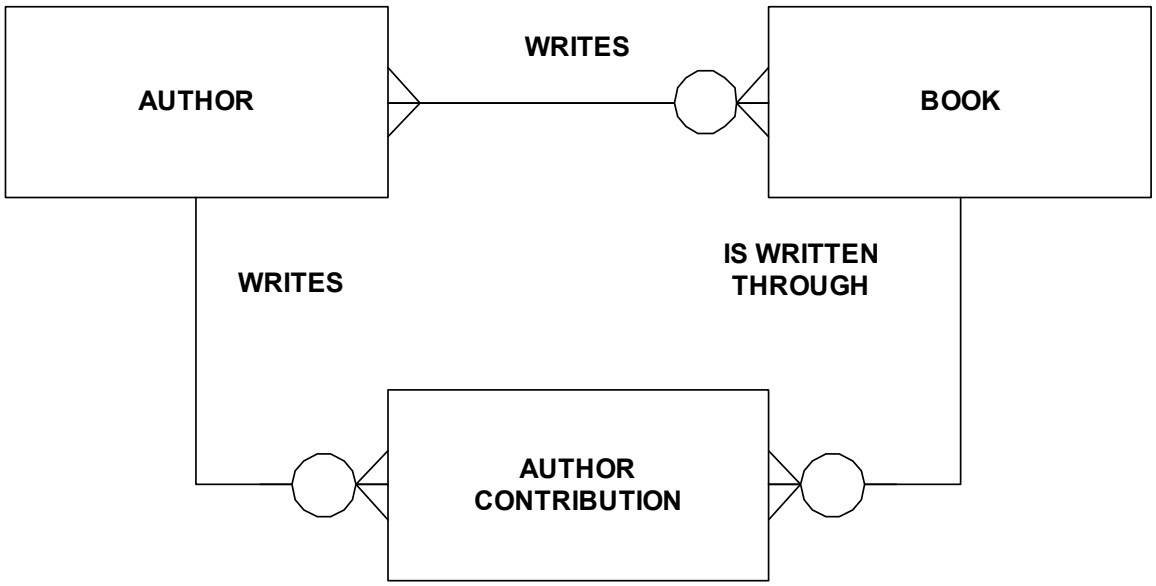
If you find many-to-many relationships, hypothesise a third entity type that participates in two many-to-one relationships with the original entity types. If

you find attributes for this third entity type, keep them and define them. If none can be found, there is a true many-to-many relationship.

For example, the original relationship:



can be extended as:



You can find attributes for this entity type of "Contribution Amount," "Data" and so on; therefore, it is a valid entity type.

The original relationship type (i.e., AUTHOR writes BOOK) is now redundant and can be removed. All the information can be gained through the use of AUTHOR CONTRIBUTION.

A common example of a many-to-many relationship is found in a bill of materials structure in which a PART may be "made from" many other PARTs.

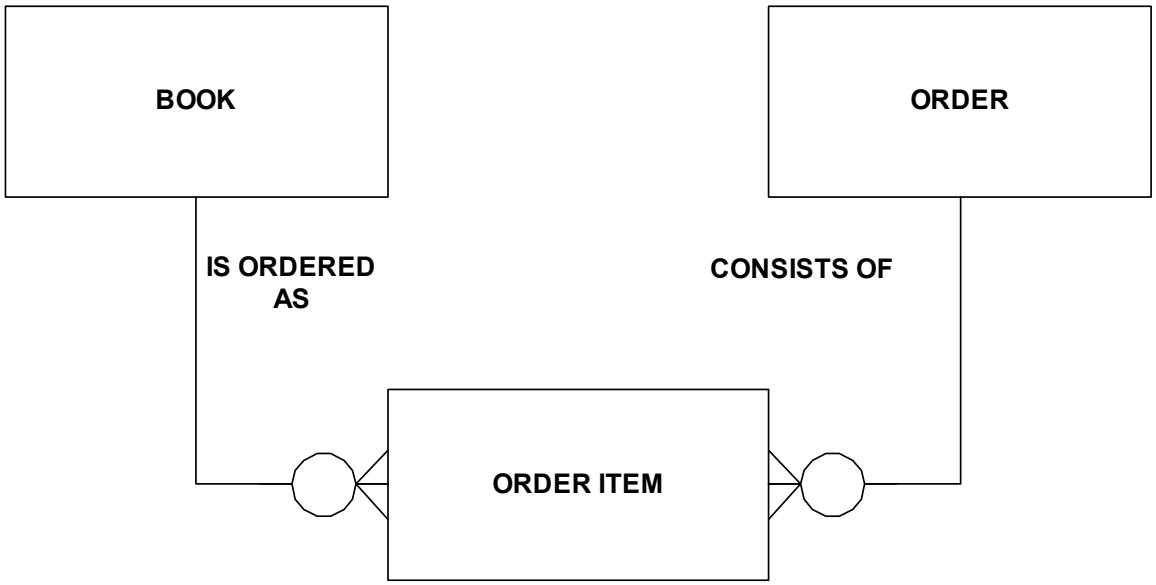
14.4.8 Drawing Entity Relationship Diagrams

Drawing a diagram is a good way of identifying possible additional relationships. The following guidelines will help create a readable document.

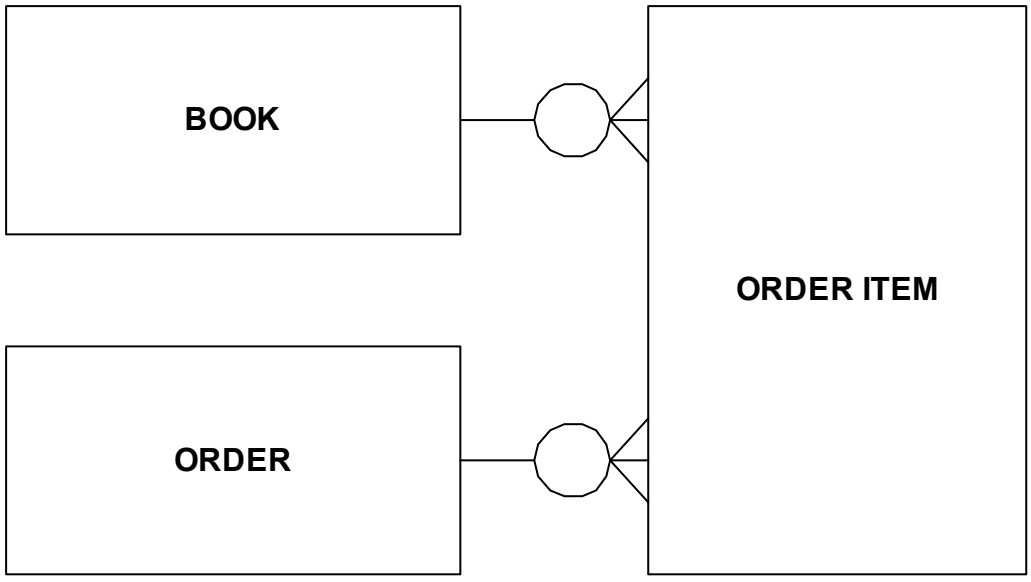
- Place subject entity types on the top of the diagram.
- Place plural entity types below a single entity type in a one-to-many relationship.

- Place entity types participating in one-to-one and many-to-many relationships alongside each other.
- Group closely related entity types when possible. Try to keep the length of relationship lines as short as possible. Also try to minimise the number of changes of direction in a single line.
- Lay out the diagram with minimal line crossing.
- Show the most relevant relationship name. One name must always be shown.

Another way to draw diagrams by hand is to use different-sized entity type boxes instead of bending a line. For example, instead of:



the following can be used:



14.4.9 Presenting an Entity Relationship Diagram

General Approach

An entire entity relationship diagram should not be presented to an audience; it should be presented bit by bit and should build up to the complete diagram. The presentation should start with subject entity types.

Initially, a single entity type should be presented, its definition and significance described, and the concept of entity type briefly explained. Next, one of its relationships should be shown, the concept of relationship, optionality, and cardinality explained, and the related entity type and its definition described.

This continues, with more and more concepts being shown at a time, until the complete diagram is displayed. The best method of achieving this is to have a series of overhead projector slides prepared, each with more on it than on the slide before.

Drawings for Working Sessions

If the session is for feedback, as opposed to being a formal presentation of deliverables, it is best to avoid carefully drawn diagrams because people may be reluctant to change them.

Audience Involvement

Try to get your audience involved in the presentation and questioning of every property on the diagram.

Presentation of Entity Types

When presenting an entity type, give its name, definition, and some examples of entities (this will necessitate a discussion of some of the main attributes).

Level of Detail

It is possible to simplify the diagram presented (e.g., by not showing optionality or relationship type names), but it is usually best to present complete detail using the above guidelines.

15. Appendix 3 - SWOT Analysis

15.1 Introduction

SWOT analysis is a useful tool for understanding and decision-making for all sorts of situations in business and organisations. SWOT is an acronym for Strengths, Weaknesses, Opportunities, Threats.

The SWOT analysis headings provide a good framework for reviewing strategy, position and direction of a company or business proposition, or any idea. Completing a SWOT analysis is very simple, and is a good subject for workshop sessions. SWOT analysis also works well in brainstorming meetings. Use SWOT analysis for business planning, strategic planning, competitor evaluation, marketing, business and product development and research reports.

A SWOT analysis is a subjective assessment of data that is organised by the SWOT format into a logical order that helps understanding, presentation, discussion and decision-making. The four dimensions are a useful extension of a basic two heading list of pro's and con's.

SWOT analysis can be used for all sorts of decision-making, and the SWOT template enables proactive thinking, rather than relying on habitual or instinctive reactions.

The SWOT analysis template is normally presented as a grid, comprising four sections, one for each of the SWOT headings: Strengths, Weaknesses, Opportunities, and Threats.

The SWOT template below includes sample questions, whose answers are inserted into the relevant section of the SWOT grid. The questions are examples, or discussion points, and obviously can be altered depending on the subject of the SWOT analysis. Note that many of the SWOT questions are also talking points for other headings - use them as you find most helpful, and make up your own to suit the issue being analysed. It is important to clearly identify the subject of a SWOT analysis, because a SWOT analysis is a perspective of one thing, be it a company, a product, a proposition, and idea, a method, or option, etc.

15.2 SWOT Template

Strengths

- Advantages of proposition?
- Capabilities?
- Competitive advantages?
- USP's (unique selling points)?
- Resources, Assets, People?
- Experience, knowledge, data?
- Financial reserves, likely returns?

- Marketing - reach, distribution, awareness?
- Innovative aspects?
- Location and geographical?
- Price, value, quality?
- Accreditations, qualifications, certifications?
- Processes, systems, IT, communications?
- Cultural, attitudinal, behavioural?
- Management cover, succession?

Weaknesses

- Disadvantages of proposition?
- Gaps in capabilities?
- Lack of competitive strength?
- Reputation, presence and reach?
- Financials?
- Own known vulnerabilities?
- Timescales, deadlines and pressures?
- Cashflow, start-up cash-drain?
- Continuity, supply chain robustness?
- Effects on core activities, distraction?
- Reliability of data, plan predictability?
- Morale, commitment, leadership?
- Accreditations, etc?
- Processes and systems, etc?
- Management cover, succession?

Opportunities

- Market developments?
- Competitors' vulnerabilities?
- Industry or lifestyle trends?
- Technology development and innovation?
- Global influences?
- New markets, vertical, horizontal?
- Niche target markets?
- Geographical, export, import?
- New USP's?
- Tactics - surprise, major contracts, etc?
- Business and product development?
- Information and research?
- Partnerships, agencies, distribution?
- Volumes, production, economies?
- Seasonal, weather, fashion influences?

Threats

- Political effects?
- Legislative effects?
- Environmental effects?
- IT developments?

- Competitor intentions - various?
- Market demand?
- New technologies, services, ideas?
- Vital contracts and partners?
- Sustaining internal capabilities?
- Obstacles faced?
- Insurmountable weaknesses?
- Loss of key staff?
- Sustainable financial backing?
- Economy - home, abroad?
- Seasonality, weather effects?

16. Appendix 4 – Change, Issue and Problem Management

16.1 Introduction

Change management is a process that introduces system changes, problems or emergency fixes into an organisation in a controlled and planned environment that includes managing and reporting activities.

A ***change*** occurs when a work product matches the baseline specification but something else is wanted. This is true even if the specification is obviously wrong.

A ***problem*** occurs when a work product departs from what is prescribed in the baseline specification.

It is the process of controlling changes to the system to ensure that only authorised changes are applied. It involves requesting changes, analysing the impact and feasibility of the changes, approving the changes before implementation, and tracking the changes through completion. Key features of change management are:

- A vehicle for proposing and tracking changes, commonly called a *change request*, and a vehicle for reporting and tracking problems, commonly called a *problem report.* In Catalyst, we use a common source document, the Change Request/Problem Report, for both purposes.

- An *authorising agent*, representing major stakeholders, to approve or reject proposed changes. Examples of authorising agents include the project sponsor, the project manager, or a *change control board (CCB)*.

- An administrative function to maintain information about change requests and ensure proper operation of the change management process. In Catalyst, this function is called the *change control office (CCO)*.

- A *change log*, maintained by the CCO to track change requests, and a *problem log* to track problem reports. In Catalyst, we use a Change/Problem Log to serve both purposes.

A successful change management process:

- Enables change decisions to be based on knowledge of the full impact of the change. This allows clients and management to understand both the cost and the benefit of a change and allows you to focus on changes that are necessary or that offer significant benefit.

- Ensures that all stakeholder interests are considered before changes are approved.

- Ensures that only understood and authorised changes are made, thus increasing the quality and maintainability of the system throughout its operational life.

A successful change management process achieves the following objectives:

- Enable change decisions that are based on knowledge of the full impact of the change. This allows clients and management to understand both the cost and benefit of a change and allows you to focus on those changes that are necessary or that offer significant benefit.

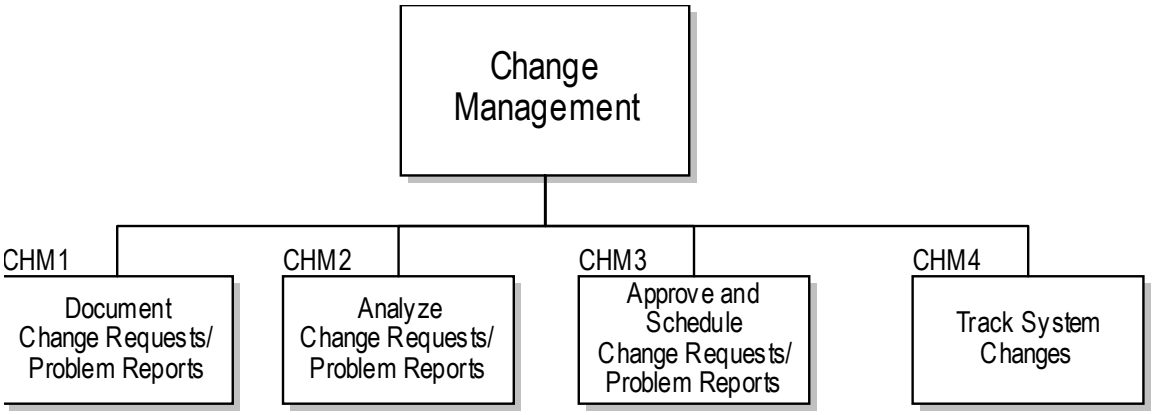
- Ensure that all stakeholder interests are considered before approval.

- Ensure that only understood and authorised changes are made, thus increasing the maintainability of the product throughout its operational life.

- Reduce the frequency and risk of change by bundling changes into releases that follow a controlled development, integration, and deployment process.

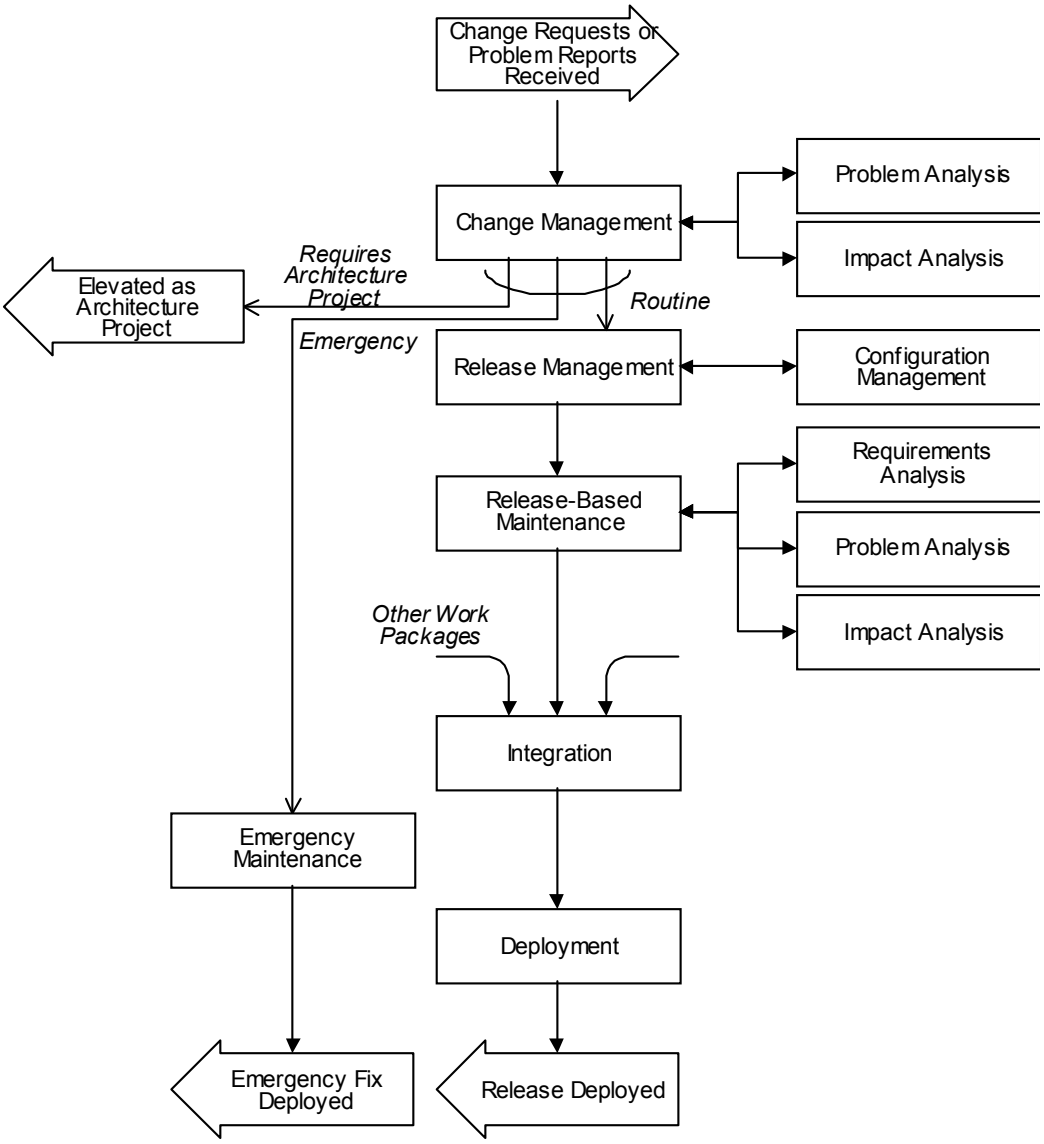
Controlling change is not easy, so change management is not a trivial concept. It is imperative to gain a real understanding of the overall degree of change. It is equally important to be able to characterise and classify individual changes.

The activities involved in change management are:

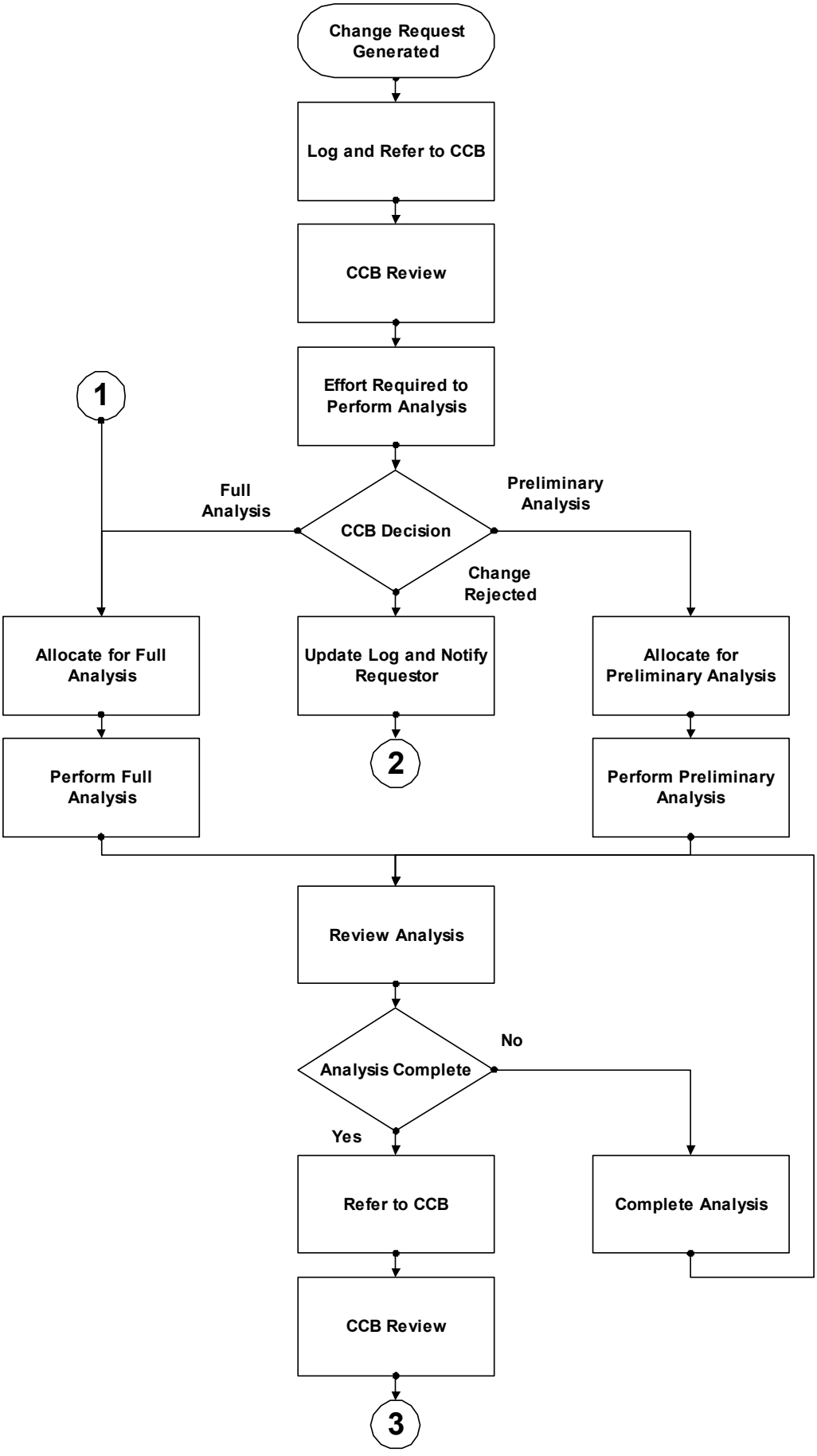


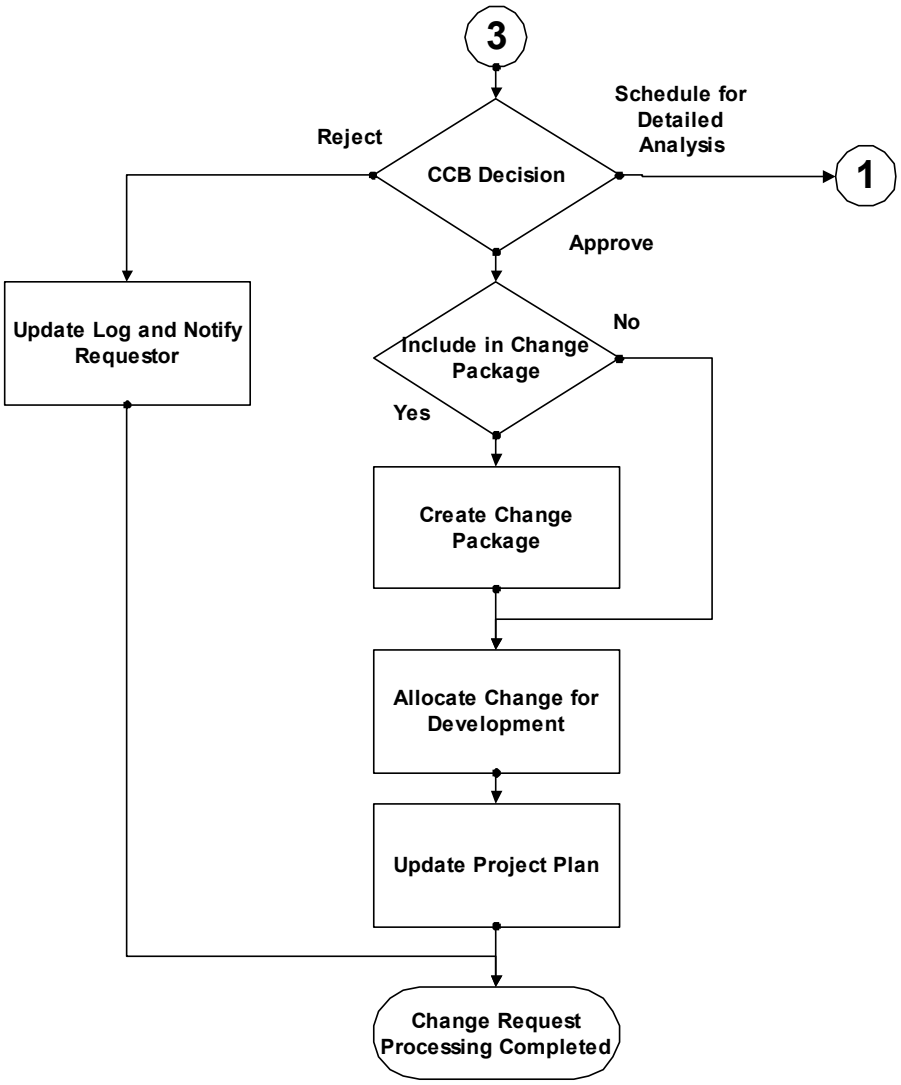
16.2 Change Management Process

Schematically, the overall change management process is:



The specific steps involved in evaluating and processing change requests are:


Change Request Generated
Log and Refer to CCB
CCB Review
Effort Required to Perform Analysis
1
Full Analysis
CCB Decision
Preliminary Analysis
Change Rejected
Allocate for Full Analysis
Update Log and Notify Requestor
Allocate for Preliminary Analysis
2
Perform Full Analysis
Perform Preliminary Analysis
Review Analysis
No
Analysis Complete
Yes
Refer to CCB
Complete Analysis
CCB Review
3



These steps are:

- A CR is generated. This is logged. The CR is flagged for review by the CCB. The Project Manager will estimate the effort required and cost of analysing the CR.

- The CCB reviews the change request. The CCB can decide to reject the change or schedule the change for either preliminary or full analysis. No CR can be accepted without an analysis that will detail its impact on the project.

- If the CR is rejected, the originator will be notified and the log will be updated.

- If the CR is scheduled for analysis, either preliminary or full, the analysis will be scheduled.

- The analysis will be performed. It will be reviewed for completeness. If it is incomplete, the analysis will be completed.

- The analysis will be referred to the CCB for review. The CCB will review the analysis. The CCB can decide to reject the change, schedule it for detailed analysis or accept the change.

- If the CR is rejected, the originator will be notified and the log will be updated.

- If the CR is scheduled for detailed analysis the analysis will be scheduled.

- If the CR is accepted, it may be included as part of a change package or schedule for separate implementation.

- The project plan will be updated.

16.3 Change and Problem Requests

During the life of a project system change requests/ problem reports (CR/PR) will arise.

It is important to distinguish between problems and changes when they are identified during all phases of the project. The CR/PR form is used to capture and document these changes. The criteria are simple:

- **Problem.** A problem occurs when a program's behaviour departs from that prescribed in the baseline specification.

- **Change.** A change occurs when a program behaves according to the baseline specification but *something else is wanted*. This is true even if the specification is obviously wrong.

A CR/PR can arise as a result of:

- New Business Requirements
- New System Requirements
- Emergency Fixes
- Production Problems

Change requests vary in size, scope and risk. The process to manage all changes effectively and efficiently depends upon the criticality, commitments and level of sponsorship that also determines the level of CCB involvement.

An informative CR/PR will have:

- Supporting material sufficient to recreate a problem that has been identified, if the problem is a recurring situation
- Will avoid describing a given solution in favour of describing the problem symptoms or desired opportunity
- Will be as complete as possible with information such as maintenance category, severity, required date, effort estimate, etc.

A change should have a good business case, not just a business reason. A CR/PR should provide the CCB with enough information to make an objective business decision.

The CR will need to include an explanation of the problem and a subsequent recommended solution, both from a business perspective and for the system:

- Business Process problem/opportunity
- Business Process recommended resolution
- Business System problem/opportunity
- Business System recommended resolution

The CR will need to include business requirements. They may be initially supplied or developed through project development team work shops or discussions (e.g. prototyping). Recommended resolutions for business and system problems can be provided, if they exist, but should not be the main emphasis of the initial request information. Solutions should be developed from an understanding of any underlying conditions to ensure the right problems are being resolved.

Business Requirements should include:

- Business background - A high level overview of the overall problem or new requirement.
- Scope - Summary of how the client views how the system(s) needs to be modified to support the requested change.
- Requirements - Facts, figures, rules, regulations, processing flows, other effected systems, etc.
- Impact - Known impacted files, transactions, screens, reports, products, etc. with marked up examples.
- Required production date.

Additional information may also be provided if available. This data serves as the basis for workshops or discussions in the early development phases and later in design. At these sessions a further definition of the request as well as additional details can be gathered if required.

The primary event triggering the change management process is the receipt of a change request or problem report.

- When authorisation to establish change management is received, this triggers a process to establish the mechanisms of change management, resulting in an operational change management process.

- When a Change Request/Problem Report is received, this triggers a process to document; analyse; approve, reject, or defer; and if approved schedule the change.

- When the status of a Change Request/Problem Report changes, this triggers a process to update the status in the log.

- When a report is requested or it is time to produce periodic reports, this triggers a process to generate reports describing the status of particular changes/problems and management reports showing progress in addressing changes/problems.

A change package is a collection of change requests and problem reports that have been aggregated for collective analysis, approval, development, and deployment. A change package also becomes a collection of change items, which are the translation of the change requests and problem reports into specific action items.

16.4 Evaluating Change Requests

Use the following matrix to evaluate the scope of the change request and to derive a score.

The score from each column is totalled to obtain a total score, representing the Domains of Change Index. The maximum score is 24 where a Change rates as Radical within each Domain of Change.

Degree of Change	Business Process	Organisation	Location	Data	Technology	Application
Minor Impact (1)	Supports existing processes	Different procedures	Impacts to 1 or a few sites only.	Same entities, new attributes	Same products, additional uses	Minor changes to existing applications
Moderate Impact (2)	Revised activities existing processes	Different job content	Impacts to 30%-30% of sites	New entities	Same products; increased capacity, workload	Enhancements to existing application
Major Impact (3)	Revised process	Different jobs and org. structure	Impacts to 40%-70% of sites.	New data structure	New products	New application (Subsytem)
Radical Change (4)	New process	Different culture	Impacts most or all sites.	New data types (image, voice, objects)	New technology types (imaging)	New application architecture (client/ server)

Problem analysis is the activity of investigating a problem, determining the root cause, and proposing one or more possible solutions.

Impact analysis is the activity of identifying what needs to be modified or created to accomplish a change or identifying the potential consequences of a change.

16.5 Analysing Change Requests

Analysis of the CR is performed in order to direct demands through the appropriate approval process and resolve issues efficiently. The central clearinghouse, with support from the CR/PR requester (and/or primary business unit for the CR/PR), the Change Control Board, and the project development team (or configuration management team) may all perform an initial assessment of CR/PRs to get them allocated to an appropriate release and assigned to correct project managers.

This activity is applicable to any issues that cross business units. Problems, research, or issues that cross business units involve the central clearinghouse and require CCB coordination.

CR/PRs that don't cross business units also need to be analysed, but the activities are more specific. Examples of CR/PRs that don't cross business units include defects uncovered during IT product reviews or test discrepancies uncovered during final application and integration activities.

Before beginning this activity, the following conditions should exist:

- A CR/PR exists as an issue, problem, change, or request for research to be authorised and assigned to a project manager.

After completing this activity, the following conditions should exist:

- The CR/PR has been analysed and completed by the requester to fill in all the appropriate information. The request has also been reviewed by the CCB for authorisation and assignment. It may be put on hold to be reconsidered as additional information or opportunities are identified.

- The CR/PR is updated with effort estimates and actuals and components affected as the issue resolution progresses.

The Change Control Board authorises requests according to budget allocations.

The Change Control Board is responsible for authorising the CR/PR, allocating it to a release, and assigning it for analysis. An optional Planning Workshop can assist with additional information needed to authorise a request or determine its true scope for proper assignment.

The Change Control Board may request that project management provide additional information, and project management decides if a planning workshop is the best mechanism for gathering that information.

The CCB reviews new and active change requests.

16.5.1 Project Manger Responsibilities

The project manager is responsible for identifying and subdividing change packages into maintenance work packages and, if needed, calling for a Requirements Workshop to build specifications.

A requirements workshop can be used to:

- Review the detailed Project Schedule
- Review the tasks for the next phase
- Finalise the Risk Analysis
- Confirm costs and benefits
- Agree to project control procedure

The Project Manager selects the appropriate individual or group of individuals to review the CR/PR and project planning begins or initiation stage. The initiation stage is a project phase in its own right. This stage is a reminder that you need to carry out project initiation before starting development activities.

An agreement between the business unit or requester of an CR/PR and the project manager is established during development. It must be understood by all business units (i.e. stakeholders) that only tentative schedules and costs are needed to assign an CR/PR to a project manager, and requirements will not be fully defined at this stage.

Requirements definition can occur during simple conversation verifying the CR/PRs intent, through formal meetings, interviews, and questions, or via workshops. The mechanism, to ensure that a clearly stated, verifiable, and testable business requirement is documented, depends on the nature of the maintenance change and project approach and organisation. Documented business requirements are needed for all maintenance activities except emergency repairs. If prototyping is used to define requirements they will still require some documentation, since prototyping evolves the requirements but does not eliminate the need to document them.

16.5.2 Consolidating CR/PR into Change Package(s)

This is performed in order prepare the CR/PR for assignment to a project development team. This activity is applicable to any CR/PR that is being processed through the development phase.

Before beginning this activity, the following conditions should exist:

- Several CR/PRs exists as either issues, problems, changes, defects, discrepancies, or requests for research which need to be assigned to a project development team either for further estimates or to begin development activities.

After completing this activity, the following conditions should exist:

- The CR/PRs are consolidated into one or more change packages for assignment to project managers.

The criteria for consolidating CR/PRs into change packages is dependent on enterprise goals, structure, and resource limitations. Questions that are dependent on enterprise areas and resources include:

- What volume can be handled in parallel?
- What volume can a particular resource handle at one time?
- Are different problem categories routed to different organisations?
- Who has access to information needed to resolve an issue?
- Is quality or productivity to be optimised? (or a combination?)
- What finances are allocated and available for an issue?
- How are the levels of the Change Control Board(s) organised?
- What system interfaces and communication structures exist?
- How easily are projects initiated and how many people are available?
- How involved will the customer/client be with issues?

Benchmarks for the demand management process will provide statistics for the CCB to establish improvement goals, and consolidated change packages should be designed to support and optimise these goals. For example, mass corrections can be handled as one large change package across multiple systems or dispersed into smaller change packages with other enhancements for each system or application.

Note: The first method has higher visibility and can raise throughput goals; the second can reduce testing effort and improve productivity (i.e. delivered value per given time frame). These types of decisions are based upon *what is important to measure* and *what is important to optimise* for the enterprise or organisation.

Factors to consolidate CR/PRs into Change Packages include the following:

- Requests impact the same application program, system, or business unit
- Requests allocated to the same planned release
- Requests already assigned to a budget account, or a specific corporate initiative or Program area
- Requests require the same test cases (e.g. the same input transactions)
- Requests assigned to the same resource
- Requests involve the same change to many programs
- Requests involve the same category of change
- Requests originate from the same source

16.6 Change Requests/Problem Reports and Change/Problem Log

16.6.1 Change Request/Problem Report

Each Change Request/Problem Report is used to record, communicate, and manage a problem involving components of a configuration item or a change to the project baselines. The Change Request/Problem Report may be documented on a form or captured on-line. It should record the following information:

- **Identification**

    - **Type.** This indicates whether the item is a change request or a problem report.
    - **Priority.** This describes how urgent it is to act on the problem or change. Possible entries include: critical (emergency), high, medium, low, and defer.
    - **Identification number.** This is a unique number to identify and control this report.
    - **Short description.** This is a brief phrase that provides a meaningful title for the problem report or change.
    - **Originator.** This is the name of the person who identified the problem or change request.
    - **Date originated.** This is the date the report was submitted.
    - **Key words.** This is one or more key words that can be used to group the reports for tracking. Possible key words include the names of the subphase, work package, prototyping or development set, and so on. The number and types of keys should correspond to the reporting and tracking requirements of the project or organisation.

Description

- **Classification.** This describes the change as corrective maintenance, adaptive maintenance, perfective maintenance, preventive maintenance, or user support.
- **Components affected.** This indicates which baseline or component (such as a window, program, or computer) the problem or change affects.
- **Full description.** This includes details about the problem reported or the change requested. For a problem, this should state the conditions under which the problem is encountered and include enough information to reproduce the problem if possible. For a change request, this should describe the business need—the underlying problem—giving rise to the request.

Analysis

- **Analyst assigned.** This is the name of the person assigned to investigate the problem or change.
- **Date assigned.** This is the date when the person responsible was assigned to find a solution.
- **Problem analysis.** This details the analysis done to determine the cause of the problem, including the hypotheses tested and the results

obtained. This may instead point to a separate Problem Analysis Report.
- **Proposed solution.** This details the solution proposed in response to the change request or problem report.
- **Impact analysis.** This summarises the analysis done to determine the impact of the proposed solution and identifies the changes required to implement the proposed solution. This may instead point to a separate Impact Analysis Report.
- **Effort and cost estimate.** This provides estimates for the effort and cost required to implement the proposed solution.
- **Cost/benefit/risk analysis.** This summarises the analysis done to assess the costs, benefits, and risks of the proposed change. This may point to a separate Cost/Benefit/Risk Analysis Report.
- **Recommended disposition.** This recommends whether the request should be rejected, deferred, added to an existing change package, or allocated to a release.

- **Disposition**

  - **Decision.** This states what action was taken, as for example, rejected, deferred, assigned to a change package, or allocated to a release.
  - **Authority.** This states who made the decision, as for example, the CCB.
  - **Date.** This is the date when the decision was made.

- **Status**

  - **Current status.** This indicates whether the request is currently proposed, accepted, rejected, completed, or punch-listed.
  - **Date of status.** This is the date when the status was assigned or changed.

Change/Problem Report

Identification Number: _____

Section 1 — Change or Problem Description

Originator: _____ Date Originated: _____

Type: ☐ Problem or ☐ Change Priority: _____

Affected Component(s): _____ Additional Keys: _____

Change/Problem Description: _____

_____

_____

Section 2 — Resolution

Person Responsible: _____ Date Assigned: _____

Proposed Resolution: _____

_____

_____

Date Resolution Expected: _____ Estimated Cost: _____ Estimated Effort: _____

Section 3 — Change or Problem Source

Source: ☐ Business Change ☐ Requirements Error ☐ Programming Error ☐ Entry Error ☐ No Trouble Found

Comments: _____

_____

Section 4 — Status

Current Status: ☐ Proposed ☐ Accepted ☐ Rejected ☐ Completed ☐ Punch List

Date of Status: _____ _____ _____ _____ _____

Approved By: _____ Date Approved: _____

0

16.6.2 Change/Problem Log

A Change/Problem Log, preferably automated, is used to summarise and track problems and changes. For each problem or change, the log lists key information, such as:

- Type
- Identification number
- Priority
- Short description
- Originator

- Date originated
- Person assigned
- Date assigned
- Current status
- Date of last status change

Change/Problem Log									
ID Num.	Change/Problem Description	Type	Originator	Priority	Resp.	Date Originated	Date Assigned	Date Completed	Status

16.6.3 CR/PR Progress and Status Reports

The purpose of this work product is to provide the Change Control Board(s), management and staff with status information on CR/PR related projects.

A Change Management Status Report is developed from the Change/Problem Log. It provides the status of the change requests and problem reports and provides statistical information regarding the overall process of change management. The change tracking system should automatically produce the following information:

- **Status by change request/problem report.** This lists change requests, problem reports, and their status. The report may be generated for all requests and problems, for requests or problems meeting particular criteria, or even for a single request or problem. The report may be sequenced in various ways for different purposes by criteria such as type (change or problem), date received, product, originator, status, priority, assignee, or release.

- **Accounting of change requests and problem reports processed.** This is a periodic accounting of the number of change requests and problem reports opened and closed during the reporting period. It shows the number of open change requests at the beginning of the period, the number received during the period, the number approved or rejected, and the number remaining open (deferred) at the end of the period. It also shows the number of open problem reports at the beginning of the period, the number received during the period, the number resolved, and the number remaining open at the end of the period. You may wish to slice this data by product, originator, priority, or other data element. This will show how well change management is keeping up with incoming change requests and problem reports.

- **Average age of closed change requests and problem reports.** For each change approved or rejected in a period and for each problem resolved during the period, the system should compute the number of days the change request or problem report has been on the Change/Problem Log. It should then compute the average across the entire set of change requests resolved during the period and do the same for problem reports. Finally, it should plot the value for each period on a trend chart, as in the figure below. This shows how promptly the change requests and problem reports are being addressed.

- **Average age of open change requests and problem reports.** For each open change request or problem report, the system should compute the number of days the request or report has been on the Change/Problem Log. It should then compute the average across the entire set of change requests open at the end of the period and do the same for problem reports. Finally, it should plot the value for each period on a trend chart, as in the figure below. A steady increase in the average age of open change requests or problem reports indicates an issue for the project or organisation.

- **Ae distribution of open change requests.** For each open change request and problem report, the system should compute the number of days the change request or problem report has been in the Change/Problem Log. It should then count the number of open change requests and problem reports by age group, as for example: less than two weeks, more than two weeks, and more than four weeks. Using this information, the system should plot the age distribution of each period as a histogram, as in the figure below.

16.7 Change Control Process Components

16.7.1 Change Control Board

The change control board (CCB) refers to one or more boards with authority for reviewing and approving changes. The change control board members should collectively have the breadth of view and level of authority to make decisions on priorities and funding for the changes they are reviewing. Sometimes multiple control boards are established to handle different types of changes. In some cases, an individual may be vested with this authority for certain types of changes.

The change control board plays a critical part in change management. In some organisations, this role is referred to as the configuration control board.

The functions of the CCB are:

- Authorises and prioritises Change Requests
- Requires formal, written Change Requests
- Allocates resources for Change Requests
- Assign Change Requests to specific releases

- Reviews metrics for future process improvements

The key issues the CCB decides on are:

- Is the requested change compatible with the original design?
- Is the change necessary?
- Will the change meet the business objectives, time and resources?
- Does the benefit outweigh the cost?
- Is the change compatible with the strategic plan, how does it match the portfolio plans?
- Is the change within the affordability/budget limits?

The purposes of the CCB are:

- Provides forum for business community to participate in the decision making process
- Examine the criticality of the change compared to others in the backlog
- Determine the business impacts
- How soon can the payback be realised?
- How certain is the payback?
- What are the associated risks?
- Analyse related changes already in the backlog and consider combinations that generate economies of scale
- Prioritise across the enterprise as well as within applications

16.7.2 Change Control Office

The change control office (CCO) provides administrative support to the change control board. It processes changes requests and problem reports, prepares agendas for the change control board, reports change and problem status, and otherwise performs the day-to-day functions of change management.

17. Appendix 5 – Risk Management

17.1 Introduction

Risk is the possibility of incurring a negative impact to the project. Proactive risk management is fundamental to effective project management. As project manager, you can use certain techniques and procedures to anticipate problems and reduce risk.

Risk management is a continuous process comprised of six activities grouped into two major areas—risk assessment and risk mitigation.

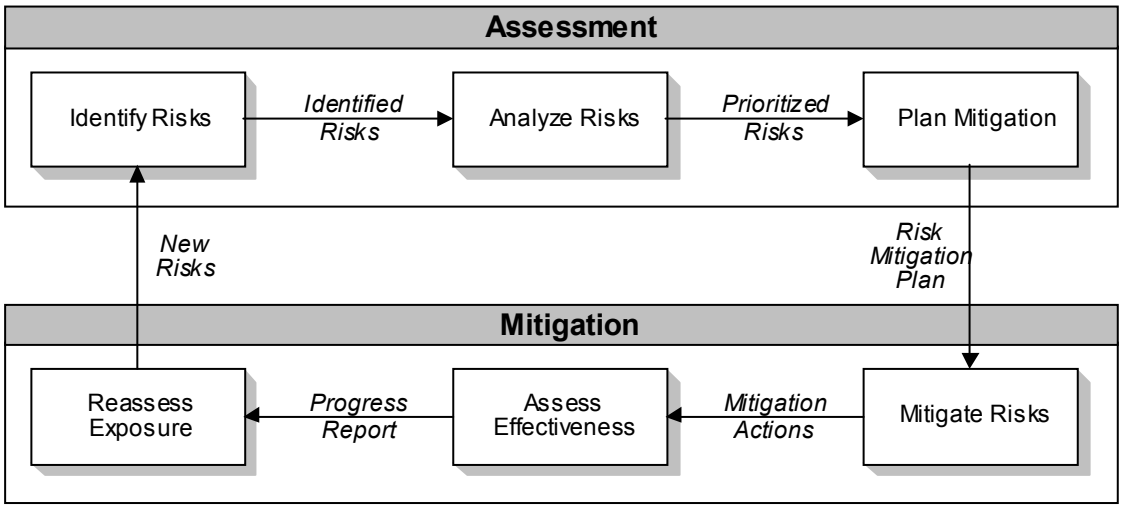
The major activities of risk assessment are:

- Identify risks
- Analyse risks
- Plan risk mitigation.

The major activities of risk mitigation are:

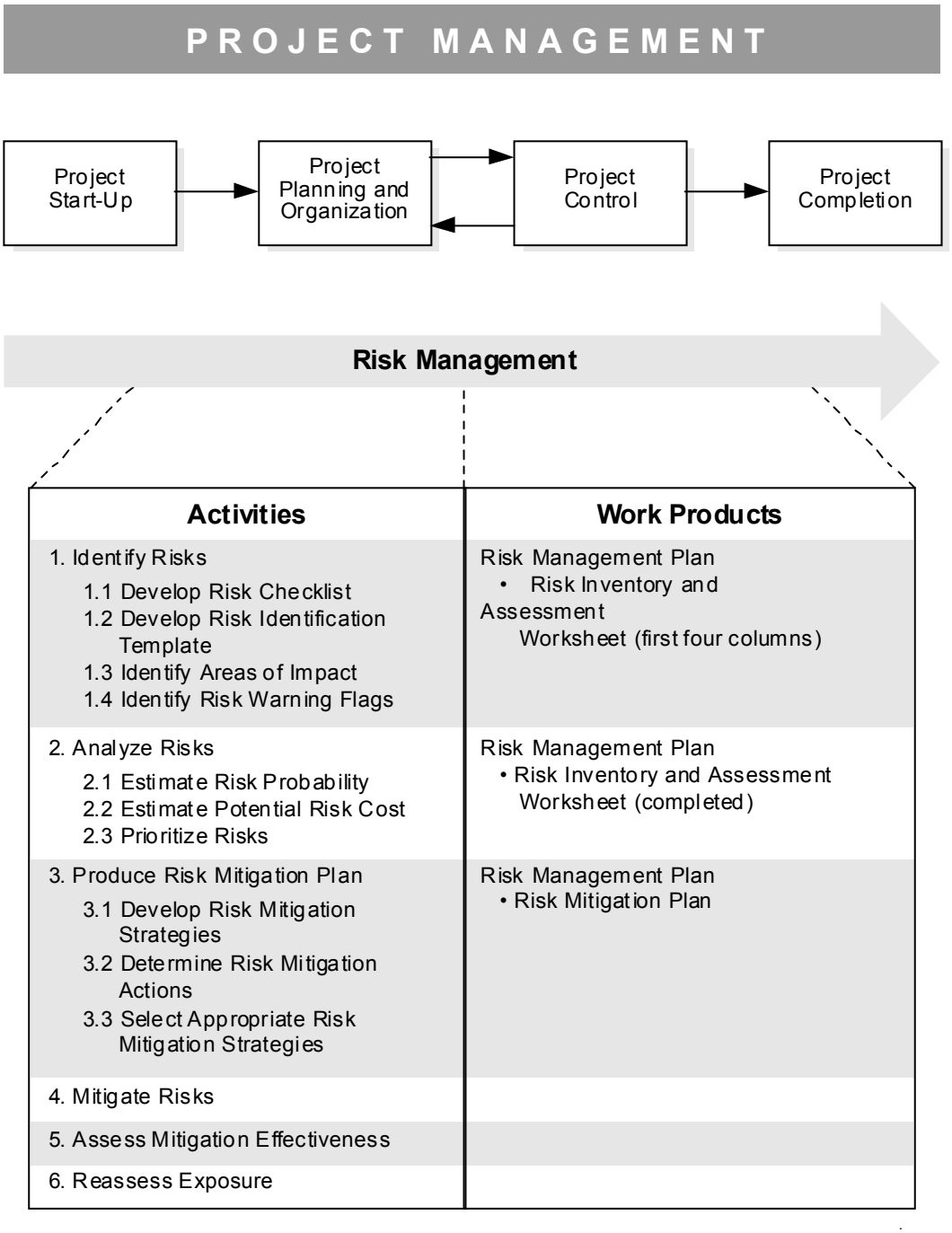
- Mitigate risks
- Assess mitigation effectiveness
- Reassess exposure.

Risk management is an iterative process established when the project begins and continuously performed until the project ends. The risk management concept is illustrated in the figure below.



17.2 Risk Management Activities

Schematically, the risk management activities are:



Activities	Work Products
1. Identify Risks 1.1 Develop Risk Checklist 1.2 Develop Risk Identification Template 1.3 Identify Areas of Impact 1.4 Identify Risk Warning Flags	Risk Management Plan • Risk Inventory and Assessment Worksheet (first four columns)
2. Analyze Risks 2.1 Estimate Risk Probability 2.2 Estimate Potential Risk Cost 2.3 Prioritize Risks	Risk Management Plan • Risk Inventory and Assessment Worksheet (completed)
3. Produce Risk Mitigation Plan 3.1 Develop Risk Mitigation Strategies 3.2 Determine Risk Mitigation Actions 3.3 Select Appropriate Risk Mitigation Strategies	Risk Management Plan • Risk Mitigation Plan
4. Mitigate Risks	
5. Assess Mitigation Effectiveness	
6. Reassess Exposure	

These activities are described in more detail in the following sections.

17.2.1 Identify Risks

Risk identification is the foundation of the risk management process. Begin identifying risks as early as possible, ideally during the proposal process, and continue it throughout the project. Identify risks by reviewing and analysing sources of information including the request for proposal, the proposal, the Statement of Work, the engagement letter, the contract, estimation worksheets, checklists, and any notes. Document identified risks by completing the first four columns of the Risk Inventory and Assessment Worksheet.

When identifying risks, do not limit yourself to written material. Speak with people who know and understand this client's environment. Apply experience and insight to identify potential risks.

A work session with key project team members and experienced personnel is an ideal vehicle for risk identification. Structure it as a brainstorming session to analyse every aspect of the project, with emphasis on the things that can go wrong.

Perform the following activities.

17.2.1.1 Develop Risk Checklist

Develop and maintain a list of potential risks. As you learn more about various risks, document them in more detail. As project manager, regularly compare earlier expectations with current project reality. Divide your risk checklist into three categories:

- **Known risks**. Known risks are the easiest to analyse since they are typically related to schedule and budget commitments or to concrete project attributes like performance criteria. They can be identified before the project begins by examining both the project environment and all existing project documentation. Review the request for proposal, the proposal, the Statement of Work, the contract, and other available sources of information.
- **Predictable risks**. By analysing problems encountered on similar projects, you can often identify predictable risks, such as those related to the reliability, availability, and effectiveness of personnel or other resources. To avoid such risks, include commitments for people and resources in the contract or in the project approach.
- **Unpredictable risks**. Some risks that cannot be evaluated in advance can threaten underlying assumptions that define the success, direction, or existence of the project. Examples include executive management changes, divestiture of product lines, and major financial problems.

The vast majority of risks are known or predictable. They are typically conditions found in the basic work arrangement or in similar engagements. The box below shows a risk checklist consisting of questions to help you identify risks in the three categories.

Risk Checklist

Known Risks

- Has the project's Statement of Work been formalised and approved?
- Are there rigid contract terms, conditions, or constraints?
- Are there unverified bid assumptions?
- Are the Acceptance Criteria defined and agreed to?
- Will it be difficult to receive additional project funding, if needed?
- Is the schedule realistic?
- Is each milestone in the project schedule reasonable?
- Can delivery dates be adjusted if needed?

- Are the bid rates adequate for the required staff skills?
- Are the project manager and staff experienced in this type of project?
- Is there a recognised project sponsor?
- Are the preliminary assumptions valid?
- Were estimates based on actual effort for similar processes and deliverables?
- Is the subcontractor known and reliable?

Predictable Risks

- Will promised information sources be available and of adequate quality?
- Will user reviews and acceptance be timely?
- Will client management decisions be timely?
- Are there client dependencies?
- Is client management committed to this project?
- Is the client known to be cooperative and reasonable?
- Are client expectations known and acceptable?
- Does the client resolve issues quickly?
- Has the client agreed to required scope changes in the past?
- Is the development environment adequate?
- Is the test environment adequate?
- Will new or unfamiliar development tools be used?
- Are the workspace and work environment adequate?
- Can the workspace accommodate a larger staff, if needed?
- Has a similar project attempt failed before?
- Are requirements completely and adequately specified?
- Is the design complete?
- Is the work completely defined?
- Will the acceptance process be difficult?
- Is the technical infrastructure adequate?
- Is a new technology required?
- Are the requirements technically complex or innovative?
- Are the functions complex or difficult?
- Are the performance requirements demanding?
- Will key stakeholders be accessible and committed to the project results?
- Will technical support be committed to the project?
- Will there be critical dependencies on subcontractors?
- Does project success depend on a few key individuals?
- Does the project plan accommodate a staff learning curve?
- Is the staff build-up plan practical?
- Has the project plan and schedule accounted for vacation, sickness, and administrative time?

Unpredictable Risks

- Will the client's priorities change?
- Will the business case remain persuasive?
- Will funding availability change?
- Will project objectives be redefined?
- Will key project personnel change?

17.2.1.2 Develop Risk Identification Template

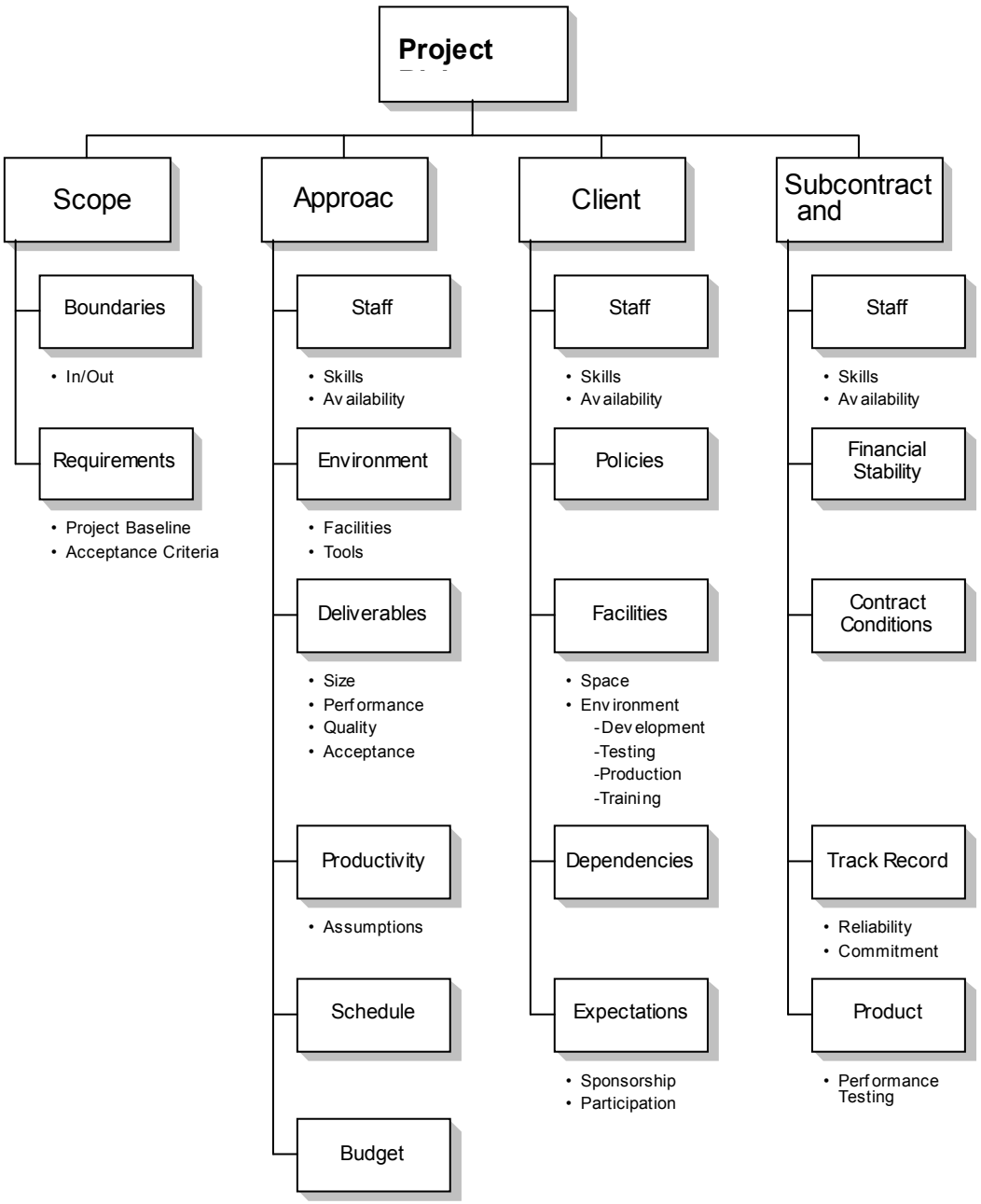
The risk identification template shown in the following figure is a second technique for identifying risks. This technique is a complement, not an alternative, to the three-category approach to risk identification used in the risk checklist. The structure of the template is based on key areas of the proposal. Apply the template when reviewing the proposal to identify known risks based on information contained in or missing from the proposal. The template focuses first on the scope section of the proposal and addresses issues such as:

- Is the definition of scope boundaries specific in identifying what is in the project scope and what is out of scope?
- Are the project requirements defined in adequate detail?
- Are the requirements testable to validate satisfactory completion of the engagement?

Next, the interrogation focuses on the approach section of the proposal and addresses issues such as:

- Are the tools and facilities adequate? If not, what specific problems or risks do they pose?
- Are the productivity estimates based on documented history?
- Are the estimates supported by documented assumptions?

Ask similar questions about the client and subcontract sections of the proposal. The template focuses the project manager when analysing the project. It is not exhaustive; it is an aid and beginning point to be applied and tailored to the unique circumstances of each engagement.



17.2.1.3 Identify Areas of Impact

An area of impact is a place where damage would be felt if the risk were to become a reality. Generally, impact occurs in cost, schedule, quality of work products, or some combination of these. A project with fixed budgets or "drop dead" dates is especially vulnerable to risks. Once risk identification is complete, hold a management review to ensure that identification is complete and accurate.

17.2.1.4 Identify Risk Warning Flags

Risk warning flags are the events, symptoms, or circumstances signalling risk. Examples of risk warning flags are:

- Lack of sponsor/user attendance at review meetings
- Actual task time or costs higher than projected
- Software modifications more extensive than projected.

Identify these risk warning flags to establish the parameters you must monitor to analyse the risk. For example, if the risk is time-dependent, monitor dates. For risks resulting from the accumulation of incremental events, establish and monitor a danger threshold for the number of occurrences of these events. Risks threatening only certain project phases should still be identified as risks. An example of such a risk is the inaccessibility of a user—a considerable risk during Business System Design but not during Application Development.

17.2.2 Analyse Risks

Risk analysis quantifies the threat posed by each risk so you can set priorities on risk management activities. Risk analysis helps determine the intensity with which the risk can affect the project and indicates what resources to apply to control or eliminate those risks. Risk analysis is an ongoing process, and any new or changed risks should be incorporated into the analysis from a project's start through to its completion.

As you perform the following activities, record your results on the Risk Inventory and Assessment Worksheet.

17.2.2.1 Estimate Risk Probability

Estimate the probability that the risk will become a reality. This risk probability is a percentage estimate based on your experience and insights. Even though these estimates are imprecise, they are worthwhile. Assign risk probability as follows:

- **High risk.** If the problem is likely to occur, assign a probability of 70 to 90 percent.
- **Medium risk.** If the problem is as likely to occur as not occur, assign a probability from 40 to 60 percent.
- **Low risk.** If the problem is unlikely to occur, assign a probability of 10 to 30 percent.

17.2.2.2 Estimate Potential Risk Cost

Estimate the potential cost associated with each risk. The two primary methods for estimating potential risk cost are:

- **Estimate the risk cost of a specific event.** For example, if the integration test is one month late, potential risk cost would equal the cost of

maintaining the integration test team for the additional month. This quantifies the event at a tactical level. Further analysis should assess and quantify the potential impact of the delay on the client organisation. For example, will there be lost revenues, or will the client be non-compliant with a legal or regulatory mandate? These impacts are at a strategic level and could be much more costly from both a financial and a relationship perspective.

- **Estimate risk cost as a percentage of project value**. For example, if the scope were understated by 10 percent, potential risk cost would equal 10 percent of the estimated project cost.

17.2.3 Prioritise Risks

Set risk priorities to direct focus where it is most critical. There are not enough resources to mitigate all risks on any project. Therefore, you must determine the risks that are potentially most harmful to the project.

Establish risk priorities by first calculating the probable risk cost. The probable risk cost is the potential risk cost multiplied by the risk probability. Risks are automatically prioritised by sequencing the probable risk cost from high to low; the highest probable risk cost is the number-one priority. At a minimum, any risk that has a probable risk cost greater than five percent of the project budget should be in the high-risk, high-priority category.

Also consider timing. You may want to raise the priority of imminent risks—those likely to cause the most immediate problems.

17.2.4 Produce Risk Mitigation Plan

Risk mitigation is the action taken to eliminate, reduce, or control project risks. Any risk for which there is no risk mitigation plan has the potential for significant negative impact. Risk management then becomes reactive rather than proactive, often in the form of damage control.

Perform the following activities to develop the Risk Mitigation Plan.

17.2.4.1 Develop Risk Mitigation Strategies

The Risk Mitigation Plan is comprised of risk mitigation strategies. A risk mitigation strategy is a set of actions directed at minimising the potential negative impacts of risks on a project's success. As project manager, you should develop strategies to mitigate a project's risk and obtain client acceptance of the proposed strategies. There are three types of risk mitigation strategies:

- **Preemptive**. This is action taken to eliminate or circumvent the risk before a problem occurs. Preemptive action is the most effective approach and

should be used for critical and serious risks. However, some preemptive actions are very expensive. Use them judiciously.

- **Direct**. This is action taken to reduce or nullify the risk before any significant damage results.
- **Indirect**. This is action taken to offset or absorb the damage after the fact. This approach is reactive rather than proactive and is the least desirable.

Each risk mitigation strategy includes one or more specific risk mitigation actions intended to lessen the risk probability or impact. Assign full responsibility and accountability for each of these actions to key individuals on the project. Document each risk mitigation strategy as illustrated in the following figure. Include the following:

1. A description of the risk
2. Planned mitigation actions
3. A risk priority category (H, M, or L)
4. The person responsible for taking action
5. When the task was assigned
6. The target removal date
7. The current risk status.

RISK MITIGATION STRATEGY

Risk ID Number: 1 **Risk Priority:** H

Risk Title: Scope Increases by 10%

Risk Description:

The functional scope of the project may increase for several reasons:
- As users review and verify the functions of reports and screens, small changes may be required.
- Gaps in the procedural flow may require additional reports, screens, and/or processes to be included.
- Users and/or MIS may desire increased functionality in the system.

Risk Mitigation Action	Risk Mitigation Results	Responsibility	Target Removal Date	Status
Set a firm base-line for the scope of the project.		Project Manager	8/94	
Gain a commit-ment from the Project Sponsor to control project budget and schedule.	User manage-ment has issued a memo to the team stating a goal of completing the project on time and under budget by 10%.	Project Manager	8/94	Complete

dk920105.43

17.2.4.2 Determine Risk Mitigation Actions

The box below provides examples of risk mitigation actions typically found within the three types of risk mitigation strategies. Note that some actions, such as providing training, could be classified as either preemptive or direct, depending on when the action is taken. The key to successful risk mitigation is to apply the most appropriate action at the proper time.

Risk Mitigation Actions

Preemptive

The preemptive risk mitigation strategy could apply the following risk mitigation actions:

- Create benchmarks for performance; make a separate prototype
- Start early
- Provide training
- Do critical functions first
- Document decisions and agreements
- Get formal sign-off
- Implement a formal scope control procedure
- Develop a personnel succession plan.

Direct

The direct risk mitigation strategy could apply the following risk mitigation actions:

- Negotiate a change
- Build in a management budget reserve
- Procure required resources and personnel
- Conduct reviews or walk-throughs
- Take corrective action to fix a specific problem
- Use productivity tools.

Indirect

The indirect risk mitigation strategy could apply the following risk mitigation actions:

- Extend work hours
- Increase staff
- Modify scope
- Delay schedules.

17.2.4.3 Select Appropriate Risk Mitigation Strategies

Effective risk mitigation is a balancing act with two critical dimensions—time and cost. The implementation of a feasible risk mitigation strategy must always cost less than the probable risk cost.

Determine the right time to apply a risk mitigation strategy. While preemptive mitigation is ideal, its cost is frequently prohibitive. Conversely, waiting too long may preclude the ability to mitigate a risk. For example, it is very difficult to

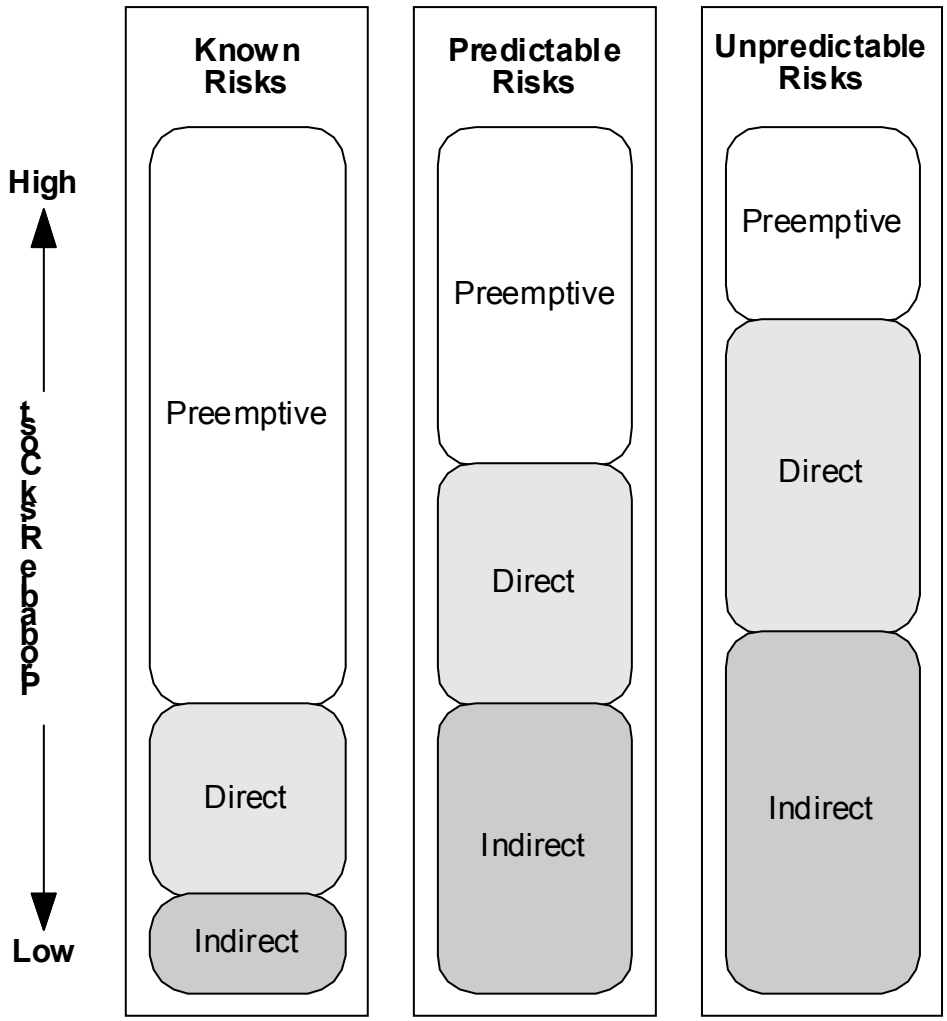
institute a scope control procedure when the project is one-third complete and there are major scope issues.

Which type of risk mitigation strategy you select depends on the nature of the risk, the likelihood of its occurrence, and the degree of potential harm. Use the following figure to help you decide the appropriate strategy. This figure depicts the likely proportional application of the preemptive, direct, and indirect risk mitigation strategies, given the:

- Three types of risk—known, predictable, and unpredictable
- Probability of the risk occurring
- Range of cost to the project if the risk occurs.

When selecting risk mitigation strategies, consider the following:

- **Known risks**. For known risks that have a high probability and a high potential cost, choose a preemptive mitigation strategy. A project scope that is very likely to expand and cause high cost overruns is an example of such a situation. In this case, it would be wise to institute a formal change-of-scope procedure to control this potentially costly risk exposure. Direct or indirect strategies may be more appropriate for known risks with a low probability and a low potential cost.
- **Predictable risks**. Direct mitigation strategies are appropriate for most predictable risks. Develop risk mitigation strategies for the risks with the highest probability and quickly implement them when appropriate to minimise risk cost. Indirect strategies may be developed for those predictable risks with a low probability of occurrence.
- **Unpredictable risks.** On the other extreme are unpredictable risks that have a low probability and a low potential cost. In these cases, choose not to invest much in preemptive mitigation. The preferred approach is to monitor the situation and apply indirect mitigation strategies only when a problem occurs. An example of this type of risk is a natural disaster, such as a flood or tornado, halting the project until new quarters can be found.



17.2.5 Mitigate Risks

Implement the Risk Mitigation Plan. Activate preemptive risk mitigation strategies. For example, implement a formal scope control process before the project experiences scope problems. Monitor the risk warning flags and react quickly to implement the planned direct risk mitigation strategies. Implement strategies in increasing levels of intensity. For example, tutor a problem staff member before deciding whether to replace him or her. When necessary, apply indirect risk mitigation strategies, such as extending work hours to reduce a schedule slippage.

Effective mitigation is proactive. Problems rarely solve themselves. As project manager you must aggressively take action to be most effective.

17.2.6 Assess Mitigation Effectiveness

It is essential to assess the effectiveness of your risk mitigation actions. Various approaches can be employed, but the key is to evaluate the project objectively. Risk mitigation effectiveness can be assessed at two levels:

- **At a low level of detail.** Evaluate the status of each risk where a mitigation strategy is in place. For example, if a scope control procedure is in place, ask questions like:

    - Is the project still within the bounds of the project baseline?
    - Are scope issues being documented and reviewed?
    - If scope changes are approved, is the project baseline being updated and have the project schedule and budget received the appropriate increases?
    - If an acceptance procedure is in place, determine if:
    - Work products are being reviewed and signed off within the time parameters initially established
    - Inefficient approval procedures are causing significant delays in the approval process
    - It is difficult to obtain approval from certain individuals.

- **At a high level of detail.** To evaluate the effectiveness of risk mitigation strategies at a higher level of detail, assess the cost, schedule, and quality dimensions of the project. Effective project tracking reflects cost and schedule status. Assess work product quality through individual reviews and testing. If the project is over budget or if schedule or deliverable quality is not acceptable, reassess the project's risk exposure.

Risk management is integral to project management and should not be treated as a separate entity. Establish procedures for risk management status reporting as a regular component of the normal project reporting process.

Once project tracking and reporting procedures are in place, regularly review their validity. Continually monitor project controls. For example, do not assume that just because you have a scope control process it is automatically effective. Talk to staff, review products, and talk to users to find out how the project is really going. Encourage regular project audits to obtain an objective view.

17.2.7 Reassess Exposure

Many experienced project managers receive surprises late in the project as they discover a significant problem caused by an undetected risk. Success can lull the project manager into a state of complacency. Sometimes he or she is just too busy to notice trouble brewing.

Avoid this situation by evaluating the project's current risk status at least monthly to address the dynamic realities of a project. This becomes more important as the project approaches integration and deployment. Frequently, new risks arise from integration issues. It is often appropriate to reassess risks on a weekly basis during the final three months of the project and during the first month of deployment.

Be sure to address changed probability of risks and changed impact, as well as any new risks that may surface. When a new risk exposure is identified or a risk probability changes, begin another iteration of the risk assessment process. Repeat the same rigorous scrutiny applied early in the project's life cycle to identify and analyse risks.

18. Appendix 6 – Sample Test Plan

18.1 Revision History

First Draft: <author> <date>
<brief description of changes>

18.2 Introduction

Single sentence describing the intent and purpose of the test plan. For example "This test plan addresses the test coverage for the XXX release of the YYY area of feature ZZZ".

18.3 Goal of Project and Feature Team

Mission statement and goal of overall project team.
Mission statement and goal of specific feature team.
This section is used to set the stage for testing's plans and goals in relation to the feature team and project's goals.

18.4 Primary Testing Concerns

A statement of what the main critical concerns of the test plan are. An itemised list, or short paragraph will suffice.

18.5 Primary Testing Focus

A short statement of what items testing will focus on. The testing concerns above state what testing is worried about. Focus indicates more of a methodology - a statement of how those concerns will be addressed via focus.

18.6 References

- document name location
- test plan test plan location
- project specifications project spec location
- feature specification feature spec location
- development docs on feature dev doc location
- bug database queries location for raid queries
- test case database queries location for test case queries
- schedule documents location for schedule documents
- build release server location of build releases

- source file tree — location of source file tree
- other related documents — other locations

18.7 Personnel

Project Manager:	name and email
Developer:	name and email
Tester:	name and email

18.8 Testing Schedule

Break the testing down into phases (ex. Planning, Case Design, Unit & Component Tests, Integration Tests, Stabilisation, Performance and Capacity Tuning, Full Pass and Shipping) - and make a rough schedule of sequence and dates. What tasks do you plan on having done in what phases? This is a brief, high level summary - just to set expectation that certain components will be worked on at certain times - and to indicate that the plan is taking project schedule concerns into consideration.

Include a pointer to more detailed feature and team schedules here.

18.9 Feature History

A history of how the feature was designed, and evolved, over time. It is a good idea to build this history up as test plans go. This gives a good feel for why the current release is focusing on what it has done. It also serves a good framework for where problems have been in the past.

A paragraph or two is probably sufficient for each drop, indicating - original intent, feedback and successes, problems, resolutions, things learned from the release, major issues dealt with or discovered in the release.

Basically, this section is a mini post-mortem. It is eventually finishes with a statement regarding the development of the specific version.

It is often helpful to update this history at each milestone of a project.

18.10 Features

This section gives a breakdown of the areas of the feature. It is often useful to include in this section a per area statement of testing's thoughts. What type of testing is best used for each area? What is problematic about each area? Has this area had a problem in the past. Quick statements are all that is need in this list.

NOTE: this is only here as a high level summary of the features. The real meat is in the area breakdown.

18.11 Files and Modules

Include in this section any files, modules and code that must be distributed on the machine, and where they would be located. Also include registry settings, INI settings, setup procedures, de-installation procedures, special database and utility setups, and any other relevant data.

18.11.1 Files List

filename purpose location on machine

18.11.2 Registry, INI Settings

setting1 purpose
Setting1 possible values
setting 2 purpose
Setting 2 possible values

18.11.3 Setup Procedures

18.11.4 De-installation Procedures

18.11.5 Database Setup and Procedures

18.11.6 Network Domain/Topologies Configuration Procedures

18.11.7 Performance Monitoring Counters Setup And Configurations

18.12 Operational Issues

Is the program being monitored/maintained by an operational staff? Are there special problem escalation, or operational procedures for dealing with the feature/program/area?

18.12.1 Backup

18.12.2 Recovery

18.12.3 Archiving

18.12.4 Monitoring

18.12.5 Operational Problem Escalation/Alert Methods

18.13 Scope of Test Cases

Statement regarding the degree and types of coverage the testing will involve. For example, will focus be placed on performance? How about client v.s. server issues? Is there a large class of testing coverage that will be intentionally overlooked or minimised? Will there be much unit and component testing? This is a big sweeping picture of the testing coverage - giving an overall statement of the testing scope.

18.14 Acceptance Criteria

How is “Good Enough To Ship” defined for the project? For the feature? What are the necessary performance, stability and bug find/fix rates to determine that the product is ready to ship?

18.15 Key Feature Issues

What are the top problems/issues that are recurring or remain open in this test plan? What problems remain unresolved?

18.16 Test Approach

18.16.1 Design Validation

Statements regarding coverage of the feature design - including both specification and development documents. Will testing review design? Is design

an issue on this release? How much concern does testing have regarding design, etc.

18.16.2 Data Validation

What types of data will require validation? What parts of the feature will use what types of data? What are the data types that test cases will address?, etc.

18.16.3 API Testing

What level of API testing will be performed? What is justification for taking this approach (only if none is being taken)?

18.16.4 Content Testing

Is your area/feature/product content based? What is the nature of the content? What strategies will be employed in your feature/area to address content related issues?

18.16.5 Low-Resource Testing

What resources does your feature use? Which are used most, and are most likely to cause problems? What tools/methods will be used in testing to cover low resource (memory, disk, etc.) issues?

18.16.6 Setup Testing

How is your feature affected by setup? What are the necessary requirements for a successful setup of your feature? What is the testing approach that will be employed to confirm valid setup of the feature?

18.16.7 Modes and Runtime Options

What are the different run time modes the program can be in? Are there views that can be turned off and on? Controls that toggle visibility states? Are there options a user can set which will affect the run of the program? List here the different run time states and options the program has available. It may be worthwhile to indicate here which ones demonstrate a need for more testing focus.

18.16.8 Interoperability

How will this product interact with other products? What level of knowledge does it need to have about other programs -- "good neighbour", program cognisant, program interaction, fundamental system changes? What methods will be used to verify these capabilities?

18.16.9 Integration Testing

Go through each area in the product and determine how it might interact with other aspects of the project. Start with the ones that are obviously connected, but try every area to some degree. There may be subtle connections you do not think about until you start using the features together. The test cases created with this approach may duplicate the modes and objects approaches, but there are some areas which do not fit in those categories and might be missed if you do not check each area.

18.16.10 Compatibility: Clients

Is your feature a server based component that interacts with clients? Is there a standard protocol that many clients are expected to use? How many and which clients are expected to use your feature? How will you approach testing client compatibility? Is your server suited to handle ill-behaved clients? Are there subtleties in the interpretation of standard protocols that might cause incompatibilities? Are there non-standard, but widely practiced use of your protocols that might cause incompatibilities?

18.16.11 Compatibility: Servers

Is your feature a client based component that interacts with servers? Is there a standard protocol supported by many servers that your client speaks? How many different servers will your client program need to support? How will you approach testing server compatibility? Is your client suited to handle ill-behaved or non-standard servers? Are there subtleties in the interpretation of standard protocols that might cause incompatibilities? Are there non-standard, but widely practiced use of protocols that might cause incompatibilities?

18.16.12 Beta Testing

What is the beta schedule? What is the distribution scale of the beta? What is the entry criteria for beta? How is testing planning on utilising the beta for feedback on this feature? What problems do you anticipate discovering in the beta? Who is coordinating the beta, and how?

18.16.13 Environment/System – General

Are there issues regarding the environment, system, or platform that should get special attention in the test plan? What are the run time modes and options in the environment that may cause difference in the feature? List the components of critical concern here. Are there platform or system specific compliance issues that must be maintained?

18.16.14 Configuration

Are there configuration issues regarding hardware and software in the environment that may get special attention in the test plan? Some of the classical issues are machine and bios types, printers, modems, video cards and drivers, special or popular TSR's, memory managers, networks, etc. List those types of configurations that will need special attention.

18.16.15 User Interface

List the items in the feature that explicitly require a user interface. Is the user interface designed such that a user will be able to use the feature satisfactorally? Which part of the user interface is most likely to have bugs? How will the interface testing be approached?

18.16.16 Performance & Capacity Testing

How fast and how much can the feature do? Does it do enough fast enough? What testing methodology will be used to determine this information? What criterion will be used to indicate acceptable performance? If modifications of an existing product, what are the current metrics? What are the expected major bottlenecks and performance problem areas on this feature?

18.16.17 Scalability

Is the ability to scale and expand this feature a major requirement? What parts of the feature are most likely to have scalability problems? What approach will testing use to define the scalability issues in the feature?

18.16.18 Stress Testing

How does the feature do when pushed beyond its performance and capacity limits? How is its recovery? What is its breakpoint? What is the user experience when this occurs? What is the expected behaviour when the client reaches stress levels? What testing methodology will be used to determine this information? What area is expected to have the most stress related problems?

18.16.19 Volume Testing

Volume testing differs from performance and stress testing in so much as it focuses on doing volumes of work in realistic environments, durations, and configurations. Run the software as expected user will - with certain other components running, or for so many hours, or with data sets of a certain size, or with certain expected number of repetitions.

18.16.20 International Issues

Confirm localised functionality, that strings are localised and that code pages are mapped properly. Assure program works properly on localised builds, and that international settings in the program and environment do not break functionality. How is localisation and internationalisation being done on this project? List those parts of the feature that are most likely to be affected by localisation. State methodology used to verify International sufficiency and localisation.

18.16.21 Robustness

How stable is the code base? Does it break easily? Are there memory leaks? Are there portions of code prone to crash, save failure, or data corruption? How good is the program's recovery when these problems occur? How is the user affected when the program behaves incorrectly? What is the testing approach to find these problem areas? What is the overall robustness goal and criteria?

18.16.22 Error Testing

How does the program handle error conditions? List the possible error conditions. What testing methodology will be used to evoke and determine proper behaviour for error conditions? What feedback mechanism is being given to the user, and is it sufficient? What criteria will be used to define sufficient error recovery?

18.16.23 Usability

What are the major usability issues on the feature? What is testing's approach to discover more problems? What sorts of usability tests and studies have been performed, or will be performed? What is the usability goal and criteria for this feature?

18.16.24 Accessibility

Is the feature designed in compliance with accessibility guidelines? Could a user with special accessibility requirements still be able to utilise this feature? What is the criteria for acceptance on accessibility issues on this feature? What is the

testing approach to discover problems and issues? Are there particular parts of the feature that are more problematic than others?

18.16.25 User Scenarios

What real world user activities are you going to try to mimic? What classes of users (i.e. secretaries, artist, writers, animators, construction worker, airline pilot, shoemaker, etc.) are expected to use this program, and doing which activities? How will you attempt to mimic these key scenarios? Are there special niche markets that your product is aimed at (intentionally or unintentionally) where mimic real user scenarios is critical?

18.16.26 Boundaries and Limits

Are there particular boundaries and limits inherent in the feature or area that deserve special mention here? What is the testing methodology to discover problems handling these boundaries and limits?

18.16.27 Operational Issues

If your program is being deployed in a data center, or as part of a customer's operational facility, then testing must, in the very least, mimic the user scenario of performing basic operational tasks with the software.

18.16.27.1 Backup

Identify all files representing data and machine state, and indicate how those will be backed up. If it is imperative that service remain running, determine whether or not it is possible to backup the data and still keep services or code running.

18.16.27.2 Recovery

If the program goes down, or must be shut down, are there steps and procedures that will restore program state and get the program or service operational again? Are there holes in this process that may make a service or state deficient? Are there holes that could provide loss of data. Mimic as many states of loss of services that are likely to happen, and go through the process of successfully restoring service.

18.16.27.3 Archiving

Archival is different from backup. Backup is when data is saved in order to restore service or program state. Archive is when data is saved for retrieval later. Most archival and backup systems piggy-back on each other's processes.

Is archival of data going to be considered a crucial operational issue on your feature? If so, is it possible to archive the data without taking the service down? Is the data, once archived, readily accessible?

18.16.27.4 Monitoring

Does the service have adequate monitoring messages to indicate status, performance, or error conditions? When something goes wrong, are messages sufficient for operational staff to know what to do to restore proper functionality? Are the "hearbeat" counters that indicate whether or not the program or service is working? Attempt to mimic the scenario of an operational staff trying to keep a service up and running.

18.16.27.5 Upgrade

Does the customer likely have a previous version of your software, or some other software? Will they be performing an upgrade? Can the upgrade take place without interrupting service? Will anything be lost (functionality, state, data) in the upgrade? Does it take unreasonably long to upgrade the service?

18.16.27.6 Migration

Is there data, script, code or other artefacts from previous versions that will need to be migrated to a new version? Testing should create an example of installation with an old version, and migrate that example to the new version, moving all data and scripts into the new format.

List here all data files, formats, or code that would be affected by migration, the solution for migration, and how testing will approach each.

18.16.28 Special Code Profiling and Other Metrics

How much focus will be placed on code coverage? What tools and methods will be used to measure the degree to which testing coverage is sufficiently addressing all of the code?

18.17 Test Environment

What are the requirements for the product? They should be reflected in the breadth of hardware configuration testing.

18.17.1 Operating Systems

Identify all operating systems under which this product will run. Include version numbers if applicable.

18.17.2 Networks

Identify all networks under which this product will run. include version numbers if applicable.

18.17.3 Hardware

Identify the various hardware platforms and configurations.

18.17.3.1 Machines

18.17.3.2 Graphics Adapters

This includes the requirements for single or dual monitors.

18.17.3.3 Other Peripherals

Peripherals include those necessary for testing such as CD-ROM, printers, modems, faxes, external hard drive, tape readers, etc.

18.17.4 Software

Identify software included with the product or likely to be used in conjunction with this product. Software categories would include memory managers, extenders, some TSRs, related tools or products, or similar category products.

18.18 Unique Testing Concerns For Specific Features

List specific features which may require more attention than others, and describe how testing will approach these features. This is to serve as a sort of "hot list".

18.19 Area Breakdown

This is a detailed breakdown of the feature or area - and is best done in an outline format. It is useful as a tool later when building test cases. The outline of an area can go on quite long. Usually it starts with a menu breakdown, and then

continues on with those features and functionalities not found on any menu in particular.

18.19.1 Feature Name

18.19.1.1 Sub Feature One

18.19.1.1.1 Sub 1.1

Feature testing approach matrix: this will repeat for each subitem, including any class of testing relevant to any item. Put in NA if not applicable. Location of this matrix in the hierarchy determines scope. For example, data validation rules global to anything under Sub Feature One should go under "Sub Feature One". Inheritance should be implied.

Class	Info	Auto?	Man.?
Design Validation			
Data Validation	Valid data & expected results (e.g. "alphanumeric") Invalid data & expected results (e.g. "no ';', '/' or '@') How to validate?		
API Testing	What are the API's exposed? What are the permutations of calling these API's (order, specific args, etc.)?		
Content Testing	What content exercises this feature? What content does this feature produce, modify or manage?		
Low-Resource Testing	What resource dimensions to test? What to do when resource is low?		
Setup Testing	What types of setups? How to confirm feature after a setup?		
Modes & Runtime Options	What modes and runtime options does this have? What should be tested during these modes? What are expected results in different modes?		
Interoperability	What do we interoperate with? Do what action with it?		
Integration Testing	What do we integrate with? Do what action with it?		
Compatibility: Clients	What clients? Doing what actions?		
Compatibility: Servers	What servers? Doing what actions?		
Beta Testing			

Environment/System	What environmental issues apply to this? What to do to expose?		
Configuration	What environmental issues apply to this? What to do to expose?		
User Interface	What are the interface points? How to exercise them?		
Performance	What are the target performance dimensions? What will you do to exercise these?		
Capacity	What is the target capacity? What will you do to test this?		
Scalability	What is the target scale, and how? What will you do to test this?		
Stress	What dimensions do you plan on stressing? What is expectation? How will you stress it?		
Volume Tests	What actions will be included in volume tests?		
International	What are the international problems of this item?		
Robustness	What robustness (crashes, corruption, etc.) errors are anticipated? How will you look for them?		
Error Testing	What are the relevant error conditions that the program expects? What are the error situations you plan on simulating?		
Usability	What are the usability issues about this item?		
Accessibility	What are the accessibility issues about this item?		
User Scenarios	How would a user typically use this item? What tests will you do to simulate user scenarios?		
Boundaries and Limits	What are the boundary conditions surrounding this item? What are the limits of this item? Max Values? Minimum Values?		
Special Code Profiling and Other Metrics			
Schedule	When?		
Code Paths and Sequences?	What are the different ways to invoke or activate this item? What are things you can do just before this item that are supposed to change the way it operates? What should NOT change the way it operates?		

18.20 Test Case Structure

Where will test cases be stored? What is the naming scheme? What is the organising structure? How do test cases correlate to the test plan?

RECOMMENDED:

The test case structure follows the area breakdown structure highlighted above. Test cases will be stored in the TCM database. TCM was chosen because it supports arbitrary depths of hierarchy, and because it SQL based - allowing a great deal of flexibility in reporting, database management, etc.

In TCM, the left pane holds the hierarchy, the right pane holds instances of test cases. The left pane will follow the hierarchy through all the levels of feature detail, and then add one more level to express test class types. For example, one might see the following hierarchy in the left pane:

Word
  Editing
    Format
      Font
        Typeface
          Data Validation
          Errors
          Boundaries
          Limits
          Etc.
        Style
        Etc.
      Paragraph

Test cases can exist at any level, but it is recommend that they be entered at the terminal level of the hierarchy so that they can be easily associated with similar classes of testing. When a given level is selected, the list of test cases associated with it is shown in the right pane.

This hierarchy allows one to avoid needing to follow numbering schemes (which are a pane to maintain and organised), allows you to express a test's location in the tree with a "item.item.item.item" naming convention, and allows one to determine to what degree different classes of tests are covered for each feature. It is recommended that if a class is being skipped that an explanatory entry be placed in the right pane justifying why tests of that class are not relevant.

The intent is to make this document drive the creation and evaluation of the test cases.

18.21 Spec Review Issues

Indicate location and method being used for reporting problems against the specification and design.

18.22 Test Tools

List whatever test tools will be used, or need to be written, and for what purpose. It is often best to point to an external location for more details, as tools usually require an entire plan and architectural summary of their own.

18.23 Smoke Test (acceptance test, build verification, etc.)

The smoke test determines whether or not the build is good enough to be submitted to testing. This section gives a statement of what the basic smoke test consists of, how it is design, and how it will be performed. A pointer to suite locations is helpful here too.

18.24 Automated Tests

What degree of automation will be used testing this area? What platform/tools will be used to write the automated tests? What will the automation focus on? Where are the automated tools, suites and sources checked in?

18.25 Manual Tests

What sorts of items will be tested manually rather than via automation? Why is manual testing being chosen over automation? Where are the manual tests defined and located?

18.26 Regression Tests

What is your general regression strategy? Are you going to automate? Where are the regressions stored? How often will they be re-run?

18.27 Bug Bashes

What is your strategy for bug bashes? How many? What goals? What incentives? What areas are targetted to be bashed? By who?

18.28 Bug Reporting

What tool(s) will be used to report bugs, and where are the bug reports located? Are there any special pieces of information regarding categorisation of bugs that should be reported here (areas, keywords, etc.)?

18.29 Plan Contingencies

Is there anything that may require testing's plans to change? Briefly describe how you plan to react to those changes.

18.30 External Dependencies

Are there any groups or projects external to the team that are dependent on your feature, or that your feature is dependent on? What testing problems and issues does this create? How are deliverables from external groups going to be tested and confirmed with your own feature? Who are the individuals serving as primary contact and liaison in this relationship?

18.31 Headcount Requirements

How many people will it require to implement these plans? Are there currently enough people on staff right now? What effect will hiring more or less people have (slip in schedule, quality, or something else?).

18.32 Product Support

What aspects of this feature have been a problem for support in the past? How are those problems being addressed? What aspects of this feature will likely cause future support problems? How are those problems being resolved? What testing methodology is being used to prevent future support problems? How is information being sent to support regarding problems and functionality of the feature?

18.33 Testing Schedule

Break the testing down into phases (ex. Planning, Case Design, Unit & Component Tests, Integration Tests, Stabilisation, Performance and Capacity Tuning, Full Pass and Shipping) - and make a rough schedule of sequence and dates. What tasks do you plan on having done in what phases? This is a brief, high level summary - just to set expectation that certain components will be worked on at certain times - and to indicate that the plan is taking project schedule concerns into consideration.

Include a pointer to more detailed feature and team schedules here.

18.34 Drop Procedures

Define the methodology for handing off the code between Development and Testing.

18.35 Release Procedures

Describe the step-wise process for getting the product from the network testing version to ready-to-ship master diskette sets.

18.36 Alias/Newsgroups and Communication Channels

List any email aliases and what they are for. List any bulletin boards, newsgroups, or other communication procedures and methodologies here.

18.37 Regular Meetings

For each meeting, when, where, what is general agenda.

- Feature Team Meetings
- Project Test Team Meetings
- Feature Team Test Meetings

18.38 Decisions Making Procedures

Who reviews decision points for the following sorts of things: build approval, bug triage, feature sign off, test plan sign off, development design sign off? What is the process for the various decisions?

19. Appendix 7 – Sample Test Case

19.1 Introduction

[If necessary, write the introduction of the **Test Cases**. It should provide an overview of the entire document, and include the purpose, scope, definitions, acronyms, abbreviations, references and overview of this set of **Test Cases**.]

19.1.1 Definitions, Acronyms and Abbreviations

[This subsection should provide the definitions of all terms, acronyms, and abbreviations required to interpret properly the **Test Cases**. This information may be provided by the project Glossary.]

19.1.2 References

[This subsection should provide a complete list of all documents referenced elsewhere in the **Test Cases**. Each document should be identified with a title, report number (if applicable), date, and publishing organisation. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

19.2 Testing Environments

19.2.1 Environment 1

Machine Name		DB Directory			
Version		DB	CISAM/Online/SE/…	Client Server /Back-End	
Tester Name			Test Date		
New Log	If necessary, list the new log after tests have been run.			State	Successful/Failed

19.2.2 Environment 2

[Repeat above chart as needed for environment 2.]

19.3 Setup Information (General Pre-conditions)

[Enter the prerequisites to execute the test cases (e.g. Flags, parameters, etc.)]

19.4 Test Cases

19.4.1 Test Case 1: <Test Case Name>

19.4.1.1 Description

[If necessary, write description text.]

19.4.1.2 Pre-conditions for this test case

[If necessary, write pre-conditions text.]

19.4.1.3 Scenario

Test Case						
UC Step	**Step Description**	**Expected Result**	**Actual Result** (if different from expected)	**Successful/Failed**	**Environment Nbr (if failed)**	**Log Number (if failed)**
			Test Case Status	Successful/Failed		

19.4.2 Array of values

[If necessary, fill an array of value.]

Array of values					
	Scenario 1	**Scenario 2**	**Scenario3**	**Scenario 4**	**Scenario 5**
Value1					
Value2					
...					
ValueX					
Expected Result					
Actual Result (if different from expected)					
Successful/Failed					
Environment Nbr (if failed)					
Log Number (if failed)					

19.5 Test Case 2: <Test Case Name>

[If necessary, repeat test case's previous section for each additional test case.]

20. Appendix 8 – Sample Quality Assurance Plan

20.1 Introduction

The introduction of the **Quality Assurance Plan** provides an overview of the entire document. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this **Quality Assurance Plan**.

20.1.1 Purpose

Specify the purpose of this **Quality Assurance Plan.**

20.1.2 Scope

A brief description of the scope of this **Quality Assurance Plan**; what Project(s) it is associated with and anything else that is affected or influenced by this document.

20.1.3 Definitions, Acronyms and Abbreviations

This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Quality Assurance Plan**. This information may be provided by reference to the project's Glossary.

20.1.4 References

This subsection provides a complete list of all documents referenced elsewhere in the **Quality Assurance Plan**. Identify each document by title, report number (if applicable), date, and publishing organisation. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document. For the **Quality Assurance Plan**, this should include:

- Documentation Plan
- Measurement Plan
- Test Plan
- Software Development Plan
- Problem Resolution Plan
- Configuration Management Plan
- Subcontractor Management Plan
- Risk Management Plan

20.1.5 Overview

This subsection describes what the rest of the **Quality Assurance Plan** contains and explains how the document is organised.

20.2 Quality Objectives

This section needs to reference the section of the Software Requirements Specification that deals with quality requirements.

20.3 Management

20.3.1 Organisation

Describe the structure of the organisation responsible for Quality Assurance. The Rational Unified Process recommends that the Software Engineering Process Authority (SEPA) be responsible for the process component of Quality Assurance. The Rational Unified Process further recommends that the evaluation of product be done within the project (most notably by an independent test team) and by joint customer/developer review.

20.3.2 Tasks and Responsibilities

Describe the various Quality Assurance tasks that will be carried out for this project and indicate how they are synchronised with the project's major and minor milestones. These tasks will include:

- Joint Reviews
- Process Audits
- Process Reviews
- Customer Audits

For each task, identify the team member responsible for its execution.

20.4 Documentation

Enclose the Documentation Plan artefact by reference.

Also, list the minimum documentation that must be produced during the project to ensure that the software product that is developed satisfies the requirements. The suggested minimum set is:

- Software Development Plan (SDP)
- Test Plan

- Iteration Plans
- Software Requirements Specification (SRS)
- Software Architecture Document
- User Documentation (for example, manuals, guides)
- Configuration Management Plan

Provide pointers to the Development Case to show where in the process the adequacy of these documents is evaluated.

20.5 Standards and Guidelines

This section references any standards and guidelines that are expected to be used on the project, and addresses how compliance with these standards and guidelines is to be determined. The relevant artefacts are enclosed by reference.

The suggested set is:

- Development Case
- Business Modelling Guidelines
- User-Interface Guidelines
- Use-Case Modelling Guidelines
- Design Guidelines
- Programming Guidelines
- Test Guidelines
- Manual Style Guide

20.6 Metrics

This section describes the product, project, and process metrics that are to be captured and monitored for the project. This is usually addressed by enclosing the Measurement Plan artefact by reference.

20.7 Review and Audit Plan

This section contains the Review and Audit Plan. The Review and Audit Plan specifies the schedule, resources, and methods and procedures to be used in conducting project reviews and audits. The plan details the various types of reviews and audits to be carried out during the project, and identifies any external agencies that are expected to approve or regulate the artefacts produced by the project.

This section should identify:

- Review and Audit Tasks

Describe briefly each type of review and audit that will be carried out on the project. For each type, identify the project artefacts that will be the subject of the review or audit. These may include Joint Customer-Developer Technical and Management Reviews, Process Reviews and Audits, Customer Audits, Internal Technical and Management Reviews.

- Schedule

Detail here the schedule for the reviews and audits. This should include reviews and audits scheduled at project milestones, as well as reviews that are triggered by delivery of project artefacts. This subsection may reference the project or iteration plan.

- Organisation and Responsibilities

List here the specific groups or individuals to be involved in each of the identified review and audit activities. Describe briefly the tasks and responsibilities of each. Also, list any external agencies that are expected to approve or regulate any product of the project.

- Problem Resolution and Corrective Action

This subsection describes the procedures for reporting and handling problems identified during project reviews and audits. The Problem Resolution Plan may be referenced.

- Tools, Techniques and Methodologies

Describe here any specific tools, techniques or methodologies that are to be used to carry out the review and audit activities identified in this plan. You should describe the explicit process to be followed for each type of review or audit. Your organisation may have a standard Review and Audit Procedures Manual, which may be referenced. These procedure descriptions should also address the collection, storage and archiving of the project's Review Records.

A suggested set of reviews and audits to use as a basis for planning is:

- Requirements Review (maps to the traditional Software Specification Review)
- Architecture Review (maps to the traditional Preliminary Design Review)
- Design Review (maps to the traditional Critical Design Review)

Note that the product-, technique-, criteria-, and metrics- related aspects of these reviews is addressed in the Rational Unified Process itself and instantiated in the Evaluation Plan section of the Software Development Plan. The Review and Audit Plan section of the **Quality Assurance Plan** will concern itself with the Joint (customer, developer) Review aspects, for example, artefacts required, responsibilities, conduct of the review meeting, pass or fail criteria.

- Functional Configuration audit (to verify all requirements in the SRS have been met)

- Physical configuration audit (to verify that the software and its documentation are complete and ready for delivery)
- Process audits
- Process reviews
- Managerial reviews (Project Approval Review, Project Planning Review, Iteration Plan Review, PRA Project Review)
- Post-mortem reviews (Iteration Acceptance Review, Lifecycle Milestone Review, Project Acceptance Review).

20.8 Evaluation and Test

This section references the Software Development Plan (Evaluation Plan section) and the Test Plan.

20.9 Problem Resolution and Corrective Action

This section references the Problem Resolution Plan.

20.10 Tools, Techniques, and Methodologies

A list of any tools, techniques and methodologies that are to be used when performing Quality Assurance activities.

20.11 Configuration Management

This section references the Configuration Management Plan.

20.12 Supplier and Subcontractor Controls

This section references the Subcontractor Management Plan.

20.13 Quality Records

Descriptions of the various quality records that will be maintained during the project, including how and where each type of record will be stored and for how long.

20.14 Training

List here any training activities necessary for the project team to achieve the needs of the **Quality Assurance Plan**.

20.15 Risk Management

This section references the Risk Management Plan.

21. Appendix 9 – Function Point Analysis

21.1 Introduction

This section is intended to provide a very brief introduction to the topic of Function Point Analysis (FPA). It is not exhaustive. Additional material is widely available elsewhere.

FPA is an technique used to estimate the complexity of an application and thus to estimate the likely duration of the project to develop it. It can be used to:

- To measure the productivity of your staff, your outsourcer or even yourself, and then track it over time
- To estimate project effort and schedule
- To measure your productivity and then compare it to other organisations
- To use the data to drive business decisions regarding which applications to retain, retire or redesign.

The two productivity metrics that get the most attention are function points implemented per staff month and function points implemented per calendar month.

Function points implemented per staff month indicates the amount of effort that goes into system development. A 500 function point application developed by 10 people in 10 months had a productivity rate of 5 function points per staff month.

Function points implemented per calendar month indicates the speed with which systems can be developed. Note that this quantity cannot be increased endlessly simply by adding staff. Software development suffers from the law of diminishing returns. Eventually adding staff will have no positive impact on schedule.

21.2 Components of Function Points

There are five components to estimating the number of function points in an application, divided into two types: Data and Transactional.

Data Functions

- Internal Logical Files (ILF)
- External Interface Files (EIF)

Transactional Functions

- External Inputs (EI)
- External Outputs (EO)
- External Inquiries (EQ)

21.2.1 Internal Logical Files

This data function type allows users to utilise data they are responsible for maintaining. For example, a pilot may enter navigational data through a display in the cockpit prior to departure. The data is stored in a file for use and can be modified during the mission. Therefore the pilot is responsible for maintaining the file that contains the navigational information. Logical groupings of data in a system, maintained by an end user, are referred to as Internal Logical Files (ILF).

Count each major logical group of user data or control information in the application as a logical internal file type. Include each logical file, or within a database, each logical group of data from the viewpoint of the user, that is generated, used, and maintained by the application. Count logical files as described in the external design, not physical files.

The logical internal file types should be classified within three levels of complexity as follows.

- **Simple**: Few record types. Few data element types. No significant performance or recovery considerations.
- **Average**: The logical internal file type is not clearly either simple or complex.
- **Complex**: Many record types. Many data element types. Performance and recovery are significant considerations.

Do not include logical internal files that are not accessible to the user through external input, output, interface file, or inquiry types.

21.2.2 External Interface Files

This data function type is related to logical groupings of data. In this case the user is not responsible for maintaining the data. The data resides in another system and is maintained by another user or system. The user of the system being counted requires this data for reference purposes only. For example, it may be necessary for a pilot to reference position data from a satellite or ground-based facility during flight. The pilot does not have the responsibility for updating data at these sites but must reference it during the flight. Groupings of data from another system that are used only for reference purposes are defined as External Interface Files (EIF).

Files passed or shared between applications should be counted as external interface file types within each application. Count each major logical group of user data or control information that enters or leaves the application, as an external interface file type. External interface file types should be classified within three levels of complexity, using definitions similar to those for logical internal file types.

Outgoing external interface file types should also be counted as logical internal file types for the application.

21.2.3 External Input

This transactional function type allows a user to maintain Internal Logical Files (ILFs) through the ability to add, change and delete the data. For example, a pilot can add, change and delete navigational information prior to and during the mission. In this case the pilot is utilising a transaction referred to as an External Input (EI). An External Input gives the user the capability to maintain the data in ILF's through adding, changing and deleting its contents.

Count each unique user data or user control input type that enters the external boundary of the application being measured, and adds or changes data in a logical internal file type. An external input type should be considered unique if it has a different format, or if the external design requires a processing logic different from other external input types of the same format. Include external input types that enter directly as transactions from the user, and those that enter as transactions from other applications, such as input files of transactions.
Each external input type should be classified within three levels of complexity, as follows:

- **Simple**: Few data element types are included in the external input type, and few logical internal file types are referenced by the external input type. User human factors considerations are not significant in the design of the external input type.

- **Average**: The external input type is not clearly either simple or complex.

- **Complex**: Many data element types are included in the external input type, and many logical internal file types are referenced by the external input type. User human factors considerations significantly affect the design of the external input type.

Do not include external input types that are introduced into the application only because of the technology used.

Do not include input files of records as external input types, because these are counted as external interface file types.

Do not include the input part of the external inquiry types as external input types, because these are counted as external inquiry types.

21.2.4 External Output

This transactional function type gives the user the ability to produce outputs. For example a pilot has the ability to separately display ground speed, true air speed and calibrated air speed. The results displayed are derived using data that is

maintained and data that is referenced. In function point terminology the resulting display is called an External Output (EO).
Count each unique user data or control output type that leaves the external boundary of the application being measured. An external output type should be considered unique if it has a different format, or if the external design requires a processing logic different from other external output types of the same format. Include external output types that leave directly as reports and messages to other applications, such as, output files of reports and messages.

Each external output type should be classified within three levels of complexity, using definitions similar to those for the external input types. For reports, the following additional complexity definitions should be used.

- **Simple**: One or two columns. Simple data element transformations.
- **Average**: Multiple columns with subtotals. Multiple data element transformations.
- **Complex**: Intricate data element transformations. Multiple and complex file references to be correlated. Significant performance considerations.

Do not include external output types that are introduced into the application only because of the technology used.

Do not include output files of records as external output types, because these are counted as external interface file types.

Do not include the output response of external inquiry types as external output types, because these are counted as external inquiry types.

21.2.5 External Inquiries

This transactional function type addresses the requirement to select and display specific data from files. To accomplish this a user inputs selection information that is used to retrieve data that meets the specific criteria. In this situation there is no manipulation of the data. It is a direct retrieval of information contained on the files. For example if a pilot displays terrain clearance data that was previously set, the resulting output is the direct retrieval of stored information. These transactions are referred to as External Inquiries (EQ).

Count each unique input/output combination, where an input caused and generates an immediate output, as an external inquiry type. An external inquiry type should be considered unique if it has a format different from other external inquiry types in either its input or output parts, or if the external design requires a processing logic different from other external inquiry types of the same format. Include external inquiry types that enter directly from the user, and those that enter from other applications.

The complexity of the external inquiry types should be classified as follows:

- Classify the input part of the external inquiry using definitions similar to the external input type.

- Classify the output part of the external inquiry type using definitions similar to the external output type.

- The complexity of the external inquiry type is the greater of the two classifications.

To help distinguish external inquiry types from external input types, consider that the input data of an external inquiry type is entered only to direct the search, and no update of logical internal file types should occur.

Do not confuse a query facility as an external inquiry type. An external inquiry type is a direct search for specific data, usually using only a single key. A query facility provides an organised structure of external input, output, and inquiry types to compose many possible inquiries using many keys and operations. These external input, output, and inquiry types should all be counted to measure a query facility.

21.3 Function Point Analysis

The function points measure count is performed in two general steps:

1. Classify and count the five user function types to derive an Unadjusted Function Point (UFP) count.
2. Adjust for processing complexity.

21.3.1 Calculating Unadjusted Function Point Count

The first step in measuring function points is to count and classify to three levels of complexity the Data and Transactional Function Types.

Use the multipliers in following matrix to derive the UFP:

Function Point Type	Simple	Average	Complex
Internal Logical Files (ILF)	7	10	15
External Interface Files (EIF)	5	7	10
External Inputs (EI)	3	4	6
External Outputs (EO)	4	5	7
External Inquiries (EQ)	3	4	6

Multiply the number of functions of each function type and complexity by the appropriate multiplier and total the values.

21.3.2 Adjusting UFP for Complexity

The procedure for adjusting the UFP to allow for complexity involves assigning one of six degrees of influence to each of 14 processing characteristics.

The six degree of influence values are:

0. Not Present or No Influence
1. Incidental influence
2. Moderate influence
3. Average influence
4. Significant influence
5. Strong influence throughout

The processing complexity characteristics are:

Characteristic	**Description**
Data Communications	The data and control information used in the application are sent or received over communication facilities. Workstations connected locally to the server are considered to use communication facilities.
Distributed Functions	Distributed data or processing functions are a characteristic of the application.
Performance	Application performance objectives, in either response or throughput, influenced the design, development, installation, and support of the application.
Heavily Used Configuration	A heavily used operational configuration is a characteristic of the application. The user wants to run the application on existing or committed equipment that will be heavily used.
Transaction Rate	The transaction rate is high and it influenced the design, development, installation, and support of the application.
On-line Data Entry	On-line data entry and control functions are provided in the application
End User Efficiency	The on-line functions provided, emphasise end user efficiency.
On-line Update	The application provides on-line update for the logical internal files.
Complex Processing	Complex processing is a characteristic of the application. Examples are: • many control interactions and decision points. • extensive logical and mathematical equations • much exception processing resulting in incomplete transactions that must be processed again.
Reusability	The application, and the code in the application, has been specifically designed, developed, and supported for reusability in other application, and at other sites.
Installation Ease	Conversion and installation ease are characteristics of the application. A conversion and installation plan was provided, and it was tested during the system test

	phase.
Operational Ease	Operational ease is a characteristic of the application. Effective start-up, back-up, and recovery procedures were provided, and they were tested during the system test phase. The application minimises the need for manual activities, such as, tape mounts, paper handling, and direct on-location manual intervention.
Multiple Sites	The application has been specifically designed, developed, and supported to be installed at multiple sites for multiple organisations.
Facilitate Change	The application has been specifically designed, developed, and supported to facilitate change. Examples are: • flexible query capability is provided • business information subject to change is grouped in tables maintainable by the user

The degree of influence values for each of the complexity factors is summed, generating a Total Degree of Influence (TDI) value. This is used to derive an adjustment factor ranging from 0.65 to 1.35 (corresponding to an adjustment of +/- 35 percent).

The Function Point count is calculated as follows:

FP = UFP x (0.65 + (TDI X 0.01))

21.3.3 Translating FPA Count Into Project Estimates

The FPA count can then be used to generate estimates of project resources, elapsed time and cost.

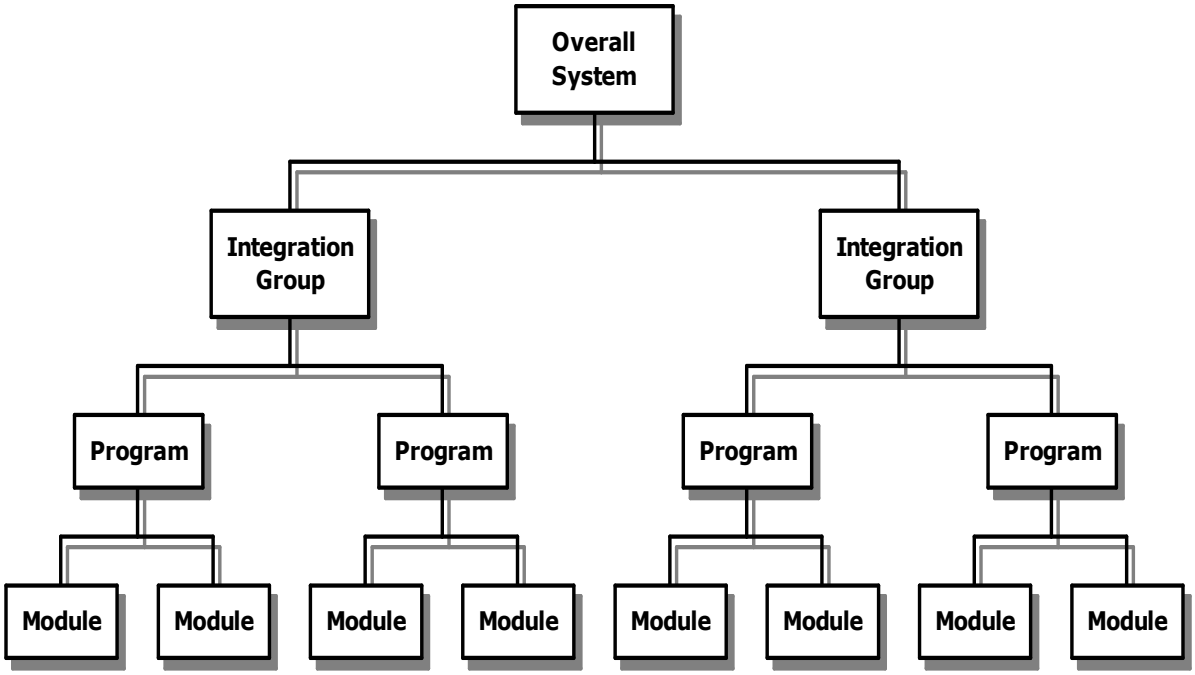
The FPA count represents the raw development work to be performed by the project team.

A development project will consist of the following phases:

- Requirements Definition
- Analysis
- Design
- Construction (including Development and Testing)
- Implementation

To this a project management overhead will be added.

The development portion of the project consists of developing clusters of modules, programs and integration groups.



Each Integration Group will have an FPA count. The Integration Group with the largest FPA count represents the largest single body of development work.

Each FPA corresponds to an amount of effort. This depends on a number of factors:

- The skill, experience and productivity of the development staff
- The development platform being used

There are benchmarks available on the average amount of time that developing an FPA takes.

In terms of lines of code, one FPA corresponds to the following estimated number of lines of code for the following development platforms:

- **Visual Basic** – 91
- **C** – 128
- **Java** – 29
- **4GL** - 16

21.4 Other Users of FPA

21.4.1 Managing Project Scope

Once a project has been approved and the function point count has been established, it becomes a relatively easy task to identify, track and communicate new and changing requirements. As requests come in from users for new displays or capabilities, function point counts are developed and applied against

the rate. This result is then used to determine the impact on budget and effort. The user and the project team can then determine the importance of the request against its impact on budget and schedule. At the conclusion of the project the final function point count can be evaluated against the initial estimate to determine the effectiveness of requirements gathering techniques. This analysis helps to identify opportunities to improve the requirements definition process.

21.4.2 Estimating Support Requirements

In addition to delivery productivity, function points can be used to evaluate the support requirements for maintaining systems. In this analysis, productivity is determined by calculating the number of function points one individual can support for a given system in a year (i.e. FP/FTE year). When compared with other systems, these rates help to identify which systems require the most support. The resulting analysis helps an organisation develop a maintenance and replacement strategy for those systems that have high maintenance requirements.

21.4.3 Using FPA to Make Application Retain, Retire, Replace and Design Decisions

A business has three possible courses of action for each of its applications. They can retain, retire, replace or redesign the application.

While the function point size of an application is a factor in this decision, there are usually several other factors to be considered.

- **Retain** - Applications that hold their value are usually retained. An application that was 1,000 function points when it was built, and still delivers about 1,000 function points of functionality to its users has held its value. Retained applications continue to be used and enhanced.

- **Retire** - Applications that are no longer useful to the company should be retired. A 1,000 function point system that no one uses anymore is really delivering no function points of functionality to its owners.

- **Replace** - Replacement is a common response to systems that have gotten out of alignment with the needs of its owners. A system that was 1,000 function points when it was built may be delivering very little functionality because of changes in the business. However, if the requirement for a 1,000 function point application still exists, then the application must be replaced with one that satisfies the requirements.

- **Redesign** - Some applications continue to deliver business value, but need to be changed for a primarily technical reason. Switching to a Web-based architecture is a common reason. In many cases, these applications have embedded in them business rules and policies that are not documented

elsewhere in the company. These applications are redesigned to function in their new production environments

21.5 Other Factors Affecting Estimation

There are a number of other factors that affect the effort and time required to develop or enhance an application.

Other Product Factors - There can be a big difference between two 1,000 function point applications. There are product factors that are not reflected in the size. These can impact the cost of developing a product. They also alter the value of the product.

- Precedentedness is a measure of the organisations experience with similar systems.
- Flexibility measures the organisations ability to relax requirements in order to ease software development effort. Jobs are harder when an application is novel and requirements are unchangeable.
- Required reliability is related to the impact of system failure. Data base size increases product complexity. There are other product complexity factors as well.
- There is a premium to be paid for developing reusable software. This is the flip side of the size savings that can be achieved through reuse. In the nominal case, reuse across a project will tend wash in terms of extra effort to develop vs. size savings.
- System development documentation can vary from insufficient to meet the needs of future life cycle phases to being very excessive for the life cycle needs. In the latter case, the documentation becomes part of the product. In addition to developing software, the team is developing some type of reading material for other parties.

Platform Difficulty - It is difficult enough to develop software under the best of circumstances. Platforms with memory, storage and processir limitations make the job that much more costly.

- Execution Time Constraint is based on the percentage of available execution time expected to be used by the system.
- Main Storage Constraint is based on the percentage of available storage expected to be used by the system. This constraint could raise the development effort by up to 56%.
- Platform, or Virtual Machine, refers to the complex of hardware and software that the application uses. It includes compilers, DBMSs and any other component. In the nominal case, there is a major change twice per year. Virtual Machine Stability can cut development effort by up to 13%. Environments that experience major changes every two weeks can expect a 30% increase in development effort.

Personnel Factors - These tend to be the second largest group of cost drivers.

- Team Cohesion is usually present when stakeholders share the same objectives and culture. However, differences can be overcome when stakeholders are willing to reconcile objectives. The stakeholders experience and familiarity in operating as a team are also relevant. In this case, team is intended to be bigger than development team.
- Personnel Continuity is related to turnover. 12% per year project personnel turnover is considered nominal. The value can range between 3% and 48%.
- Analyst and Programmer Capabilities measures the calibre of the people who make up the team. It addresses their analytical and programming capabilities. It does not consider specific experiences. It is usually measured by assigning a percentile of the respective team members relative to the IS community as a whole.
- Applications Experience is primarily the experience that the analysts have with the problem domain for which a system is being built. It can take 10 years for someone to become an expert in a business area. Most analysts have far less experience than this.
- Platform, Language and Tool Experience of the technical team has impact on the estimate. Because of the speed of technological change, most professionals only have a year of experience with the tools they are using for any given project. It can take three years to truly master any specific computer language or major set of software development tools.

Other Project Factors - The project equals the product plus the platform the personnel plus the process. The process followed might not be formalised, but it still impacts the project cost.

- Architecture/Risk Resolution involves the availability of top architects. The presence of risk management plans is also considered. In the nominal case, 17% of the development schedule will be dedicated to architecture.
- Process Maturity is a new cost driver. In theory, more mature organisations are more efficient.
- Facilities breaks down into the Use of Software Tools and Multisite Development.
- Required schedule is always an aea of difficulty. If a project is planned for 10 people for a year, it requires 120 person months of effort. If the decision is made to develop this application in nine months, how many people are needed? Some people might say that 14 (120/9) people would be enough. However, such schedule compression would increase effort by 22%. The project should require 16 (120*1.22/9) people.