

International procedure specification for Logistics Support Analysis LSA

S3000L-B6865-03000-00

Issue No. 1.1



Applicable to: All

S3000L-A-00-00-0000-00A-001A-A

Table of contents

The listed documents are included in Issue 1.1, dated 2014-07-01, of this publication.

Document title	Chapter	Applicable to
Introduction to the specification	Chap 1	All
Purpose	Chap 1.1	All
Scope	Chap 1.2	All
How to use the specification	Chap 1.3	All
Maintenance of the specification	Chap 1.4	All
General requirements	Chap 2	All
LSA business process	Chap 3	All
Configuration management in LSA	Chap 4	All
Influence on design	Chap 5	All
Human factors analysis	Chap 6	All
Results of FMEA/FMECA in LSA	Chap 7	All
Damage and special event analysis	Chap 8	All
Logistic related operations analysis	Chap 9	All
Development of a scheduled maintenance program	Chap 10	All
Level of repair analysis	Chap 11	All
Maintenance task analysis	Chap 12	All
Software support analysis	Chap 13	All
Life cycle cost considerations	Chap 14	All
Obsolescence analysis	Chap 15	All
In-service LSA	Chap 16	All
Disposal	Chap 17	All
Interrelation to other ASD specifications	Chap 18	All
Data model	Chap 19	All
Data exchange	Chap 20	All
Terms, abbreviations and acronyms	Chap 21	All
Terms, abbreviations and acronyms - Introduction	Chap 21.1	All
Terms, abbreviations and acronyms - Glossary of terms	Chap 21.2	All
Terms, abbreviations and acronyms - Abbreviations and acronyms	Chap 21.3	All
Data element list	Chap 22	All

Copyright and user agreement

1 Copyright

Copyright © 2010, 2014 AeroSpace and Defense Industries Association of Europe - ASD.

All rights reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage or retrieval system, except as may be expressly permitted by the copyright act or in writing by the publisher.

S3000L™ is a trade mark owned by ASD.

All correspondence and queries should be directed to:

ASD
10 Rue Montoyer
B-1000 Brussels
Belgium

2 Agreement for use of the specification S3000L™ suite of information

2.1 Definitions

S3000L™ suite of information means, but is not limited to:

- the International procedure specification for Logistics Support Analysis LSA - S3000L
- examples (eg XML instances, pdf files, style sheets) and schemas
- any other software or information under the heading "**S3000L™ suite of information**", available for download from www.s3000l.org

Copyright holder means AeroSpace and Defense Industries Association of Europe (ASD).

2.2 Notice to user

By using all or any portion of **S3000L™ suite of information** you accept the terms and conditions of this user agreement.

This user agreement is enforceable against you and any legal entity that has obtained **S3000L™ suite of information** or any portion thereof and on whose behalf it is used.

2.3 License to use

As long as you comply with the terms of this user agreement, the copyright holders grant to you a non-exclusive license to use **S3000L™ suite of information**.

2.4 Intellectual property rights

S3000L™ suite of information is the intellectual property of and is owned by the copyright holder. Except as expressly stated herein, this user agreement does not grant you any intellectual property right in the **S3000L™ suite of information** and all rights not expressly granted are reserved by the copyright holder.

2.5 No modifications

You must not modify, adapt or translate, in whole or in part, **S3000L™ suite of information**. You may however add business rules.

2.6 No warranty

S3000L™ suite of information is being delivered to you "as is". The copyright holder does not warrant the performance or result you may obtain by using **S3000L™ suite of information**. The copyright holder makes no warranties, representations or indemnities, express or implied, whether by statute, common law, custom, usage or otherwise as to any matter including without limitation merchantability, integration, satisfactory quality, fitness for any particular purpose, or non-infringement of third parties rights.

2.7 Limitation of liability

In no event will the copyright holder be liable to you for any damages, claims or costs whatsoever or any consequential, indirect or incidental damages, or any lost profits or lost savings or for any claim by a third party, even if the copyright holder has been advised of the possibility of such damages, claims, costs, lost profits or lost savings.

2.8 Indemnity

You agree to defend, indemnify, and hold harmless the copyright holder and its parents and affiliates and all of their employees, agents, directors, officers, proprietors, partners, representatives, shareholders, servants, attorneys, predecessors, successors, assigns, and those who have worked on the preparation, publication or distribution of the **S3000L™ suite of information** from and against any and all claims, proceedings, damages, injuries, liabilities, losses, costs, and expenses (including reasonable attorneys' fees and litigation expenses), relating to or arising from your use of the **S3000L™ suite of information** or any breach by you of this user agreement.

2.9 Governing law and arbitration

This user agreement will be governed by and construed in accordance with the laws of the Kingdom of Belgium.

In the event of any dispute, controversy or claim arising out of or in connection with this user agreement, or the breach, termination or invalidity thereof, the parties agree to submit the matter to settlement proceedings under the ICC (International Chamber of Commerce) ADR rules. If the dispute has not been settled pursuant to the said rules within 45 days following the filing of a request for ADR or within such other period as the parties may agree in writing, such dispute shall be finally settled under the rules of arbitration of the International Chamber of Commerce by three arbitrators appointed in accordance with the said rules of arbitration. All related proceedings should be at the place of the ICC in Paris, France.

The language to be used in the arbitral proceedings shall be English.

Chapter 1

Introduction to the specification

Table of content

Document

Chap 1	Introduction to the specification.....	S3000L-A-01-00-0000-00A-040A-A
Chap 1.1	Purpose	S3000L-A-01-01-0000-00A-040A-A
Chap 1.2	Scope	S3000L-A-01-02-0000-00A-040A-A
Chap 1.3	How to use the specification	S3000L-A-01-03-0000-00A-040A-A
Chap 1.4	Maintenance of the specification.....	S3000L-A-01-04-0000-00A-040A-A

Chapter 1.1

Purpose

Table of contents

	Page
Purpose	1
References.....	1
1 General	2
2 Purpose	2
3 Background.....	2

List of tables

1 References	1
-------------------------	---

References

Table 1 References

Chap No./Document No.	Title
DEF-STAN-00-60	Integrated Logistic Support - UK MoD
DEX1A&D	DEX1A&D - Aerospace and defense business DEX for exchange of product breakdown for support
DEX3A&D	DEX3A&D - Aerospace and defense business DEX for exchange of a task specification
GEIA-STD-0007	Logistics Product Data
ISO 10303-239 PLCS	Product Life Cycle Support (PLCS)
MIL-HDBK-502	Department of defense handbook acquisition logistics
MIL-STD-1388-1A	Logistics support analysis - DoD
MIL-STD-1388-2B	DoD requirements for a logistic support analysis record
S1000D	International specification for technical publications using a common source database
S2000M	S2000M - International specification for materiel management
S4000P	International specification for developing and continuously improving preventive maintenance
S5000F	Specification for operational and maintenance data feedback

1 General

This chapter gives a basic overview of the S3000L purpose including the history of the development.

2 Purpose

The Logistic Support Analysis (LSA) is one of the most important processes of product support.

It is the principal tool to:

- design the Products relevant to maintainability, reliability, testability and to optimize life cycle cost
- define all required resources to support the Product in its intended use, during in-service operation

S3000L defines the processes, general requirements and related information exchange governing the performance of LSA during the life cycle of aerospace, defense and commercial Products.

3 Background

The development of this specification originated within the Aerospace and Defence Industries Association of Europe (ASD) in 2005. At that time, MIL-STD 1388-1A had already been cancelled and DEF STAN 00-60 had become too unique to UK-MoD acquisitions to be applicable as an international handbook for general purposes.

The lack of a common valid procedure resulted in the development of various project specific LSA handbook versions and associate IT-solutions. For new projects (such as Eurofighter Typhoon, NH-90, Tiger, A400M and Gripen) these specific tasks needed accomplishing. This caused considerable effort for both industry and its military customers.

This situation prompted an initiative by ASD and the Aerospace Industry Association of America (AIA) to consider the joint development of a new common international specification for LSA.

This initiative started with an "Inaugural meeting on S3000L" held in Brussels on January 18th, 2006. The attendees gave their expectations on the need for an international LSA specification based on shortfalls in current standards, specifications and handbooks. It was agreed that there is not a shortfall of standards, there are too many of them.

In a subsequent project definition phase meeting in Munich in March 2006, the following basic requirements were identified:

The LSA specification will:

- generally be based on the processes of MIL-STD-1388-1A, MIL-HDBK-502, DEF-STAN-00-60 and the activity model given by ISO 10303-239 PLCS
- be the standard for creation and development of LSA data exchanged by:
 - DEX1A&D - Aerospace and defense business DEX for exchange of product breakdown for support
 - DEX3A&D - Aerospace and defense business DEX for exchange of a task specification
 - another exchange mechanism with the intention to address the relevant data from the MIL-STD-1388-2B for LSA purposes.
- use experience gained from the performance of LSA for the current programs such as Eurofighter Typhoon, NH-90, Tiger, A400M, Rafale, Gripen, JSF
- include process application guidelines and rules for information exchange
- be tailorable and include guidelines for tailoring
- take into account current ISO/EN baseline documents



- enable interfaces to the suite of the ASD specifications S1000D, S2000M, S4000P and the S5000F (the latter in preparation)

The development work was then allocated to an international team of experts working under the joint chairmanship of AIA and ASD representatives. The following companies/organizations contributed to the initial developing work:

- AgustaWestland	UK
- Airbus Deutschland GmbH	Germany
- Boeing	United States
- Dassault Aviation	France
- EADS Casa	Spain
- EADS Military Air Systems	Germany
- Eurocopter	France
- LOGSA	United States
- MBDA	France
- Saab AB	Sweden

The final draft of the specification S3000L (Issue 0.1) was officially published in June 2009. The main purpose of this draft was to enable experts from interested companies and organizations to provide comments on the first approach to the S3000L expert team. The commenting phase was officially closed by end of 2009. In this phase, experts from several nations contributed with their inputs to improve the final draft for the publication of the first official Issue 1.0.

In June 2010, Issue 1.0 of S3000L was finally released and published. With the signing of a Memorandum of Understanding between ASD and AIA at the Farnborough Air Show in July 2010, the ASD/AIA ILS Council was formed and the ILS community implemented a new platform for harmonization and coordination of the different ILS specification activities.

S3000L Issue 1.1 is the consequence of the continual review to maintain and improve the specification steadily and to harmonize it with the other specifications of the ASD/AIA ILS Specification Suite.

Chapter 1.2

Scope

Table of contents

	Page
Scope	1
References.....	1
1 Scope.....	1

List of tables

1 References	1
-------------------------	---

List of figures

1 The integrated logistics support functional elements	2
---	---

References

Table 1 References

Chap No./Document No.	Title
S1000D	International specification for technical publications using a common source database
S2000M	International specification for materiel management
S4000P	International specification for developing and continuously improving preventive maintenance

1 Scope

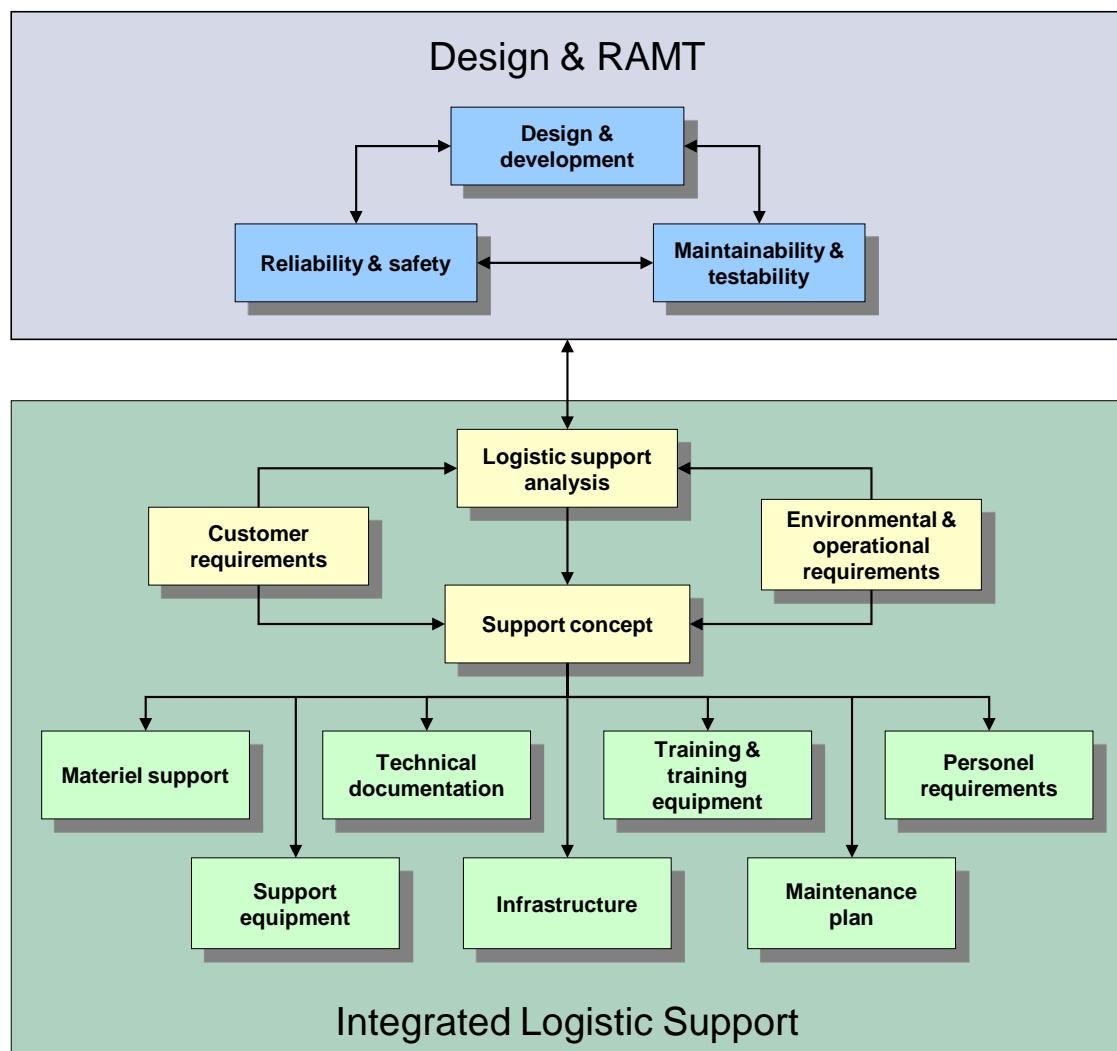
S3000L is designed to cover all processes and requirements governing the performance of LSA activities:

- It provides rules for the usage of the Product breakdown for logistic purposes and for the selection of LSA candidate items
- It describes type and methodology for performance of the specified analyses
- It gives guidelines on how to process the results of the analysis tasks and on how to achieve a cost-efficient support solution
- It covers the interface between LSA and the support engineering areas eg, Reliability, Availability, Maintainability and Testability (RAMT)
- It covers the interface between LSA and the ILS functional areas such as supply support, technical data services, special tools/test equipment or training. Refer to [Fig.1](#).

In particular, S3000L describes the interface between industry (contractor) and the customer, which, when based upon contractual agreements, will provide the typical deliverables of LSA as given in this specification. Examples for typical deliverables of an LSA process are:

- a reliable and maintainable Product because of influence on design from a logistics perspective
- cost efficient support system
- logistics product data
- ILS support products

Activities such as technical documentation or materiel management are described in detail within other specifications such as S1000D and S2000M respectively and are out of the scope of S3000L. This also applies for methodologies on carrying out technical/logistic analysis activities such as those described in S4000P, for example performance of scheduled maintenance analysis or detailed performance of Level Of Repair Analysis (LORA).



ICN-B6865-S3000L0001-002-01

Fig 1 The integrated logistics support functional elements

Chapter 1.3

How to use the specification

Table of contents

	Page
How to use the specification	1
References.....	1
1 General	2
2 Application	3
2.1 S3000L application policy.....	3
2.2 S3000L application	3
2.3 Tailoring of S3000L processes	3
3 Basic definitions.....	3
4 Acronyms.....	3
5 Organization of the specification	3
5.1 Chapter 1 - Introduction to the specification.....	3
5.2 Chapter 2 - General requirements.....	3
5.3 Chapter 3 - LSA business process	4
5.4 Chapter 4 - Configuration management in LSA	4
5.5 Chapter 5 - Influence on design	4
5.6 Chapter 6 - Human factors analysis	4
5.7 Chapter 7 - Results of FMEA/FMECA in LSA	4
5.8 Chapter 8 - Damage and special event analysis.....	4
5.9 Chapter 9 - Logistics related operations analysis.....	5
5.10 Chapter 10 - Development of a scheduled maintenance program.....	5
5.11 Chapter 11 - Level of repair analysis.....	5
5.12 Chapter 12 - Maintenance task analysis	5
5.13 Chapter 13 - Software support analysis	5
5.14 Chapter 14 - Life cycle cost considerations.....	5
5.15 Chapter 15 - Obsolescence analysis.....	6
5.16 Chapter 16 - In-service LSA	6
5.17 Chapter 17 - Disposal.....	6
5.18 Chapter 18 - Interrelation to other ASD specifications	6
5.19 Chapter 19 - Data model	6
5.20 Chapter 20 - Data exchange	7
5.21 Chapter 21 - Terms, abbreviations and acronyms	7
5.22 Chapter 22 - Data element list.....	7

List of tables

1 References	1
-------------------------	---

References

Table 1 References

Chap No./Document No.	Title
Chap 1	Introduction
Chap 2	General requirements

Applicable to: All

S3000L-A-01-03-0000-00A-040A-A

Chap 1.3

Chap No./Document No.	Title
<u>Chap 3</u>	LSA business process
<u>Chap 4</u>	Configuration management in LSA
<u>Chap 5</u>	Influence on design
<u>Chap 6</u>	Human factors analysis
<u>Chap 7</u>	Results of FMEA/FMECA in LSA
<u>Chap 8</u>	Damage and event analysis
<u>Chap 9</u>	Logistic related operations analysis
<u>Chap 10</u>	Development of a scheduled maintenance program
<u>Chap 11</u>	Level of repair analysis
<u>Chap 12</u>	Maintenance task analysis
<u>Chap 13</u>	Software support analysis
<u>Chap 14</u>	Life cycle cost considerations
<u>Chap 15</u>	Obsolescence analysis
<u>Chap 16</u>	In-service LSA
<u>Chap 17</u>	Disposal
<u>Chap 18</u>	Interrelation to other ASD specifications
<u>Chap 19</u>	Data model
<u>Chap 20</u>	Data exchange
<u>Chap 21</u>	Terms, abbreviations and acronyms
<u>Chap 22</u>	Data element list
S1000D	International specification for technical publications using a common source database
S2000M	International specification for materiel management
S4000P	International specification for developing and continuously improving preventive maintenance
S5000F	International specification for operational and maintenance data feedback
ISO 10303-239 PLCS	Product Life Cycle Support (PLCS)

1 General

This chapter gives an overview of:

- how to apply S3000L to a project
- the organization of the specification
- the fundamental reading rules

2

Application

2.1

S3000L application policy

It is intended that S3000L will be the common Logistic Support Analysis (LSA) specification to be used by customers and contractors (eg, governments, procurement authorities, support agencies and industry). By agreement between customer and contractor, S3000L can be supplemented by additional international or national requirements for specific projects. The use of the specification and any supplemental processes is subject of the contractual agreement between the customer and contractor.

2.2

S3000L application

At the start of any project using S3000L, it is necessary for the customer and contractor to agree how the specification will be implemented and to define and agree the variables and options that S3000L provides. These agreements must be recorded in the project's Guidance Conference Document (GCD). The process used by the customer and contractor to establish the information included in the GCD is known as the "LSA Guidance Conference". Refer to [Chap 3](#).

In order to supplement the GCD, the definition of a project interchange specification (data and other deliverables) will be required. The complexity of the project will determine whether this is a stand-alone specification, or integrated within the GCD.

2.3

Tailoring processes

In order to ensure efficient application, S3000L has been designed to help users select the functionality that is appropriate to their specific projects.

Individual chapters can be included or excluded. Tailoring can also be applied to the number and range of LSA candidate items, the number of analyses per candidate item and to data to be used during an analysis.

3

Basic definitions

Product	Any platform, system or equipment (air, sea, land vehicle, equipment or facility, civil or military)
project	The task to develop, maintain and dispose of the Product

4

Acronyms

S3000L uses many terms, abbreviations and acronyms that are specific to LSA. Refer to [Chap 21](#).

5

Organization of the specification

S3000L is organized into chapters, each dealing with the individual aspects of LSA.

5.1

Chapter 1 - Introduction to the specification

[Chap 1](#) provides a summarized view on purpose, background, scope and application of S3000L. It also explains the rules regarding the maintenance of S3000L and how to access the specification.

5.2

Chapter 2 - General requirements

[Chap 2](#) provides general information regarding the set-up and administration of an LSA program as part of an overall development and Integrated Logistic Support (ILS) program. It describes the interfaces, dependencies and time constraints in regard to the other ILS disciplines within a typical project scenario.

5.3

Chapter 3 - LSA business process

[Chap 3](#) is designed to provide a full, in-depth description of the complete LSA process. It starts with establishing Product usage data by the customer and LSA significant Product design and performance data by the contractor. These are joint inputs for the LSA Guidance Conference, which establishes agreed rules regarding type and depth of breakdown and on methodology of LSA candidate item selection and identification.

[Chap 3](#) further defines the type and methodology of the potential analysis activities and acceptance criteria for the analysis results. It covers the customer involvement in the LSA business process and the execution of an LSA review conference. This is the final step in accepting the LSA results and determining the support concept. It also starts the production of ILS elements to support operation.

5.4

Chapter 4 - Configuration management in LSA

Configuration management is the discipline that ensures the proper identification and versioning of the Product configuration, including Product breakdown. It controls changes, and records the change implementation status of the physical and functional characteristics of the Product.

Configuration management is the means through which integrity and continuity of the design, systems engineering and supportability are recorded, communicated, and controlled. Configuration management efforts provide a complete audit traceability of decisions and design modifications.

[Chap 4](#) defines the details and methodology of how the principles of configuration management are applied to the LSA process.

5.5

Chapter 5 - Influence on design

Influence on design is the goal of an industrial activity known as supportability engineering. Supportability engineering influences those Product characteristics which are vital for enabling the operation of the Product according to performance and design requirements and to minimize Life Cycle Cost (LCC).

The Product characteristics to be influenced are primarily Reliability, Maintainability and Testability. The tools for supportability engineering are early Reliability, Maintainability and Testability activities/programs and the performance of LSA.

Supportability engineering acts as the enabler between design/development and ILS.

[Chap 5](#) describes how the methods of supportability engineering are applied and how the results are linked to the LSA process.

5.6

Chapter 6 - Human factors analysis

[Chap 6](#) describes the relationship and integration between human factors engineering and the support analysis process. Like all other supportability engineering functions, human factors analysis provide source data that is used by the support analysis team to determine maintenance crew and support equipment requirements. This relationship begins in the design review process and continues through the development of the maintenance task analysis.

5.7

Chapter 7 - Results of FMEA/FMECA in LSA

[Chap 7](#) covers the methodology and decision logic that is applied to identify corrective maintenance tasks to be applied to the Product in case of a failure occurrence during normal use.

5.8

Chapter 8 - Damage and special event analysis

[Chap 8](#) covers the methodology for identifying and justifying those maintenance tasks which are appropriate in case of special events or damage found on the Product during general or detailed inspections.

5.9

Chapter 9 - Logistics related operations analysis

[Chap 9](#) covers the methodology for identifying and justifying logistic related operations tasks. These are tasks which can neither be assigned directly to the area of direct usage of a Product (eg, documented in a user guide) nor to the area of maintenance (eg, documented in a maintenance handbook). However, they are accomplished in operations and maintenance facilities in support of normal operation (eg, packaging, handling, storage, mooring).

These tasks can be very important for the proper usage of any Product. There are many operational aspects (eg, ease of operation, usability, flexibility of usage, mobility) that are restricted by logistics related operations tasks.

5.10

Chapter 10 - Development of a scheduled maintenance program

The identification of required preventive maintenance is of vital importance for the operation of a complex Product. For this purpose, Preventive Maintenance Task Requirements (PMTR) are developed by analysis methods given by, for example, S4000P. After the identification of the PMTRs, the development of a scheduled maintenance program is required, which results in maintenance task packages performed at specific intervals.

[Chap 10](#) gives rules for the transformation of PMTRs into LSA tasks and the grouping of the LSA tasks based on appropriate intervals/thresholds. Additional rules are given for a required adaptation of PMTR intervals/thresholds during the grouping process and for the final completion of a maintenance task package.

5.11

Chapter 11 - Level of repair analysis

[Chap 11](#) describes the actions taken when performing a Level of Repair Analysis (LORA) as part of any acquisition program. The requirement for a LORA is usually located in logistic support plans or similar documents.

LORA is a stand-alone activity that establishes an optimized maintenance or support concept using technical, commercial, operational and environmental data. A LORA can be performed on each LSA candidate item to determine the most cost-effective method to restore each item to its fully serviceable condition.

5.12

Chapter 12 - Maintenance task analysis

[Chap 12](#) describes how to analyze an identified maintenance task in terms of its logistic requirements including spare parts and consumables, support equipment, personnel, facilities and task duration information. Additional information such as task criticality, training needs, pre- and post-conditions, safety and environmental requirements are also considered.

5.13

Chapter 13 - Software support analysis

[Chap 13](#) provides guidance on dealing with the specific requirements concerning software in the environment of maintenance and operation of technical systems. Additionally, the interrelation between software and hardware will be defined clearly and an explanation will be given on how to integrate software aspects in the overall LSA process.

In modern technical Products, software aspects are of increasing significance. More and more Product functions are performed by complex software packages. For hardware components, concepts and processes are established to guarantee a proper supportability of the system during its entire life cycle. These concepts and processes are documented in the LSA process.

Software has the same requirement and is of equal importance for the proper function of a Product. Therefore, an analysis methodology is be applied to develop an adequate software support concept, called Software Support Analysis (SSA).

5.14

Chapter 14 - Life cycle cost considerations

LCC consists of all the direct costs plus any indirect-variable costs associated with the procurement, operations, support and disposal of the Product throughout its life-cycle.

[Chap 14](#) provides an overview of the LCC aspects and indicates which information, developed by an LSA process, can be used to support the LCC analysis activities, and vice versa.

5.15

Chapter 15 - Obsolescence analysis

[Chap 15](#) describes the relationship between obsolescence analysis and the support analysis process.

In the design and development process it is the goal of LSA to avoid/control the use of components and materiel that are likely to result in obsolescence in the early operational phase.

During the in-service phase, LSA will be involved in the analysis of obsolescence events and will contribute to the definition of economic alternatives with regard to the maintenance and the support concept.

5.16

Chapter 16 - In-service LSA

[Chap 16](#) gives an overview about LSA activities after the design and development phase. To ensure the continuous maintainability and operability of a Product during its in-service phase, the principles of LSA are applied. Feedback information and data from the Product user enables a comparison of initial assumptions based on analysis activities with real in-service experience and data. Technical and logistic analysis results and the corresponding decisions for a maintenance solution are continuously scrutinized throughout the whole lifetime of a Product. Amongst others, this approach enables industry to manage performance based contracts.

Refer also to S5000F.

5.17

Chapter 17 - Disposal

[Chap 17](#) deals with the disposal of Products and Product components, eg, their disposal from active and/or passive operation. Although disposal of Products normally occurs during the last phase of the life cycle, disposal must be considered in the early phases of every program's planning when acquiring a Product.

Adequate disposal activities can include:

- destruction/neutralization of toxic substances
- enabling a sustainable development, by recycling materials
- demilitarization of defense systems, to avoid weapons proliferation

5.18

Chapter 18 - Interrelation to other ASD specifications

[Chap 18](#) addresses the benefits of using the ASD suite of ILS Specifications in conjunction with this specification. The LSA relationships to the following specifications are addressed:

- S1000D - International specification for technical publications using a common source database
- S2000M - International specification for materiel management
- S4000P - International specification for developing and continuously improving preventive maintenance
- S5000F - International specification for operational and maintenance data feedback

5.19

Chapter 19 - Data model

[Chap 19](#) defines a coherent data model supporting the S3000L LSA process and its interaction with the related business processes. These processes can either be dependent upon data coming out from LSA or business processes that provide input to the LSA process. Additionally, the chapter defines all data elements that are used in the S3000L data model.

5.20 Chapter 20 - Data exchange

[Chap 20](#) defines how exchange of data (electronically) will be realized using Extensible Markup Language (XML).

[Chap 20](#) covers:

- exchange of the Product data needed for the performance of an LSA
- exchange of task set data needed by eg, technical publications and maintenance management
- exchange of LSA activities selection data
- exchange of LSA Failure Mode Effect Analysis (LSA-FMEA) data

5.21 Chapter 21 - Terms, abbreviations and acronyms

[Chap 21](#) provides a glossary of terms, abbreviations and acronyms used within S3000L.

5.22 Chapter 22 - Data element list

[Chap 22](#) lists all the data elements that are used in the S3000L data model and in the S3000L data exchange specifications.

The data element list is organized alphabetically by the data element name, and contains:

- Data element name
- Data element data type
- Data element definition, contains a textual definition and a list of valid values
- Class name, identifies Classes in the S3000L data model where the data element is used as a property
- Unit of Functionality, identifies in which paragraph of [Chap 19](#) the Class is defined

Chapter 1.4

Maintenance of the specification

Table of contents

	Page
Maintenance of the specification.....	1
References.....	1
1 Maintenance of the specification	1
1.1 General information	1
1.2 WEB portal.....	2

List of tables

1 References	1
-------------------------	---

List of figures

1 Login screen for ASD/AIA specification maintenance WEB portal.....	2
2 Sign up for an account.....	2

References

Table 1 References

Chap No./Document No.	Title
DEX1A&D	Aerospace and Defense Product Breakdown for Support
DEX3A&D	Aerospace and Defense Task Set
S1000D	International specification for technical publications using a common source database
S1003X	S1000D and S3000L interface specification

1 Maintenance of the specification

1.1 General information

The S3000L suite of information is maintained by the S3000L Steering Committee which comprises experts from member nations and organizations of ASD and AIA.

The suite consists of the following parts:

- S3000L - International procedure specification for Logistics Support Analysis, Issue 1.0 and Issue 1.1
- DEX1A&D - Aerospace and defense business DEX for exchange of product breakdown for support (limited to S3000L, Issue 1.0)
- DEX3A&D - Aerospace and defense business DEX for exchange of a task specification (limited to S3000L, Issue 1.0)
- S1003X - S1000D and S3000L interface specification (limited to S3000L, Issue 1.0 and S1000D, Issue 4.0.1)

1.2

WEB portal

A common process for change proposals and requests for clarification is implemented for all specifications within the ASD/AIA ILS Specification Suite. This process is realized by an interactive WEB portal. The link to access the portal is www.SX000i.org/cpf.

At the first access to the WEB portal the signup for a login username is required. For this purpose, the function “*Signup for a new account*” must be used. Refer to [Fig 1](#).



Login	
Username	<input type="text"/>
Password	<input type="password"/>
Remember my login in this browser	<input type="checkbox"/>
Secure Session	<input type="checkbox"/> Only allow your session to be used from this IP address.
<input type="button" value="Login"/>	
[Signup for a new account] [Lost your password?]	

ICN-B6865-S3000L0113-001-01

Fig 1 Login screen for ASD/AIA specification maintenance WEB portal

To create a user account follow the instructions on the screen after selecting “*Signup for a new account*”. Enter your favored username and your email address, confirm the control code and click on the “*Signup*” button. Refer to [Fig 2](#).



Signup	
Username:	<input type="text"/>
E-mail:	<input type="text"/>
Enter the code as it is shown in the box on the right:	<input type="text"/> 
On completion of this form and verification of your answers, you will be sent a confirmation e-mail to the e-mail address you specified. Using the confirmation e-mail, you will be able to activate your account. If you fail to activate your account within seven days, it will be purged. You must specify a valid e-mail address in order to receive the account confirmation e-mail.	
<input type="button" value="Signup"/>	
[Login] [Lost your password?]	

ICN-B6865-S3000L0114-001-01

Fig 2 Sign up for an account

The sign up will be confirmed by email and the creation of a password is requested. To complete the sign up process, the password must be assigned to the new user account, the provided email contains a corresponding link. After the successful password assignment, the login to the WEB portal is enabled and change proposals or requests for clarification can be reported as “issues” to the ASD/AIA ILS community.

Chapter 2

General requirements

Table of contents

	Page
General requirements.....	1
References	2
1 General	2
1.1 Introduction	2
1.2 Objective.....	3
1.3 Scope	3
2 Logistics support analysis program	3
2.1 Implementation	3
2.1.1 LSA activities and performance	3
2.1.2 Develop an early LSA strategy.....	4
2.2 Program management principles	5
2.2.1 Integrated logistics support management	5
2.2.2 Operating logistics organizational structures	5
2.2.3 Formal logistics organizations.....	5
2.3 Logistics management objectives and policies	6
2.3.1 Logistics management objectives.....	6
2.3.2 Logistics management policies	6
2.4 LSA program plan.....	6
2.4.1 Organizational requirements.....	6
2.4.2 General requirements	7
2.4.3 Functional requirements identification	8
2.4.4 Unique functional requirements	9
2.4.5 LSA program plan - generic example	9
2.4.6 LSA Guidance conference document - generic example.....	12
2.5 LSA program organization.....	14
2.6 LSA management responsibilities	14
2.6.1 The LSA manager.....	14
2.6.2 Program LSA managers and leads	15
2.6.3 Program technical staff (logistics engineers and analysts)	15
2.6.4 Supportability analysis integrated product team	15
3 Product development organization	16
3.1 Integrated product teams.....	16
3.2 Multidiscipline teams	16
3.3 Single point accountability	17
3.4 Empowerment	17
3.5 Management plans	17
3.6 Clear product definition and interfaces	17
3.7 Disciplined processes	17
3.8 Effective communications	17
3.9 Performance metrics	18

List of tables

1 References.....	2
------------------------	---

List of figures

1	Simplified LSA process	4
2	Example of content of an LSA Program Plan	10
3	Example of content of a Guidance Conference Document	12

References

Table 1 References

Chap No./Document No.	Title
Chap 20	Data exchange
MIL-STD 1388-2B	Military Standard - DoD requirements for a logistic support analysis record

1 General Introduction

1.1

The Logistic Support Analysis (LSA) program is considered to be a subset of Integrated Logistic Support (ILS). ILS has overall responsibility for the development of technical information and the support environment that will be used to support a Product throughout its intended life cycle.

The different disciplines in the context of supportability (eg technical documentation, spare parts, support equipment, personnel and training) need to be harmonized. The main disciplines are:

- Design interface
- Supply support and provisioning
- Support and test equipment
- Technical data/technical documentation
- Personnel and manpower
- IT/Software support
- Facilities
- Scheduled maintenance/maintenance planning
- Packaging, Handling, Storage and Transportation (PHST)
- Training and training devices

The LSA program is the principal source of technical data for ILS planning and resource decision. LSA will be used:

- To link product design and ILS requirements to product readiness thresholds and to define detailed support element requirements
- Throughout the acquisition cycle to assess and alter product design and to establish and update support element requirements
- As an important source of design related data for determining and integrating all logistics support requirements, for analyzing alternative design, operational, and support concepts, and for conducting tradeoffs among design and the various elements of logistics support of technical data for ILS planning and resource decisions

The logistic analysis activities results and support resource data will be stored in the LSA database to give all program elements access to a common data source for evaluating the supportability of design alternatives and objectives. The integrated product teams' practices will ensure the timely interaction of the LSA activities and data for all elements of the design process.

1.2

Objective

ILS supported by an embedded LSA process ensures there is an open exchange of information to and between the logistic disciplines and to the design. It pursues two thrusts simultaneously:

- Design for support**

A focus on designs that minimize operation, maintenance, training, support tasks, and life-cycle costs while optimizing operational readiness.

- Design of support**

The design, development, funding, test, and acquisition of all support resources needed to assure optimum performance and readiness of the Product in its intended operational environment and mission profiles.

1.3

Scope

This chapter is directed at ILS managers and LSA managers for both the customer and the contractor. The understanding of how an LSA program needs to be integrated into the superordinate ILS program is of crucial importance. The approach for implementing an LSA program varies across organizations and sometimes across projects within an organization. Regardless of the organization or project, there must be policies in place to establish the functional responsibilities and expectations of the LSA program.

2

Logistics support analysis program

LSA is a systematic iterative process that integrates product design and support system requirements and evaluates supportability requirements relative to achieving the specified system/component sustainability and availability requirements. The analyses documentation process is optimized by employing an electronic database. The logistics relevant information consists of the identification, optimization, and traceability of ILS resources (eg spares, manpower, personnel, training, support and test equipment, training and training equipment, facilities, PHST and technical documentation).

2.1

Implementation

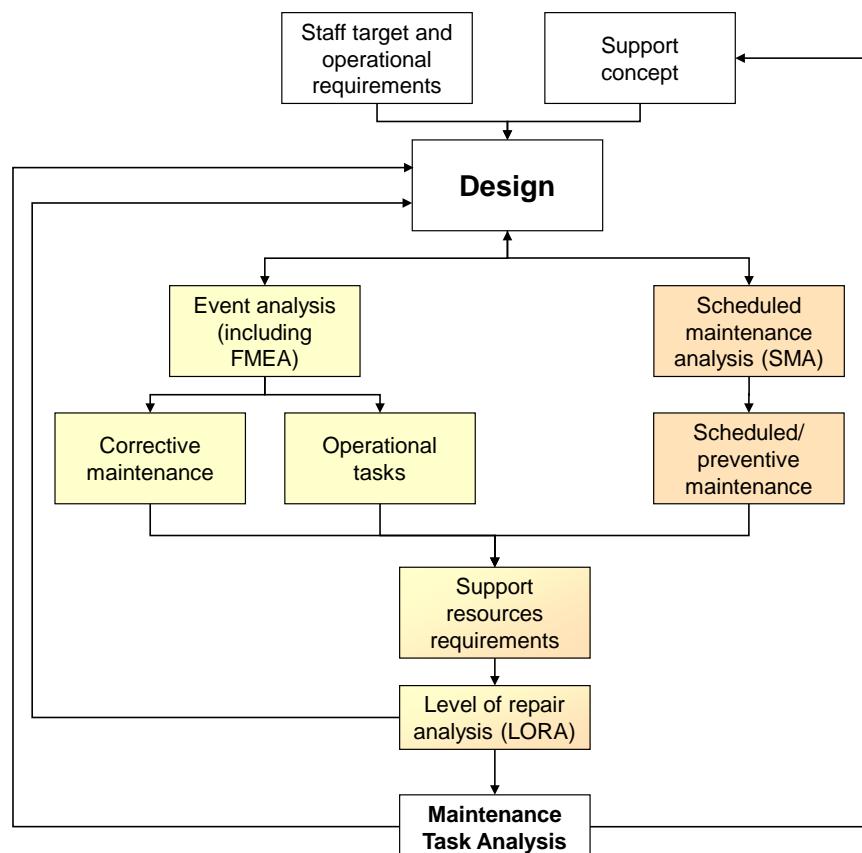
Design reviews and technical information meetings are conducted throughout the program. Host design reviews and conduct in-house design reviews to ensure that Design Program goals and objectives are being achieved. The cognizant LSA analysts and integrated product support personnel will be aware of supportability and supportability related design issues through direct participation in design reviews and/or by receiving all design review meeting minutes. Supportability concerns involving equipment design that arises during meetings, or throughout the overall and continuous design process, will be brought to the attention of the cognizant integrated product engineers. All design related supportability issues will be documented, including status, as necessary to ensure that a timely and appropriate resolution is obtained. LSA reviews are conducted in conjunction with scheduled design reviews.

2.1.1

LSA activities and performance

The LSA process (refer to [Fig 1](#)) includes the application of selected quantitative methods, to aid in the:

- Initial determination and establishment of logistic criteria as an input to system design**
- Evaluation of various design alternatives**
- Identification and provisioning of logistic support elements**
- Final assessment of the system support capability during use**



ICN-B6865-S3000L0095-003-01

Fig 1 Simplified LSA process

The LSA program is based on an integrated product definition concept and can be based on the following major elements:

- An LSA program plan that identifies all the required LSA activities that must be performed in order to influence design for supportability and determines the appropriate logistic resources
- Scheduling that identifies the timing of the LSA requirements. The LSA schedules are based on project phase needs and established to be mutually beneficial and supportive of other project requirements
- Assignment of responsibilities for performance of LSA activities to the design, supportability, and ILS personnel skilled and qualified for the activity
- Effective management of a wide range of design, supportability, and ILS disciplines

2.1.2

Develop an early LSA strategy

A proposed LSA strategy to be performed early in the acquisition program needs to be developed. This strategy will identify the scope of the proposed supportability objectives for the system/equipment, and the qualitative analysis that will be performed to provide the best cost benefit for the acquisition program. The LSA requirements will be analyzed in order to establish a comprehensive LSA program, including quantitative, qualitative and test LSA requirements for the systems, subsystems, and components comprising the overall Product. The resultant LSA program is a tailored cost effective program that accomplishes each customer requested procedure and provides a system that meets or exceeds each customer LSA requirement.

2.2 Program management principles

The logistics team is responsible for development of the support system. This team is a subset of the integrated product team responsible for the design, development, manufacture, operations, and support of a specific Product.

2.2.1 Integrated logistics support management

The maintenance of an interface with the LSA program manager (LSA manager) to provide continuity between program management direction and the development of the LSA results is of crucial importance. Maintaining a proper interface between the LSA program manager/lead and other functional management ensures a smooth execution of the LSA program.

The ILS manager is responsible for providing logistics and supportability expertise and resources to the program teams. In the product development process, each integrated product team leader controls the budget and is held accountable for development of that team's products. Consequently, most logistics and supportability personnel are assigned to, and work with, the integrated product teams. Logistics and supportability resources are provided to the integrated product teams to:

- Incorporate supportability into the design of the system and support system by providing reliability, maintainability, testability, human engineering, integrated diagnostics, and environmental suitability resources
- Develop the support system and training system, and plan logistics resources by providing supply support, support equipment, technical data, facilities, PHST, manpower and personnel, and training and training equipment personnel
- Accomplish logistics engineering in the product development systems engineering and design process
- Provide for LSA, standardization, interchangeability and interoperability, and environmental assurance

2.2.2 Operating logistics organizational structures

The operating organizational structure for a support program can be aligned with the Work Breakdown Structure (WBS). The hierarchical characteristics of the WBS can be used to establish lines of responsibility, authority, accountability and reporting chains from the lowest level member to the program manager. Integration of logistics into the depicted organization is designed to:

- Reflect each logistics and supportability element identified by the WBS and system specification
- Provide effective interfaces for logistics-related design functions most important to developing the support system, and achieving readiness goals

Each ILS manager is vested with total responsibility for all aspects of that team's products. Product team leaders at each organizational level combine the responsibilities of all subordinate product teams. This responsibility and reporting chain continues to the point that the program manager, as leader of the integrated product team, has responsibility for all products. A significant feature of this organization is every Product has a single point of responsibility, authority and accountability in the organization.

2.2.3 Formal logistics organizations

Functional leaders such as product support, production, and engineering department heads are members of the product team and provide support to the program manager. In that capacity, these leaders bring functional expertise and resources to assist in program execution and ensure uniform, timely assignment of needed skills to the product teams to assist in program execution and ensure uniform, timely assignment of added skill to the integrated product teams.

2.3 Logistics management objectives and policies

2.3.1 Logistics management objectives

Integration of logistics and supportability into the integrated product development program organization ensures:

- Design reflects test data assessment, supportability alternatives and tradeoff evaluations
- Detailed specification requirements
- Logistic resource planning is adjusted as necessary
- Operational availability and readiness thresholds are met
- The item is supportable in the expected operational environment
- The operational environments is/are accurately assessed
- The Support System achieves expected performance

An underlying logistic program objective is to identify and resolve supportability technical risk issues early, prior to beginning production and deployment of the Product.

2.3.2 Logistics management policies

To achieve logistics management objectives, a proper organization must be established for design-for-supportability through the integration of design and the development of the support system and training system. This will be achieved by:

- Structuring an integrated product development organization which provides for active logistics participation and influence on design
- Structuring the ILS process to be continuously interactive, on a working level, with design engineering through the system engineering process
- Planning to work closely with customers and suppliers to develop the Product

Additionally, established an LSA process which:

- Provides the logistic analysis procedures for integrating supportability requirements into the baseline design
- Requires the support system configuration match the product design configuration
- Provides the detailed maintenance planning and bottoms-up identification of total logistics resource requirements

As a mean of controlling, establish an ILS program progress, status and management reporting system which will document that program design, development, test and evaluation, and transition accomplishments meet or exceed logistics priorities and developing supportability requirements.

2.4 LSA program plan

2.4.1 Organizational requirements

The LSA Program Plan (LSA PP) describes the strategy for developing the LSA activities during the engineering and manufacturing development phase of the complete program. It identifies and integrates the LSA activities, identifies management responsibilities and activities, and establishes the approach for accomplishing the analysis activities. This plan provides a roadmap of what LSA will be performed and how it will be accomplished. Furthermore, this plan provides an approach to LSA and a basis to measure progress throughout the various phases of the program. The LSA process is iterative and dynamic in nature, therefore this plan will be updated to reflect current program status and planned changes.

This LSA PP describes the LSA process and the management structure established to execute this process. It details how the LSA process will be accomplished to satisfy the intended requirements. LSA is an iterative process that usually continues for as long as the item under analysis is in continuous use. Once the item has been retired, the LSA data can be used as baseline comparison data for future supported items.

The LSA PP describes the management, organizations, and procedures required to accomplish program requirements. It includes the following:

- Description, interrelationships, and schedule of procedures to be accomplished by each organizational element concerned in support of the LSA program
- Statement of requirements for LSA program participants (eg subcontractors, suppliers, partner companies, agencies)
- Identification of LSA program monitoring points
- Dissemination of LSA requirements to design and description of techniques to be employed to ensure that desired LSA is inherent in the product design
- Description of the planned approach to LSA activities, and specific methods and sources to be employed to satisfy qualitative supportability analysis and resolve design issues
- Description of the LSA database, WBS, Breakdown element identification numbering system, and handling government furnished information
- Description of how problems will be identified, controlled, and reported
- Description of provisions for updating the plan

Planning and coordination of LSA is the responsibility of logistics engineering management. LSA managers provide technical guidance in conducting LSA and have special analytical skills required to accomplish LSA-peculiar activities. Procedures must ensure supportability requirements are identified as an integral part of systems engineering and design. Supportability design baselines identify:

- Quantified reliability and maintainability comparative analysis requirements, like:
 - Maintenance man-hours per operating hour
 - Mean Time Between Failure (MTBF)
 - Mean Time to Repair (MTTR)
- The maintenance concept including:
 - Justification for any support activities (events and operational requirements)
 - Preventive and corrective maintenance activities
 - Operational support activities
 - Maintenance level and maintenance location information
- Field/fleet supportability improvements and status
- Lessons learned and status
- Support drivers
- New technology requirements

In general the LSA PP can be a part of an overall ILS management plan or ILS PP. For more complex projects it is recommended to have a separate document. Many aspects to be covered are also harmonized and agreed between customer and contractor within an LSA Guidance Conference (GC).

2.4.2

General requirements

The LSA PP includes the following elements of information, but not limited to, with the range and depth of information for each element tailored to the project phases:

- A description of how the LSA program will be conducted to meet the system and logistic requirements defined in the applicable project documents
- A description of the management structure and authorities applicable to LSA. This includes the interrelationship between line, service, staff, and policy organizations.
- Identification of each LSA task that will be accomplished and how each will be performed
- A schedule with estimated start and completion points for each LSA program activity or task. Schedule relationships with other ILS program requirements and associated system engineering activities must be identified.

- A description of how LSA activities and data will interface with other ILS and system oriented tasks and data. This description will include analysis and data interfaces with the following programs, as applicable:
 - Equipment design
 - Equipment maintainability
 - Equipment reliability
 - Equipment testability
 - Facilities/infrastructure
 - Human factors integration
 - Initial provisioning
 - PHST
 - Parts control
 - Standardization
 - Support and test equipment
 - Survivability
 - System safety
 - Technical documentation
 - Test and evaluation
 - Training and training equipment
- Breakdown structure identification of items upon which LSA will be performed and documented, including software items. Identification of an LSA Candidate Item List (CIL), and LSA candidate item selection criteria. The list must include all items recommended for analysis, items not recommended and the appropriate rationale for selection or non-selection.
- Explanation of the breakdown element numbering system to be used
- The method by which supportability and supportability related design requirements are disseminated to designers and associated personnel
- The method by which supportability and supportability related design requirements are disseminated to subcontractors and the controls levied under such circumstances.
- Government data to be furnished to the contractor
- Procedures for updating and validating of LSA data to include configuration control procedures for LSA data
- LSA requirements on Government Furnished Equipment/Materiel and subcontractor/vendor furnished materiel including end items of support equipment
- The procedures to evaluate the status and control of each task and identification of the organizational unit with the authority and responsibility for executing each task
- The procedures, methods and controls for identifying and recording design problems or deficiencies affecting supportability, corrective actions required, and the status of actions taken to resolve the problems
- Description of the data collection system to be used by the contractor to document, disseminate and control LSA and related design data

2.4.3

Functional requirements identification

This activity identifies the required operations and support functions for the Product. The primary inputs are the results of the supportability and supportability-related design factors, and results of the reliability and maintainability programs and analyses. The results of this task form the basis for the support system alternatives. Through coordination and interaction with design engineering, maintainability, human engineering and technical manuals, LSA personnel will revise as applicable functional block diagrams identifying subsystem component changes and their functional interface within the subsystem and with associated subsystem components. LSA personnel will then evaluate and identify operations and maintenance functions that are required to implement the maintenance and operational concept for LSA candidates.

The functional requirements inherent in maintenance and operation of the Product include requirements such as:

- Inspections
- Servicing
- Testing
- Operating
- Repairing
- Configuring for specific usage

2.4.4 Unique functional requirements

The same process used to identify operations and maintenance functions will identify any unique functional requirements. These unique functional requirements are normally associated with new technologies and equipment incorporated in the system or its support system. Hazardous materials, hazardous wastes, and environmental pollutants will also be considered.

Although the same analysis techniques are utilized to identify and evaluate these unique functions, special attention will be given to them because they have a tendency to exhibit higher risks. The LSA organization will ensure that configuration changes are tracked and their impact on identified unique functional requirements are identified and evaluated.

2.4.5 LSA program plan - generic example

The LSA PP is a crucial output document of the LSA GC. It is organized with general sections and descriptive subsections. These are followed by the appendices. In the general section, the following aspects are covered as a recommended minimum:

- Management structure of the program for both the contractor and customer
- Time schedule of the program and meeting calendar
- Definition of milestones
- Responsibilities and reporting levels
- Change process (categories and levels of authority)
- Risk management
- Work share agreements

[Fig 2](#) shows a simplified and generic example of the structure of an LSA PP:

Project XXX		LSA Program Plan	Date
CONTENT			
1	General		
2	Scope		
3	General program definitions		
3.1	Management structure of the program on customer side		
3.2	Management structure of the program on contractor side		
3.3	Time schedule of the program		
3.4	Definition of milestones		
3.5	Meeting calendar		
3.6	Responsibilities and reporting levels		
APPENDICES			
Appendix A	Purpose of LSA and of logistic analysis tasks		
Appendix B	System breakdown methodology		
Appendix C	Breakdown depth		
Appendix D	Rules for LSA candidate selection		
Appendix E	Candidate Item List		
Appendix F	Rules for analysis tasks selection		
Appendix G	Rules for performance measuring and verification		
Appendix H	IT aspects		

ICN-B6865-S3000L0006-001-01

Fig 2 Example of content of an LSA Program Plan

The LSA PP appendices will provide additional clarity and specific needs which supports the content in the subsections.

2.4.5.1 Purpose of LSA and logistic analysis activities

As a basic result of the LSA GC, the contractor and customer must agree to the principles of how to use the data coming from the logistic analysis activities. The documentation of data within a logistic database must define the purpose of collecting the data. Therefore, it is strongly recommended to carefully select which data elements will be documented in a logistic database and then link the data with its corresponding purpose. This also applies to the logistic analysis activities. The selection must consider technical and economical aspects, especially for very extensive analysis. Examples can be the following:

- Detailed or simplified Level of Repair Analysis (LORA)
- Optimization methods, such as via simulation runs
- Detailed Scheduled Maintenance Analysis (SMA)

2.4.5.2 Hierarchical product breakdown methodology

In this appendix to the LSA PP, the rules for establishment of a product breakdown methodology must be documented. It must be clarified between the customer and the contractor, how the breakdown elements are to be identified, eg using the S1000D Standard Numbering System (SNS), Logistic Control Number (LCN) in accordance with MIL-STD 1388-2B, or any other identification system preferred by the customer.

If it is necessary to use different breakdown methodology (physical and functional) within a program in parallel, it must be clarified whether and how to interconnect these different breakdowns. Additionally, the purpose of its use must be documented (eg a functional breakdown).

Note

It is recommended that detailed examples in the appendix are included, covering all possible cases (eg it must be clarified how software will be incorporated within the hierarchical breakdown, it must be clarified how to cover different end item variants).

2.4.5.3 Breakdown depth for the item under analysis

In addition to the decisions required for the types of breakdown, the breakdown depth is a crucial attribute of the breakdown itself. The following aspects must be considered for a final decision:

- Is a full breakdown of the item required to identify all relevant spare parts and is this breakdown type effective and applicable?
- Is a breakdown that only contains LRUs applicable and effective?
- What depth of breakdown is required to identify all spare parts for a repair concept at the authorized level?
- For what purpose a functional breakdown is established (eg for SMA) and how is the functional breakdown linked to the physical breakdown?

2.4.5.4 LSA candidate selection criteria

The criteria for LSA candidate selection shall be documented in an appendix of the LSA PP. A decision flowchart can also be part of the appendix.

2.4.5.5 Candidate item list

As an input to the LSA GC, a draft of the CIL must be prepared as an input document by the contractor. During the GC, this draft must be discussed between the contractor and the customer, resulting in an agreed upon CIL. The CIL will be a formal output document of the LSA GC.

2.4.5.6 Potential analysis activities for candidate items

The CIL does not only contain the selection of the LSA candidates from the product breakdown, it can also contain the analysis activities to be performed for each LSA candidate. In addition to the analysis activities selection, the depth of the analysis must be clarified. For example, in the area of the Maintenance Task Analysis (MTA), a decision must be made about how deep the description of the maintenance task is documented or whether it is necessary for the project to identify personnel requirements. Finally, the CIL can serve as an LSA program overview, in which the complete effort is summarized. Additionally, the CIL can be used as a helpful management tool, documenting the progress of the LSA program.

Note

As a final output from the LSA GC, a CIL must be generated. It is recommended that this list is designated as a reference table and part of the projects master data which is shared and synchronized across the project's master data management system. It can be used as a monitoring tool to supervise the progress of the project. This will avoid duplicating data, such as breakdown information, in external lists which are not synchronized to cleansed master data.

2.4.5.7 Performance measurement and verification of LSA activities

The performance of LSA activities must be continuously evaluated and documented eg by status information within the logistic database. The criteria, when an analysis activity is fulfilled or when an LSA candidate is completely documented within the logistic database, must be clear and documented in a list of acceptance criteria that are a part of an appendix within the Guidance Conference Document (GCD). The process and the rules of performance measurement and verification must be clear. For each delivery of any analysis result, these rules must be obeyed by both the customer and the contractor.

2.4.5.8 IT aspects

It is recommended to avoid the usage of different software packages for the same analysis activities at different locations. Integration and harmonization processes can be extensive and are a potential risk factor. However, many projects currently use different software packages because of existing licenses, contractual constraints or IT environment necessities. In this case, for any task to be performed, the usage of suitable software packages compatible between all partners are recommended.

The decisions about software packages must be documented at the LSA GC. Changes of software releases during a program must be harmonized and agreed to between all impacted partners.

2.4.6 LSA Guidance conference document - generic example

The GCD is another crucial output document of the LSA GC. It contains the implementation rules of the LSA process on reporting, data element definitions, data exchange procedures and selection of software packages to perform analysis activities.

An example for a possible structure of this document is given in [Fig 3](#).

Project XXX	LSA Guidance Conference Document		Date
CONTENT			
1	General		
2	Scope		
3	Implementation rules		
3.1	Simple data element list (DEL)		
3.2	Reporting process		
3.3	Definition of required reports		
3.4	Data exchange processes		
APPENDICIES			
Appendix A	Detailed data element list (DEL & input instructions)		
Appendix B	List of selected software packages per analysis task		
Appendix C	Report examples		
Appendix D	Mathematical appendix		

ICN-B6865-S3000L0007-001-01

Fig 3 Example of content of a Guidance Conference Document

The GCD is organized with a general section and a number of appendices, which are described in the following paragraphs. In the general section, the definition of the implementation rules cover as a minimum:

- Data Element List (DEL):
This is a list of all the required data that must be recorded in a logistic database.
- Reporting process and definition of required reports:
All report content and format must be defined clearly. Any data elements required for the reports must be available and accessible. All contracting parties will agree on content, format and derived calculations. Any business rules necessary to calculate and/or select source data will be identified and documented.

Data exchange process:

For a proper exchange of information, a data exchange process must be established and harmonized between the customer and the contractor. Especially in programs with several partners on both sides, customer and contractor, a clear process for data integration and

harmonization is required. A time schedule must be agreed to during the LSA GC and the consequences of missing the time schedule must be clarified.

Note

A detailed description of the recommended data exchange format by the usage of ASD DEXs is given in [Chap 20](#).

2.4.6.1 Data element list and input instructions

The detailed DEL describes the context and description of the data being captured in an LSA database. This includes syntax (if required) and allowed codification. Also, any calculation methods of numeric values can be described in a mathematical appendix.

To support the initial selection of data elements, it is recommended to create a simple draft DEL as an input to the LSA GC. The justification for the project's LSA DEL selection is based on the following questions:

- Is the data element required for the proof of any specification values?
- Is the data element required for the calculation of any logistic parameters?
- Is the data element required for the following disciplines such as technical documentation, materiel support, identification of special support equipment, and identification of facilities or training requirements?
- Is the data element required for the performance proof of the LSA itself?
- Is the data element required to document the results of the logistic analysis activities?
- Is the data element required for any report, necessary for the customer and/or the contractor?
- Is the data element a special interest for the customer/contractor (eg internal usage of contractor)?

Note

It is recommended to select the data elements based on a logical, traceable project requirements need. Listing of any data elements that do not link to a requirement or the development (calculation) of a requirement are a candidate for elimination.

The DEL must be agreed to during the LSA GC and will be a part of the official GCD.

Nevertheless, it is to be considered as a living document. The DEL is updated due to the arising of additional needs during program maturation (eg matured project design as indicated by contractor and customer).

2.4.6.2 List of selected software packages

In the GC, the decision must be made or at least prepared, on which software packages will be used by the contractor and the customer to support the required analysis activities. The selection of software packages can be contractual. An upgrade or a complete change of software packages within the project must be investigated carefully, including all participants involved with such changes in the IT environment.

2.4.6.3 Reports examples

To guarantee a common understanding of reporting, examples can be part of the GCD. These examples are distributed to the conference members: formatting, sorting, filling and calculations.

2.4.6.4 Mathematical appendix

To complete the implementation rules, calculation methods of calculation can be documented. The calculation of values must be evaluated by both, contractor and customer, and must be available for review by those who have a need. For this reason, a mathematical appendix covers two basic aspects:

- Mathematical equations and their explanations including a list of mathematical abbreviations and symbols used
- Sources and business rule logic from which the data is obtained for calculation

2.5 LSA program organization

The program manager is the final authority for effecting implementation of the LSA program integrated product development organization. LSA is part of the system engineering integration. LSA is an integral element in the integrated product team's commitment of providing the highest quality products and services at the lowest possible cost.

The system engineering integration organization is responsible for defining the requirements analysis for the LSA program.

- Performing the functional analysis and allocation and providing them to the integrated product teams
- Providing product and process solutions which satisfies the supportability requirements
- Providing system analysis and control activities through the system engineering process

The integrated product team leader is responsible for completing the LSA program on his Product.

The ILS manager is responsible for monitoring the LSA activities on all teams, ensuring commonality across the program and for delivery of LSA data submittals (if required). The program manager has final authority for all program functions. Supportability is an active member of the integrated product teams. The objective of these teams is to develop guidance in system design that meets performance, producibility and supportability requirements and achieves low LCC.

2.6 LSA management responsibilities

The following list provides an overview of the potential responsibilities of the different players in the LSA process.

2.6.1 The LSA manager

The typical responsibilities of an LSA manager can be, but not limited to the following:

- Implement established company operating policies and procedures concerning LSA
- Accomplishes quality reviews of LSA data through in process reviews and formal reviews prior to data submittal
- Addresses questions and coordinate required corrective actions pertaining to the development, implementation, and modification of integrated maintenance concepts and logistics resources and related problems/status
- Assesses subsystem design and support concepts for supportability influence/impact for their assigned subsystems
- Assists in resolution of problems pursuant to acquisition of supplier/customer-furnished LSA data necessary to support the assigned subsystems
- Assists in the development and implementation of formal LSA review activities to ensure integration of maintenance concepts and compatibility with contractor and customer requirements for their respective system
- Conducts LSA reviews with the customer and incorporate the results into the LSA database
- Coordinates evaluation of design changes for LSA impacts and ensure impacts are highlighted to decision makers when design changes will have an adverse impact on ease and cost of support
- Coordinates with appropriate Program and Functional management to ensure problems are resolved for their assigned subsystems
- Documents LSA task analyses and coordinate LSA task scheduling and planning
- Maintains technical liaison with design/supportability engineering and logistics groups
- Manages the technical aspects of the LSA process and the documentation of design considerations and logistic resource identification
- Monitors/coordinates the technical aspects of LSA relative to established program objectives, schedules, and directives

- Participates in customer and supplier design and program reviews and ensures LSA is an agenda topic as appropriate
- Participates in technical coordination meetings and design reviews with suppliers and/or the customer, ensuring that each review include a formal review and assessment of supportability and related design requirements
- Provides assistance and information to Program and Functional management in achieving contractor and customer objectives
- Schedules LSA in accordance with engineering documentation release dates/support milestones, and accomplish LSA database management and configuration
- Tracks LSA task accomplishment and participate in problem resolution
- Establish company LSA operating policies and procedures
- Help establish program statement of work and approve man-hour estimates for all LSA activity
- Help assign personnel to programs and projects as required satisfying the LSA statement of work
- Monitor and assist LSA program managers/leads in the performance of program statements of work
- Establish training requirements for LSA personnel and LSA related tasks

2.6.2

Program LSA managers and leads

The typical responsibilities of an overall program LSA manager (eg required within international projects with several participating companies) can be, but not limited to the following:

- Ensure LSA program requirements are being met
- Implement established company operating policies and procedures
- Control program cost and maintain program schedule
- Maintain a functional interface with team members, subcontractors and/or vendors
- Maintain a program interface with program management and the customer
- Be technically responsible for the accuracy of the LSA

2.6.3

Program technical staff (logistics engineers and analysts)

The technical staff is responsible for the proper operation of the IT systems. This includes the evaluation and provisioning of LSA data for other ILS disciplines or for management purposes.

- Responsible for the performance of the data analysis
- Operate data collection software and generate reports
- Manage supplier and customer interfaces for data collection
- Have working relationships with engineering, all ILS elements, suppliers, and customer

2.6.4

Supportability analysis integrated product team

Participate in the development of an early LSA strategy:

- Develop the LSA PP
- Participate in program and design reviews
- Coordinate the operational requirements
- Coordinate the requirements of mission hardware, software, and support system standardization
- Participate in developing the requirements for comparative analysis
- Coordinate the requirements for technological opportunities
- Coordinate the requirements of supportability and supportability related design factors
- Perform scheduled maintenance analysis
- Participate in the development of support system alternatives
- Participate in evaluation of alternatives and trade-off analyses
- Coordinate inputs from maintainability and packaging, and develop maintenance task analysis
- Coordinate the requirements for early fielding analysis

- Coordinate the requirements for post production support analysis
- Coordinate the requirements for supportability test, evaluation, and verification

3 Product development organization

The product oriented organization integrates the unique capabilities of each engineering discipline through the product development process. This process is a systematic approach to the integrated, concurrent design of Products and their related processes, including production and support. Use of this approach is intended to cause the developer, from the outset, to consider all elements of the product life cycle from conception through disposal, including cost, schedule, performance, supportability, quality and user requirements.

Integrating product development during design will help achieve first time quality and improved compliance with requirements will:

- Support completing the program within target cost
- Meet affordability requirements through design for producability and concurrent manufacturing process improvements
- Reduce operational and support costs through design-to-supportability with emphasis on reliability, maintainability and life cycle cost

Fundamental characteristics of an effective integrated product development organization are:

- Integrated product teams
- Clear product definition and interfaces
- Multidiscipline teams (integrated teams)
- Disciplined processes
- Single point accountability
- Effective communications
- Empowerment
- Performance metrics

3.1 Integrated product teams

Each of the product items in the entire system, including supplier and customer furnished items, is owned by a product team. Resource allocation and integration normally occur through a WBS hierarchy of product teams starting with the overall system team. The flow down is from team to team to the detailed subassemblies or components. These teams represent significant, identifiable subsystems, sub-subsystems and components, operation planning and logistics segments.

Each level in the organization is fully accountable for the aggregate of its subordinate teams, as well as for the integration and overall performance of its end product. Team leaders at all levels embody the full responsibility, authority and accountability of their teams. A single unbroken line of authority flows through the team leaders from the program manager to the subassembly or component team leaders. Team leaders allocate roles and responsibilities based on specific, tangible Products. Each team member has a clear charter that focuses on the development or creation of a Product.

3.2 Multidiscipline teams

Early consideration of the product life cycle results in the identification of design conflicts. This feature is made possible with multidiscipline teams where there is diversity of perspective to a common mission. Each team member, whether they are skilled in:

- Design
- Quality assurance
- Manufacturing
- Reliability, testability and maintainability

- Technical data and technical documentation
- Support equipment
- Supply support
- Facilities
- Technical services
- PHST
- Personnel and training

is forged into a single effort. The objective of an integrated team is to identify and resolve as much design conflict as possible in the shortest period of time with the understanding that each perspective is of equal value. Each team member brings a unique perspective to the creation of a single, concurrently engineered design. Without this integration of perspectives there is no integrated product development or concurrently engineered design.

3.3 Single point accountability

The integrated product team leader is held accountable for the development of all Products. This feature provides the benefits of avoiding redundant efforts and defining clarity of purpose.

The integrated product team leader brings together the right resources (manpower, skills, budget and facilities) to achieve the goal.

3.4 Empowerment

Empowerment enables integrated product team members to achieve their mission and provides them with the ability to influence and to the maximum extent possible, control their disciplines destiny. Team members are empowered to the extent necessary to fulfill their individual roles and responsibilities but their ultimate destiny is controlled by the performance of their teammates. By definition, every member within a team is dependent upon the other team members. Empowerment is the principle of enabling team members, with sufficient and adequate resources, to achieve their specified roles and responsibilities.

3.5 Management plans

Corrective actions for resource problems consist of procurement actions (early orders for long-lead-time items or schedule, cost, and budget changes), support concept changes or design changes. These resource challenges will be brought to management's attention through technical analysis reviews and schedules. Problems that are identified will be worked through the responsible functional management. The responsible subsystem logistics manager will ensure that appropriate action is assigned and that proposed changes are implemented.

3.6 Clear product definition and interfaces

Mission success is dependent upon understanding clearly and completely what the Product is that the team is accountable for providing. The product description includes not only the requirements like performance, supportability, and producibility but also how that Product must relate or interface with other Products.

3.7 Disciplined processes

The processes for designing, developing, testing, producing and supporting system and support system Products are founded upon accepted standardized principles. The focus of integrated product development is rigorous and consistent application of standardized processes for the development of all Products along with a continuous search for methods to improve the process. As the processes are documented, the team is committed to management by process rather than management of the Product.

3.8 Effective communications

Communications in the integrated product development process simply means that a free and open exchange of data, information and opinions is the fuel that keeps the process working. Effective communications begins with establishing the roles and responsibilities of the teams,



defining the team's products, establishing team goals and objectives, and providing the ability to identify and resolve design conflict. Instrumental to effective communications are collocation of team members, regularly scheduled integrated product team meetings, all-hands meetings, functional and program staff meetings, effective use of memos, use of electronic communications such as E-mail, video conferences, and knowledgeable interchange of data and information.

3.9

Performance metrics

It is important to know how well each integrated product team is progressing toward development of their Products. Measuring is achieved by identifying those critical parameters in the product development process and establishing an associated metric. The parameters selected by an individual or team must characterize:

- Product quality
- Schedule
- Cost
- Risk assessment

Chapter 3

LSA business process

Table of contents

	Page
LSA business process	1
References.....	4
1 General.....	5
1.1 Introduction.....	5
1.2 Objective.....	5
1.3 Scope.....	5
2 Establishing Product usage data.....	6
2.1 General usage aspects.....	6
2.2 Operational requirements document	7
2.2.1 General usage scenario	7
2.2.2 Geographical position of locations and special conditions of each location	8
2.2.3 Product support and product deployment	8
2.2.4 Usage overview	8
2.3 Customer requirements document	8
2.3.1 Supply concept	9
2.3.2 Support equipment concept.....	9
2.3.3 Personnel integration and staff training.....	9
2.3.4 Facilities.....	9
2.3.5 IT and communication resources	9
2.3.6 New organizational structures	9
2.3.7 Schedule considerations	10
2.3.8 Additional aspects	10
2.3.9 Detailed checklist for the creation of a customer requirements document	10
2.4 Time schedule for document creation	10
2.5 Site surveys	10
2.6 Qualification requirements.....	10
2.7 Certification requirements.....	11
3 Establishment of Product design and performance data	11
3.1 Selection criteria concerning LSA relevant data and information.....	11
3.1.1 Selection of LSA data and information derived from contractual documents	11
3.1.2 Selection of LSA data and information derived from Product usage data	12
3.1.3 Selection of LSA data and information derived from design and performance specifications	12
3.1.4 Selection of LSA data and information relevant for Product certification and verification contained in other Product documents.....	12
3.2 Influence of overall LSA strategies and principles on LSA data selection	13
3.2.1 Procedures and principles influencing selection of LSA data	13
3.3 Acceptance rules concerning the verification of values	14
3.3.1 Category of measurement values.....	14
3.3.2 Identifying tolerances.....	14
3.3.3 Establishment of acceptance criteria.....	14
3.3.4 Establishment of special rules	14
3.4 Criteria and procedural aspects for verification of projected values.....	14
3.4.1 Determination of measurable target values.....	14
3.4.2 Verification of the individual projected values	15
3.4.3 Verification method	15
3.4.4 Verification for special purposes.....	15
3.4.5 Update of status codes	15

3.5	Customer involvement.....	15
3.5.1	Checklist for the LSA guidance conference	15
3.5.2	Documentation of the results of the LSA GC	16
4	LSA guidance conference	16
4.1	The transition from a request for proposal to the LSA guidance conference.....	16
4.2	LSA business process overview.....	18
4.3	Documents and Information for LSA guidance conference.....	19
4.3.1	Checklist of documents for the LSA guidance conference.....	20
4.3.2	Checklist of documents and information from the LSA guidance conference.....	21
4.4	Candidate item list.....	21
4.5	Contractual documents.....	21
5	Candidate item selection and identification	22
5.1	Definitions and general terms.....	22
5.1.1	Candidate and non-candidate	22
5.1.2	Maintenance relevant items and maintenance significant items	23
5.1.3	Structural items, structure significant items and structural details	23
5.1.4	Non-hardware items	23
5.2	Classification of LSA candidates	24
5.3	Selection process and criteria list.....	24
5.3.1	Preconditions for the selection process - Product breakdown	24
5.3.2	Preconditions for the selection process - Existing analysis results	25
5.3.3	Preconditions for the selection process - Candidate selection rules	25
5.3.4	Preconditions for the selection process - Candidate classification selection rules	26
5.4	Influencing factors	27
5.5	Recommendations for LSA candidate selection.....	27
5.6	Candidate item list (draft)	29
5.7	Candidate item list as an output of LSA guidance conference.....	29
6	Logistic analysis activities.....	30
6.1	Principles for analysis activity selection	30
6.2	Potential analysis activities	30
6.2.1	Analysis for identification of general LSA needs	31
6.2.2	Comparative analysis	31
6.2.3	Human factor analysis	31
6.2.4	Configuration assessment.....	32
6.2.5	Assessment of reliability analysis	32
6.2.6	Maintainability analysis.....	32
6.2.7	Testability analysis	33
6.2.8	Analysis activities concerning event driven maintenance	34
6.2.9	Level of repair analysis	35
6.2.10	Maintenance task analysis	36
6.2.11	Software support analysis	36
6.2.12	Operations analysis	37
6.2.13	Simulation of operational scenarios	37
6.2.14	Training needs analysis.....	37
6.3	Analysis relations and general overview	37
6.4	Analysis activities selection criteria	38
6.5	Checklist for analysis activity recommendation.....	40
6.5.1	Recommendation and decision sheet	40
6.5.2	Selection matrix example and rating values.....	40
6.6	Analysis workflow processes.....	41
6.6.1	Early phase analysis activities	42
6.6.2	Design freeze analysis activities.....	43
6.6.3	In-service phase analysis activities	43
6.6.4	Summary of activities over the life cycle phases.....	43
7	Customer involvement.....	44
7.1	Customer assessment of candidate items and recommended analysis activities	44

7.1.1	Establishing LSA assessment rules	45
7.1.2	Initial assessment and re-assessments	45
7.1.3	Establishing an assessment procedure.....	46
7.2	Information exchange between customer and contractor	46
7.2.1	Commenting process.....	46
7.2.2	Informing the contractor by the customer	46
7.3	Final clarification on open issues.....	48
7.4	Customer decision (status influencing)	48
7.5	Exchange of analysis data with the customer	49
7.5.1	Establishment of appropriate rules for data and document exchange	49
7.5.2	LSA data and document collection by the contractor	49
7.5.3	Delivery of LSA data/documents by the contractor	49
7.5.4	Customer assessment and distribution of assessment results	49
7.5.5	Distribution of customer response by the contractor	49
7.5.6	Distribution of the contractor response concerning disagreed comments	50
8	LSA review conference.....	50
8.1	LSA review process overview.....	50
8.2	Subject for review	51
8.3	Examples for review structuring	51
8.3.1	LSA review step - CIL and maintenance analysis allocation.....	51
8.3.2	LSA review step - LORA results and maintenance concept information.....	51
8.3.3	LSA review step - FMEA and SMA results	52
8.3.4	LSA review step - Maintenance Task Analysis (MTA)	52
8.4	Status code examples	52
8.5	Depth of status code allocation	53
8.6	Functions of the status code.....	53
8.7	Status code definition at the LSA guidance conference.....	53
9	Starting point and management of the creation of the logistic products	53
9.1	Starting point recommendations.....	53
9.1.1	Technical documentation.....	53
9.1.2	Materiel support.....	55
9.1.3	Common and special support/test equipment.....	56
9.1.4	Training.....	57
9.2	Management of the development of the logistic products	58
9.3	Influence of modifications on logistic products	58
10	Checklists	59
10.1	Detailed checklist for the creation of an ORD	59
10.2	Detailed checklist for the creation of an CRD.....	63
10.3	LSA candidate selection flowchart examples	66
10.3.1	Non structural items.....	66
10.3.2	Structural items.....	67

List of tables

1	References	4
2	Required input documents for LSA GC	20
3	List of output documents of the LSA GC	21
4	Definitions of MSI and MRI.....	23
5	Definitions of Structural Item, Structure Significant Item and Structural Detail	23
6	LSA candidate categories.....	24
7	Classification criteria for full LSA candidates	26
8	Classification criteria for partial LSA candidates	26
9	Classification criteria for LSA candidate family	27

10	Recommendations for LSA candidate selection.....	28
11	Depth of maintainability analysis depending of item type	33
12	Classification of different depths of LORA.....	36
13	Analysis activities selection criteria	39
14	Summary of LSA activities during the Life Cycle Phases	44
15	Explanation of time schedule of the commenting process	47
16	Examples for status codes	52
17	List of different spare part types identified by LSA	55
18	Checklist of detailed questions to support the ORD creation.....	59
19	Checklist of detailed questions to support the CRD creation	63

List of figures

1	Schedule for the creation of the basic documents	10
2	LSA activities in conjunction with the preparation of an offer/contract.....	17
3	Flowchart of LSA business process at a glance (Sheet 1 of 2).....	18
4	Flowchart of LSA business process at a glance (Sheet 2 of 2).....	19
5	LSA GC, inputs and outputs	20
6	Analysis activities relations and overview	38
7	Example of a simple analysis activity recommendation sheet	41
8	Position of LSA in the overall project schedule	42
9	Process of information exchange between customer and contractor (example)	47
10	Correlation between LSA and technical documentation	54
11	Correlation between LSA and materiel support.....	56
12	Correlation between LSA and support/test equipment.....	57
13	Correlation between LSA and training.....	58
14	Influence of design modifications on logistic disciplines in general.....	59
15	Full LSA candidate selection flowchart.....	66
16	Partial LSA candidate selection flowchart	67
17	LSA candidate selection flowchart for structural items.....	68

References

Table 1 References

Chap No./Document No.	Title
<u>Chap 5</u>	Influence on design
<u>Chap 9</u>	Logistic related operations analysis
<u>Chap 10</u>	Scheduled maintenance analysis
<u>Chap 12</u>	Maintenance task analysis
<u>Chap 13</u>	Software support analysis
DEF-STAN-00-60	Integrated Logistic Support - UK MoD
MIL-STD 1388-2B	DoD requirements for a Logistic Support Analysis Record
S1000D	International specification for technical publications using

S4000P	a common source database
SAE ARP5580	International specification for developing and continuously improving preventive maintenance
	Recommended Failure Modes and Effects Analysis (FMEA) Practices for Non-Automobile Applications

1 General

1.1 Introduction

With the introduction of a new Product, all logistic requirements must be made available in a timely manner. This requires projects to establish a process that ensures logistic requirements are taken into account during the design of the Product. This process includes a number of analysis activities concerning a wide range of logistic considerations. The contractor and the customer must both agree on the activities to be carried out in order to achieve proper supportability. Early consideration of logistic aspects is increasingly important with regard to both operational and economic considerations. A Product that cannot be operated and maintained properly and cost effectively is not acceptable to the end user.

Modern Products normally contain software elements. The overall Logistic Support Analysis (LSA) process for software and hardware is very similar. Therefore, this chapter is valid for the supportability of software, as well as hardware. However, some specific aspects of software must not be ignored (refer to [Chap 13](#)). This chapter is referenced as appropriate.

Logistic considerations are significant in terms of costs. Over the life cycle of a complex technical Product, support costs are much higher than acquisition costs. Because of this, projects are tending to consider logistic aspects as important as performance aspects.

1.2 Objective

This chapter is a guideline for the establishment of an effective LSA business process. It considers each life cycle phase of a Product and emphasizes the importance of the logistic requirements. Additionally, the interaction between the contractor and the customer during the different phases of the Product's life cycle is described in detail. The LSA process has two main purposes:

- 1) influence the design to allow appropriate supportability with the help of different analysis methods, and
- 2) the creation of logistic end products should be managed.

The requirements for spare parts, consumables, technical documentation, support equipment, personnel, training, facilities and software support are identified and the creation of the logistic end products should be supported. Altogether, this is an extensive task, which must be carried out with accuracy. A properly established LSA process that is embraced and used by the customer and the contractor is an example of an efficient plan for a successful introduction of new complex technical equipment.

1.3 Scope

This chapter is directed at Integrated Logistic Support (ILS) and LSA managers for both the customer and the contractor. LSA offers a powerful methodology to build the core needs for the realization of ILS. Additionally, monitoring and control functions can be achieved by using relevant LSA information. The quality and commonality of the logistic products can be positively influenced by the application of an LSA process. With the incorporation of various analysis results, it can be assured that customer's needs for operability, supportability and readiness can be achieved.

2

Establishing Product usage data

To identify the pertinent supportability aspects of a new Product, all relevant information related to the intended use must be collected and documented in a set of consistently structured, mandatory documents. In an early stage of a project, sometimes complete relevant information is not always available. In this case, iterative steps would be necessary for a complete definition of the customer's usage of the Product to be analyzed. In any case, changes to the usage scenario are of crucial importance for logistic purposes and must be taken into consideration by the logistic analysts accordingly.

2.1

General usage aspects

To establish an initial overview that identifies pertinent supportability aspects, some very general decisions are necessary. These first decisions must consider the overall preconditions for the usage of the new Product to be introduced and also consider some strategic aspects coming from the Product design and performance data, which are described in [Para 3](#). These general decisions should be documented in a general project document describing the overall support strategy. This strategy should be a basic guideline for the further execution of any logistic analysis and for creation of the Operational Requirements Document (ORD) and the Customer Requirements Document (CRD).

The general questions, which should be answered before, or at least in parallel to, the creation of the ORD and the CRD, are the following:

- How will the Product be used?
A short description of the Product should be given including key features, key requirements and basic technical data needs.
- Will contractor logistics support be required? If yes, at which maintenance levels?
This is a very central question that can only be answered by a deeper examination of some additional aspects described in the following questions.
- Where will the Product be operated and maintained?
 - In which operational environment will the Product be operated and maintained (from a fixed/industrial/benign environment to a mobile/austere/hostile one)?
 - Will the environment influence the item's characteristics (eg, reliability, maintainability)?
 - Will the environment significantly change the manner in which the item must be repaired? If so, there could be a better approach than contractor support.
- How long will the Product be used (predicted service life)?
If the Product will only be on the inventory for a few years, then contractor support could be preferable to a lengthy and costly gearing-up of an organic logistics support structure.
- How many maintenance levels are planned for the Product support?
The repair strategy must be clearly established before the ORD is written.
- What are the features and/or typical actions within each maintenance level?
For example in a classical 2-level maintenance concept, the exchange of "black boxes" and the returning to the contractor or to the original manufacturer will be typical.
- Are there existing maintenance capabilities that can be adapted to the support concept for the new Product?
- Is the usage of existing maintenance capabilities from other Products or other similar Products at or nearby the operating locations possible and effective?
- Should the customer be involved in any repair activity of the Product or should the corrective maintenance be limited to simple "black box" exchange (replace only)? The involvement of the customer is dependent on the abilities which can be provided by the customer itself.

- What frequency and duration of preventive or scheduled maintenance actions (expressed in measurable terms) is acceptable?
- Are there limitations for preventive maintenance (eg, because of special preconditions concerning available personnel or facilities)?
- How much of the software is mature? How much is unique to the customer?
Software that is not delivered 100% bug-free, could take several years to mature. The logistics support structure should also address software maintenance of potential user required upgrades.
- Is there any software/data loading and/or unloading which should be performed by the customer?
Which concept should be established to guarantee proper function of the Product after the loading of software/data at an acceptable level (eg, simulated integrity test based on a hardware-software compliance matrix in a test bench, General Purpose Test Equipment concept for software, required software device and encrypting system for loading and/or unloading).
- What is the expected need for Product replacement or upgrade due to changes in technology? This questions concern how a support structure can keep up with changes in the Product and modify the support strategy. If it is difficult or impossible, contractor logistics support should be preferred.

All general questions should be answered as completely as possible. Additional questions to those listed above could arise. This depends on the project and on the corresponding Product to be used. The basic principle should be to use the best information available. It is recommended that a basic document with all general and programmatic information be created and agreed to by both the contractor and the customer. Ideally this should be done before the creating the ORD and the CRD.

2.2

Operational requirements document

Operational requirements must be defined in a quantitative and qualitative way. Previously conducted analyses concerning area of operation and Product usage, that identified relations between hardware, usage and supportability, should be considered. A further definition of the identified operational requirement must be performed and metrics gathered. Supportability requirements are developed directly from the operational requirements. Each supportability requirement should be based on an operational requirement, and that relationship should be clearly identified. If the basis for the supportability requirement is not clear, that requirement should be regarded with suspicion. The main goal is to identify and document the pertinent operational requirements related to the intended use of the new Product. The writers of the ORD are required to identify key performance indicators. Any parameter not identified as a key parameter, should be a candidate for revision when supportability attributes are negatively impacted. The ORD requires the Product developer to make difficult choices between "must have" and "nice to have" at an early stage of the project. This information is of vital importance to the logisticians so they can understand what they must support, regardless of costs, and what they can trade off.

2.2.1

General usage scenario

In the general usage scenario, all information that describes the environmental aspects of the usage of the Product should be collected. These include but are not limited to:

- General description of overall usage areas and/or mission areas
- Possible operation scenarios and requirements for each scenario must be documented
- Influence on the environment by the usage of the Product and resulting actions to avoid or reduce negative influence
- Supportability problems that have arisen over the life of the currently used Product
- Interaction and dependability with existing Products

- Mobility requirements

2.2.2 Geographical position of locations and special conditions of each location

Details that describe each location in which the Product will be operated and any special aspects of those locations should be collected. These include but are not limited to:

- Number and geographical position of the operating locations
- Type of each operating location
- Special conditions at each operating location
- Is the location in a peacetime or wartime region?
Threat situations require a special emergency maintenance concept limited to minimum intervention.
- Is sufficient infrastructure available to access each operating location?
- Special infrastructural requirements for reaching a location
- Capabilities (existing and planned) of each location (eg, equipment, infrastructure, personnel, facilities, supply depot, repair station)
- Interactions between different locations concerning, as an example maintenance support, from one location for another location

2.2.3 Product support and product deployment

Details of the deployment of the Product and the interactions coming from the deployment should be collected. The location at which the Product will be based, affects the decision to use either organic or contractor support. These include but are not limited to:

- Number of systems supported per location
- Deployment of Products per location
- Interactions between the different locations concerning usage

2.2.4 Usage overview

Depending on the customer, define the planned usage of the Product as a basic input to logistic analysis. The frequency and the duration of the usage, in combination with the reliability of the Product, provide the initial basis for determining the range and quantity of support resources that will be required. These include but are not limited to:

- Key usage/key missions per location
- Performance parameters and constraints. The Product performance parameters, such as range, accuracy, payload or speed, are to be identified in measurable quantifiable terms. General terms or those whose interpretation is potentially ambiguous must be avoided.
- Predicted operational availability and usage success rates
- Operation per unit of time
- Operation profile per operating day, week, month or year
- Usage of the Product as training equipment as a portion of operating time
- Permanent operational conditions/possible maintenance windows
- Average duration of each unique usage event
- Key measurement base of usage per unit of time

For a detailed checklist on how to create an ORD, please refer to [Para 10.1](#).

2.3 Customer requirements document

To ensure the proper usage of the Product, the customer should consider all the logistic requirements. The logistician can most efficiently influence the supportability and design of a new system during its early stages of development. One method to convey these needs is to document them in a CRD. The scope of the content of the CRD is given in [Para 2.3.1](#) thru [Para 2.3.9](#).

2.3.1 Supply concept

The supply concept is of crucial importance for the logisticians. A general decision on how to organize the provision of spare parts and consumables can have high impact on the maintenance scenario itself. The logistician must decide whether or not facilities must be planned because of the need for supply storage areas and who will manage the supply chain when outsourcing the Supply Chain Management. Costs (especially facility construction) and obsolescence are important considerations.

2.3.2 Support equipment concept

Common support equipment, instead of special support equipment, should be acquired when possible to be cost effective. To reduce costs, it is important to evaluate whether or not existing support equipment can be used or can be adapted for use.

2.3.3 Personnel integration and staff training

Manpower issues are crucial to the supportability of many systems. It is necessary to plan for the timely training of all support personnel. Acceptable risk levels, training level needs, and manpower ratios must be addressed as supportability concerns. The training needs consist of two distinctive components: initial and on-going. Both are important in order to ensure adequate personnel skill levels. Given the high level of turnover in personnel, maintaining a specific skill is often a crucial issue. Repair and maintenance personnel can also turn over rapidly. Support planning must deal with these issues.

2.3.4 Facilities

Early planning considerations must be given to facilities because of the long lead times associated with site acquisition and allocation. Facilities that address maintenance actions must be planned with care to ensure process needs and activity work flow are supported.

2.3.5 IT and communication resources

IT and communication resources is another area where logisticians need proper preparation. These include but are not limited to:

- What constraints are necessary in order to provide interfaces with other services?
- What is the trade-off when X architecture provides a desirable improvement in operational availability but denies access to Y communications network used by another service?
- Which IT architecture must exist or must be additionally established?

IT resources must include all aspects such as computer hardware, computer network components, network wiring, communication protocols, software packages, data security aspects and standards.

This subject also requires an understanding of future capabilities. Designing a system to interface with those "forecasted to exist at the time the system will be fielded" requires the engineer and logistician to be aware of the status of other related projects. How the Product interfaces with planned future communications architecture must be addressed. The logistician must assess the impact of IT-system changes to be expected and determine necessary adjustments to the logistics structure.

2.3.6 New organizational structures

New organizational structure considerations have two aspects:

- Any changes to the established organizational structure at a location of the customer that must be made to support and operate the system
- Changes in the organizational structure (eg, reduction in personnel) that can be made because the system replaces old existing systems or because the new Product is easier to maintain

Organizational structure changes have impacts on available logistics support infrastructure that must be accounted for in the development of a new Product.

2.3.7 Schedule considerations

The logistician is obviously concerned with scheduling decisions. Logistic support is a vital and integral part of any Product that is fielded. Only when logistics is an afterthought would it cause delays and inefficient use of resources. If logistic considerations are interwoven with each project in each of its phases, then the supportability schedule will have been efficiently synchronized and integrated with other system schedules.

2.3.8 Additional aspects

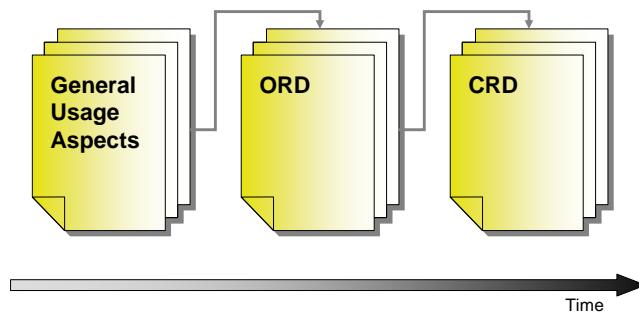
Special aspects concerning areas such as packaging, storage, handling or transportation considerations can be addressed here. Unique data requirements are defined here. Logisticians must know how and when they will use the data they request, and they must be able to distinguish between "nice to have to cover possible contingencies" data and essential data. Packaging, handling, transportation, disposal and environmental impact considerations are far from the forefront for system designers, developers, and users, but they are important and potentially expensive considerations. Logisticians must understand the potential impact of these issues on the Product from its inception, and must raise these issues whenever they have an impact on project planning.

2.3.9 Detailed checklist for the creation of a customer requirements document

For a detailed checklist how to create a CRD, please refer to [Para 10.2](#).

2.4 Time schedule for document creation

To ensure that all required information is available for each of the three basic documents which identify Product usage data, a sequence or work flow for the creation of these documents should be identified. Refer to [Fig 1](#).



ICN-B6865-S3000L0002-001-01

Fig 1 Schedule for the creation of the basic documents

2.5 Site surveys

Site surveys of operational units and maintenance/repair workshops can provide a significant input into the operational and customer requirements in terms of identifying existing capabilities, resources, and potential problems. Site surveys can be useful once the operational environment for the new equipment is identified in sufficient detail to determine existing operational units and repair facilities that would most likely be involved in the operations and support of the new equipment. The results of site surveys should be documented carefully and should become part of the ORD (eg, as an additional appendix).

2.6 Qualification requirements

When the customer requires a qualification process, it must be clear between customer and the contractor, at an early stage, which aspects of maintenance activities must be taken into account to achieve a qualification of the Product by customer authorities. The requirements for

the qualification must be defined by the customer and should be transmitted to the contractor together with the ORD.

2.7

Certification requirements

When a required certification process is necessary for the operation of the Product in the predicted environment, special attention should be given to the collection and documentation of the maintenance activities that are a part of the certification. These activities are necessary, regardless of the costs. All aspects of maintenance activities that influence certification must have first priority. The project, in its early stages, must clarify which efforts must be considered in order to fulfill the requirements of the certifying authority. The requirements for the certification are defined by the certifying authority and must be well known to both the customer and the contractor.

The customer and the contractor must ensure that the differences between the qualification process and the certification process are evident and well known to each responsible person.

3

Establishment of Product design and performance data

LSA requires the identification and documentation of relevant Product design and performance data/information. This includes but is not limited to:

- Selection criteria concerning LSA relevant data and information. Refer to [Para 3.1](#).
- Influence of overall LSA strategies and principles on LSA data selection. Refer to [Para 3.2](#).
- Acceptance rules concerning the verification of values. Refer to [Para 3.3](#).
- Criteria and procedural aspects for verification of projected values. Refer to [Para 3.4](#).
- Customer involvement. Refer to [Para 3.5](#).

3.1

Selection criteria concerning LSA relevant data and information

In this context, the data and information that should be considered as relevant are those that are subject to verification and control within the intended LSA process. Related criteria must be established in detail depending on the individual Product that is subjected to LSA, the related contract and specifications, as well as the established ILS philosophy. Relevant information can be documented within the LSA database as requirements in order to set a goal to be verified within the established LSA process. As a general approach, the selection criteria described in the following paragraphs should be considered.

3.1.1

Selection of LSA data and information derived from contractual documents

Contractual documents should be considered carefully for LSA relevant Product design and performance features that could be verified by information documented within the LSA database. Usually, overall Product requirements or so-called Key Performance Indicators of importance are covered within these documents to establish mandatory goals and/or thresholds that are potentially related to contract incentives.

Examples of Key Performance Indicators:

- Specified Maximum Maintenance Man Hours per Operating Hour.
This value could serve as a benchmark concerning successful maintenance design.
- Specified Maximum Mean Time to Repair paired with Specified Maximum Mean Time to Repair Percentile. These values could serve as an indication for successful design as it is related to repair within established time constraints.
- Specified Maximum Failures per Operating Hour. This value is an indication of the desired reliability of the Product and inversely the maintenance workload.
- Specified figures concerning minimum availabilities.
These values could serve as an indication for successful design with respect to readiness for operation.

- Testability characteristics.
These values indicate the ability of the design for monitoring vital functions, detection and localization of potential malfunctions by internal means like Built in Test Equipment (BITE) and overall test architecture.
- Minimum operational lifetime.
This value indicates a minimum lifetime requirement. This should indicate any risk of falling below the established threshold.

3.1.2 Selection of LSA data and information derived from Product usage data

Relevant information can be documented within the LSA database as a baseline reference, in this context, refer to [Para 2](#).

Examples:

- Annual Operating Requirements (AOR) with its measurement base
- Number of operating locations to be considered
- Number of systems to be operated at each location
- Maintenance levels to be established at each location
- Maintenance personnel available at each location (number of persons by trade and skill)

3.1.3 Selection of LSA data and information derived from design and performance specifications

LSA relevant parameters influencing the design and performance of the Product can be documented within the LSA database for verification and/or control purposes, in this context, refer to [Chap 5](#).

Examples:

- Specified Mean Time Between Failures (MTBF) together with the related measurement base indicating the minimum value
- Reliability growth
- Specified testability features concerning BITE
- Specified maximum allowable time for replacement
- Specified Maximum Mean Time to Repair paired with Specified Maximum Mean Time to Repair Percentile. These values could serve as an indication for successful design related to repair within established time constraints.

3.1.4 Selection of LSA data and information relevant for Product certification and verification contained in other Product documents

LSA relevant information can also be derived from other documents originated by the design department, Configuration Management (CM), safety, stress department or from the customer's documents.

Examples:

- Special operation and/or repair limitations (eg, temperature ranges, anti-static protection requirements, clean air repair conditions)
- Delivery plans
- Validity information (eg, version applicability)
- Hazardous classification of failures
- Storage limitations and/or requirements
- Criticality classification of specific parts
- Scheduled requirements (eg, established time limits, overhaul requirements)

3.2

Influence of overall LSA strategies and principles on LSA data selection

Prior to the determination of relevant Product design and performance data, an overall LSA strategy should be established as part of the tailoring of the LSA program. The tailoring processes are dependent on the project's complexity, value to business units (contractor), and known risk factors. These constraints and performance requirements are then balanced between the required analysis efforts, time, schedule, and allocated budget with a cost/benefit that is best for the project. These requirements, issues, and constraints are then reviewed and evaluated at the Guidance Conference (GC) to achieve a consensus between all participants. These findings, analyses and resulting agreements are documented in the Guidance Conference Document (GCD).

This tailoring activity is an iterative process and should be repeated in each program phase. Results and lessons learned from each prior phase should be a part of and included in the tailoring analysis for each of the following phases.

3.2.1

Procedures and principles influencing selection of LSA data

The overall LSA strategies and principles could influence the selection of Product design and performance data in relation to the LSA process. Examples for supportability needs to be considered in this area are:

- Pre-determined two level maintenance concept

Maintenance is mainly limited to only two maintenance levels, one for the replacement of an item and the other for its repair. This will help avoid requiring duplicate inventory at many repair facilities. The two level maintenance concept is normally preferred because of overall cost considerations if the equipment (LRU) to be removed and replaced at the operational site have a low failure rate and a high built-in-test detectability rate, and a responsive supply chain between the operational sites and suppliers is configured.

With this concept, LSA data are limited to pre-determined Maintenance Levels (ML).

- Repair concentrated on one certain ML's

Repair is mainly limited to user sites in order to achieve maximum autonomy, independent of cost-effectiveness.

With this concept, repair option data are limited to the pre-determined ML.

- Limited Repair at ML 1 and/or 2

The corrective maintenance action could be limited to an item exchange (no in situ repair) in order to shorten any operational downtime.

With this concept, maintenance task are limited to pre-determined criteria.

- Single source principle

When major repair is necessary, the supplier of the item should be most-favored since because of, for example, related experience, personnel and equipment are available and ready.

With this concept, repair data are limited to the supplier information.

- Interim support concept

In order to acquire experience and reduce risks prior to final Maintenance Concept (MC) decisions, an interim support phase could be established.

With this concept, MC data are limited to preliminary information.

- Commercial off-the-shelf (COTS) concept

When equipment already available on the market is used, related conditions must be accepted.

With this concept, data must reflect related supplier information/conditions.

3.3 Acceptance rules concerning the verification of values

Clearly defined acceptance rules should be established in order to avoid uncertainty during the verification process. These rules will allow accurate acceptance or rejection of the results for LSA relevant Product design and performance data.

3.3.1 Category of measurement values

The type of requirement should be stated regarding the importance of the related values such as:

- Mandatory values
- Objectives
- Thresholds (minimum or maximum values)

3.3.2 Identifying tolerances

For each identified value, the associated tolerances should be expressed.

- Tolerances for the dedicated value expressed by the requirement for one single value
- Tolerances concerning a group of similar values, for example, possible compensating of failure rates within a given area of items under analysis in order to meet the failure rate of the group instead of the single items, (eg, by following additional constraints)

3.3.3 Establishment of acceptance criteria

Measurable acceptance rules should be established for each specified value along with its related requirement. These rules should identify the basic conditions (eg, fulfillment of minimum required values based on sufficient statistic confidence levels) that lead to the acceptance or rejection of the values documented within the LSA database.

In addition, the consequences of the established status information should be stated clearly:

- Status of the Item Under Analysis (IUA) indicating the acceptance of the documented value
- Status of the IUA indicating the rejection of the documented value with related justification
- Status of the IUA indicating the conditional acceptance of the documented figure (eg, in general acceptable but requiring minor rework)

3.3.4 Establishment of special rules

When special rules are established, the related conditions and related values must be well defined in order to avoid uncertainty:

- Establishment of penalty regulations concerning failed specified LSA significant data
- Establishment of rewards in case of exceeding specified LSA significant data

3.4 Criteria and procedural aspects for verification of projected values

Finally, the criteria for verification of data and/or other information being subject to verification should be established.

3.4.1 Determination of measurable target values

For data elements under consideration the related projected value range should be established and documented within the LSA database as a requirement by considering the original target along with its established tolerances and relevant acceptance rules, if any. The projected value range can be allocated to a single item and/or a group of dependent items covered by relevant acceptance rules in order to establish the related acceptable values that fulfill the requirements.

3.4.2 Verification of the individual projected values

The relevant data and other information documented in the LSA database derived from the analysis process should be compared to the associated acceptable values. The result should be reported to the analyst and/or involved management indicating whether or not the actual LSA values consistent with the projected values.

3.4.3 Verification method

Projected values and their method of verification should be agreed upon:

- Verification by furnishing analytical proof (documented within the LSA database)
- Verification using an analytical method based on appropriate tests (eg, at a test rig)
- Verification by demonstration on a prototype or serial versions of the Product
- Verification under the rules of certification and/or demonstration programs
- Verification by trial sessions (eg, performed by customer's staff)
- Verification during long term exercises (eg, during a defined "maturity phase")

3.4.4 Verification for special purposes

When verification for special purposes are necessary in order to gain certifications, qualifications or licenses, particular rules can be established and required.

3.4.5 Update of status codes

Depending on the verification results, the related status information should be linked to the verification result by following the rules established for status allocation.

3.5 Customer involvement

The selection of Product design and performance data subject to verification and control within the intended LSA process should be agreed upon by the customer during the LSA GC and documented in the GCD. This also applies to the agreement of related rules and projected values.

3.5.1 Checklist for the LSA guidance conference

The checklist should contain all relevant proposals for design and supportability requirements. For example:

Have LSA relevant Product design and performance data been identified?

- List of selected LSA data and information derived from contractual documents
- List of selected LSA data and information derived from Product usage data
- List of selected LSA data and information derived from design and performance specifications
- List of selected LSA data and information contained in other Product documents
- Have selected LSA data and information subject to certification and verification or other special requirement (eg, for license purposes) been noted accordingly?

Have acceptance rules concerning the verification of relevant values been identified?

- Have measurable target values been identified for the selected data/information?
- Have the category of measurement figures been stated against the selected data/information (eg, mandatory values, goal values, thresholds)?
- Have tolerances been defined for the selected data/information?
- If applicable, have compensating rules been established?
- Are projected values acceptable/agreed to by the customer?
- Are the consequences of acceptance/agreement concerning status information identified?
- If applicable, have penalties and/or rewards regulations been established?

Are overall LSA strategies and principles established that influence these?

- Identify the preferred maintenance concept (eg, is two level maintenance mandatory?)
- Identify the supply chain management support concept
- Have the repairs focused on one maintenance level and, if so which level?
- Is repair at Maintenance Level1 and/or 2 limited?
- Is single source principle determined?
- Is an interim support concept applicable?
- Is COTS concept to be considered?

3.5.2 Documentation of the results of the LSA GC

The details of the LSA GC checklist must be documented as part of the GCD.

4 LSA guidance conference

The LSA GC should be the central event with participation of management staff and specialists from both the customer and the contractor. At this conference, the binding agreements for the performance of the LSA process must be established. To ensure best performance of this conference, it is necessary to have prepared inputs and to have a clear expectation of the results and final agreements. It is strongly recommended to have checklists for the LSA GC preparations and expectations.

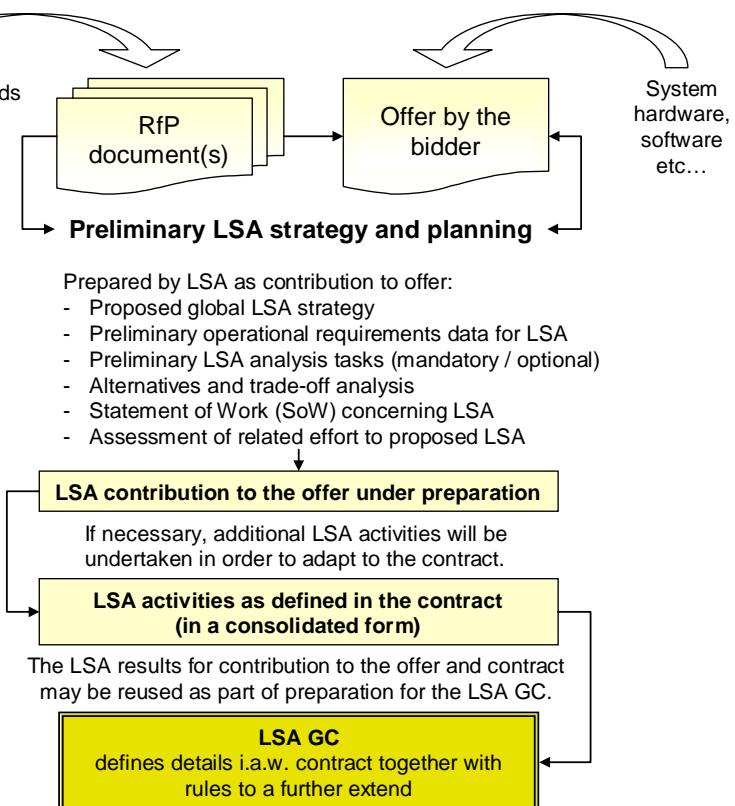
Note

It is recommended that "to be determined" (TBD) is not used as an input for any subject to be discussed in the LSA GC. Discussions should always be based on concrete proposals prepared by specialists, on how to realize the required task. Evaluate the advantages and disadvantages (cost/benefit) of the alternatives under consideration. Provide documentation of all alternative analysis and decisions.

4.1 The transition from a request for proposal to the LSA guidance conference

The majority of essential decisions influencing the LSA effort must be negotiated before the LSA GC. Usually, the LSA process begins while creating an offer and will be similar to the final version contained in the corresponding contract. This applies to any LSA significant aspect within the contract (eg, related Statement of Work) as well as for contractual details such as eg deliverable items, indispensable specified values or major milestones. This implies a series of investigations to be carried out prior to the contractual offer (eg, identification of LSA activities considered as mandatory, recommended or voluntary, depending on early strategy judgment and/or the kind of systems and equipment to be assessed). Refer to [Fig 2](#).

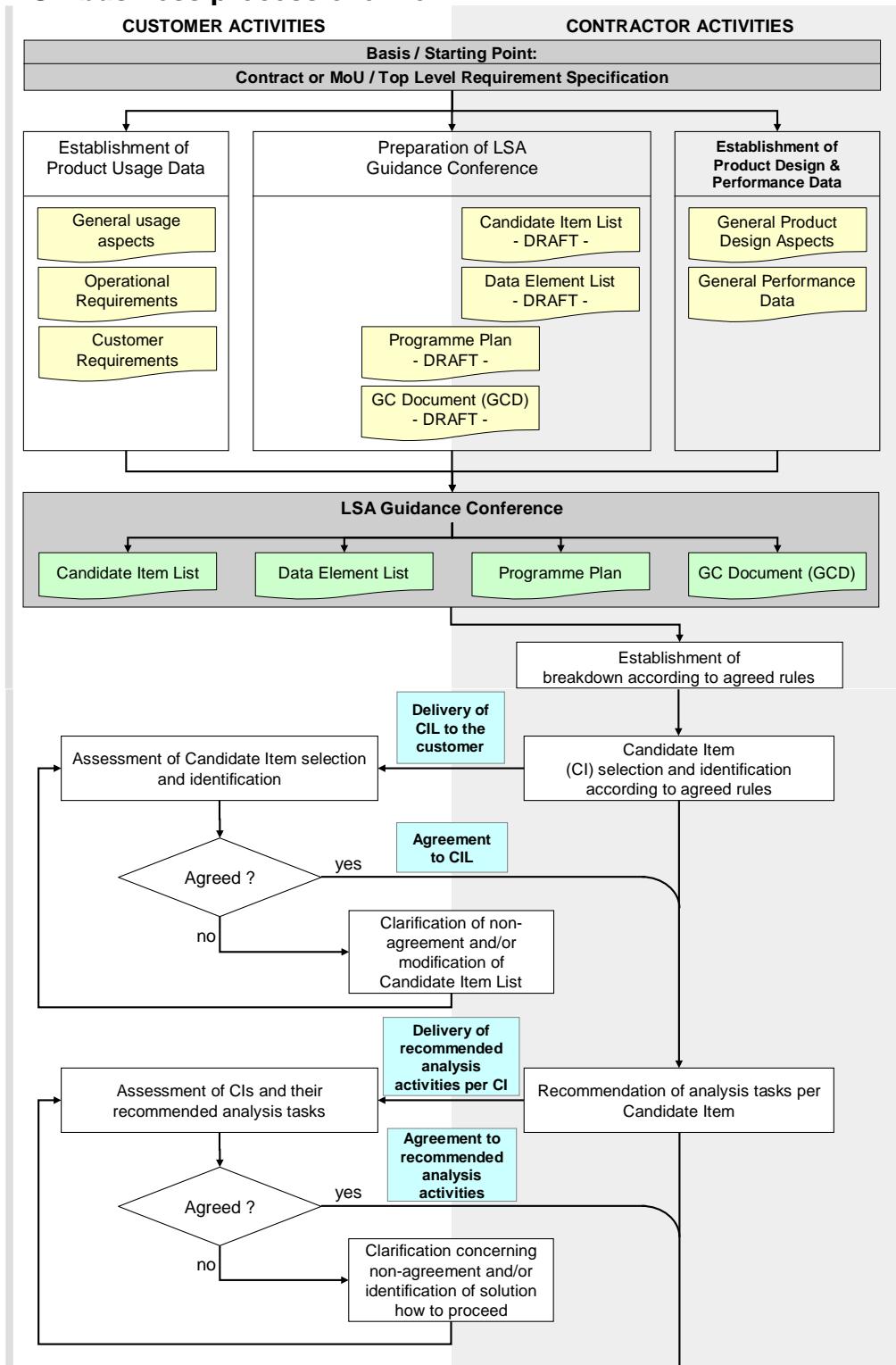
The LSA GC serves as a vehicle to communicate to the customer the work that will be carried out in detail, along with the associated rules and time schedules, based on the contractual requirements and further agreements. The LSA GC also clarifies any questions the customer might have regarding the effort. Nevertheless, changes to the LSA work effort, to a certain extent, must be allowed during the LSA GC without the need of contract changes and changes in the cost of the effort. Considering the iterative nature of LSA activities, customization (tuning) must be possible in order to be flexible.



ICN-B6865-S3000L0003-002-01

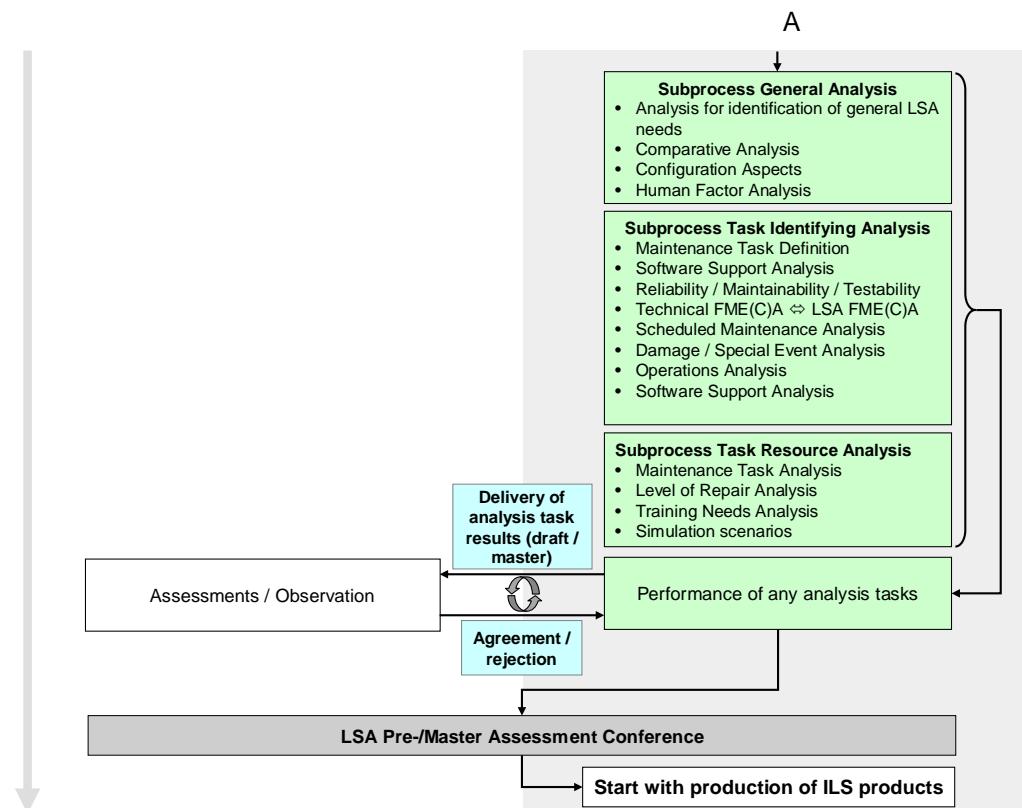
Fig 2 LSA activities in conjunction with the preparation of an offer/contract

4.2 LSA business process overview



ICN-B6865-S3000L0004-002-01

Fig 3 Flowchart of LSA business process at a glance (Sheet 1 of 2)



ICN-B6865-S3000L0004-002-01

Fig 4 Flowchart of LSA business process at a glance (Sheet 2 of 2)

The flowchart gives an overview of complete LSA process, from the collection of operational and customer requirements to the creation of the logistic products. Refer to [Fig 3](#) and [Fig 4](#).

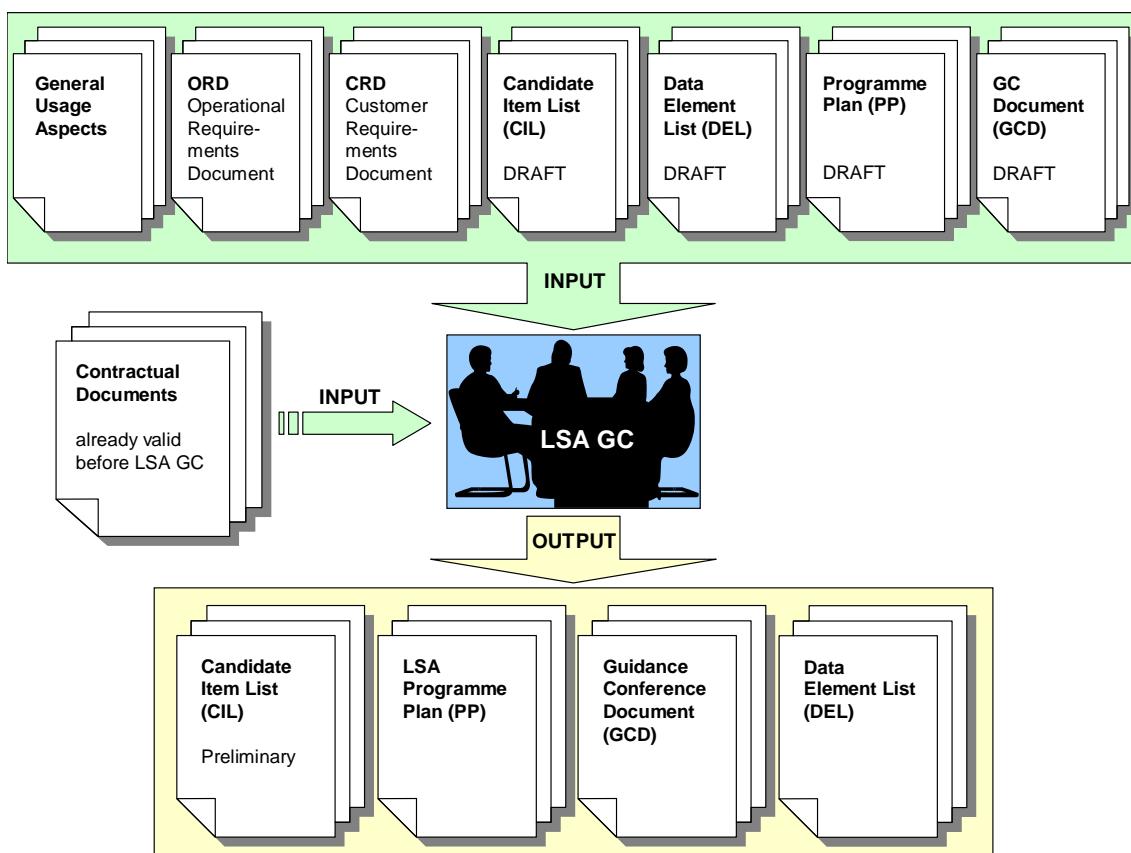
4.3

Documents and Information for LSA guidance conference

A list of applicable documents relevant for the LSA GC is given at [Para 4.3.1](#) and [Para 4.3.2](#). This list should provide a guideline for which information should be documented in which way. The project could add additional aspects or skip some information. [Fig 5](#) gives a summarized overview of the LSA GC concerning the required documents:

Note

The majority of essential decisions influencing the LSA effort should be negotiated before the LSA GC and documented in the contract. This applies to any LSA relevant aspect within the statement of work and the corresponding parts of the commercial offer, as well as for contractual details such as deliverable items, indispensable specified values and major milestones. This mandates a series of investigations to be carried out prior to the contractual offer (eg, identification of LSA activities considered as mandatory, recommended or voluntary, depending on early strategy judgment and/or the kind of systems/equipment to be assessed).



ICN-B6865-S3000L0005-001-01

Fig 5 LSA GC, inputs and outputs

4.3.1 Checklist of documents for the LSA guidance conference

[Table 2](#) provides an overview of required information is given to prepare for an LSA GC.

Table 2 Required input documents for LSA GC

Type of document	Content	Responsible for creation
General usage aspects	Strategic aspects of the Product design and performance data. Refer to Para 3 .	Customer
Contractual documents	Contractual decisions, which can influence the LSA process, must be available at the LSA GC and must be considered.	Customer and contractor
Operational Requirements Document (ORD)	Details about the usage of the systems in the planned scenario at all operational locations. Refer to Para 2 .	Customer
Customer Requirements Document (CRD)	Details of customer requirements regarding the operational scenario described in the ORD. Refer to Para 2 .	Customer
Candidate Item List (CIL)-DRAFT	A list of Breakdown Elements (BE) to be considered as potential LSA Candidate Items (CI) and for the performance of any logistic analysis activities	Contractor

Type of document	Content	Responsible for creation
Data element list (DEL)-DRAFT	A list of proposed data elements required to document the performance of all potential logistic analysis activities. This list should be an appendix to the preliminary GCD.	Contractor
LSA Program Plan (PP)-DRAFT	Management plan for the LSA program	Customer and contractor
Guidance Conference Document (GCD)-DRAFT	Implementation rules for the LSA program	Customer and contractor

4.3.2 Checklist of documents and information from the LSA guidance conference

The result of the LSA GC must be an integrated consensus of measurable rules for guiding the LSA process. These rules, checklists and documents provide a clear road map of needs and expectations between the contractor and the customer. This collection of official documents must be agreed to and signed by the customer and the contractor. At a minimum, these documents must contain the items shown in [Table 3](#).

Table 3 List of output documents of the LSA GC

Type of document	Content	Responsible for creation
Preliminary Candidate Item List (CIL)	List of LSA candidates containing all BEs that have been selected for logistic analysis activities, including the analysis activities to be performed for each BE.	Customer and contractor
LSA Program Plan (PP)	Initial release of the preliminary LSA PP. For details concerning an LSA PP. Refer to Chap 2 . LSA GC consensus of attending members.	Customer and contractor
LSA GC Document (GCD)	Initial release of the LSA GCD. For details concerning an LSA GDC. Refer to Chap 2 . LSA GC consensus of attending members.	Customer and contractor

4.4 Candidate item list

The creation of the CIL including the associated business rules is described in detail in [Para 5](#).

4.5 Contractual documents

Contractual conditions are the basis for a project. All contractual decisions that were made prior to the LSA GC must be considered. If these decisions impact the ILS process itself, the selection of the logistic analysis activities will be influenced significantly.

Another contractual consideration is the significance of the output documents of the LSA GC. The supporting LSA GC documentation should support contractual requirements and needs for both the contractor and the customer.

5

Candidate item selection and identification

The LSA process should be seen as considerably cost intensive. For that reason, the selection of the LSA candidates and the associated analysis activities should be carried out with adequate care in order to keep a good balance between the effort for LSA and the benefits acquired from the LSA process. The candidates can be of different quality, which requires the type of analysis processes and the depth of the analysis to be dependent on the particular LSA candidate.

The selection of the LSA candidates requires a proper Product breakdown, which must be driven by logistic aspects. The configuration of the IUA is of crucial importance. The establishment of the logistic breakdown is the first and the baseline analysis which must be performed as a leading activity for all further technical/logistic analysis which is performed during the entire LSA process. The representation of a Product by a Product breakdown must include a set of aspects which are summarized by the general term "configuration" of a Product:

- Types of Product breakdown (functional/physical)
- Installation location and realization by hardware (parts) or software
- Multiple installation of same part at different installation locations
- Alternate parts for the realization at one and the same installation location
- Substitute components for the repair of equipment
- Handling of variants of equipment/components within variants of the end Product (configuration aspect)

In [Chap 4](#), all aspects of configuration management are described in detail. This also includes the description of all details concerning establishment of a proper Product breakdown.

5.1

Definitions and general terms

First, some definitions of general terms used in the logistics environment concerning systems/items impacted by logistic support analysis will be introduced.

5.1.1

Candidate and non-candidate

The LSA candidate is the driver for all LSA activities. A potential LSA candidate can, in general, be an item on system, subsystem, equipment, module or sub module level, which is repairable or which requires maintenance support, scheduled or unscheduled. The decision on whether an IUA will be part of the CIL is dependent upon the criteria above, plus some additional criteria which must be defined individually by the project.

In addition to this hardware related definition, special activities or descriptions can be part of a Product breakdown. For example, the description of standard practices for the repair of structural components can be documented within a special chapter of the Product breakdown. In this case, these specific non-hardware breakdown elements also can be LSA candidates, because, for example, special events or maintenance tasks are addressed to them.

For non-candidate items, a detailed logistic analysis is not required. These items do not need to appear in a physical breakdown under a unique Breakdown Element Identifier (BEI). If a deep breakdown is established, all items that do not fit into any of the selecting criteria given in the LSA candidate selection checklist are potential non-candidates. Breakdown items that are only subject to standard tasks, for which no special logistic resource is required, are also potential non-candidates. Typical examples for non-candidates are consumables and bulk items, such as screws, bolts, nuts or washers. Nevertheless, even though an item could be identified as a non-candidate during the LSA candidate selection process, the result needs to be documented in the CIL.

5.1.2 Maintenance relevant items and maintenance significant items

For a clear definition of existing and new terms, a separation between the established term Maintenance Significant Item (MSI) and the definition of the LSA candidate must be established. Any IUA that is impacted by any maintenance action must be regarded as relevant for logistic analysis. The type of maintenance tasks is divided into two general categories: the scheduled or preventive maintenance and the unscheduled maintenance (failure or damage correcting). Therefore, the terms are defined in [Table 4](#).

Table 4 Definitions of MSI and MRI

Item type	Definition
Maintenance Relevant Item	A Maintenance Relevant Item is an item that can be repaired or replaced in case of a failure or damage. Normally this item is automatically a potential LSA candidate, but not automatically an MSI.
Maintenance Significant Item	A Maintenance Significant Item is an item that was identified by any selection process as the result of a Scheduled Maintenance Analysis (SMA) procedure such as S4000P or Reliability Centered Maintenance (RCM). For this item, an SMA will be performed (eg, a system and power plant analysis as described in S4000P). An MSI can become an LSA candidate if the SMA identifies a scheduled task. This task will be documented in the LSA database.

5.1.3 Structural items, structure significant items and structural details

The structure of a system, such as the bodywork of a car, is also part of the Product breakdown. Structural items can be typical LSA candidates and also candidates for SMA. Refer to [Table 5](#).

Table 5 Definitions of Structural Item, Structure Significant Item and Structural Detail

Item type	Definition
Structural Item	A Structural Item is a part of the system's bodywork. It can be an LSA candidate as well as a Structure Significant Item.
Structure Significant Item	A Structure Significant Item is an item which was identified by any selection process from an SMA procedure such as S4000P. For this item, an SMA will be performed (eg, Structure Analysis as described in S4000P).
Structural Detail	A Structural Detail is a specific area of an SSI which was identified by any selection process from an SMA procedure such as S4000P. For this detail, an SMA will be performed. In the breakdown, a structural detail can be identified by an additional artificial BEI below the super ordinate structural component.

Note

A detailed classification of structural items, with impact on a Product breakdown, and a definition of zones are given in S4000P for the application of the SMA.

5.1.4 Non-hardware items

Within a hierarchical hardware breakdown, only hardware is normally documented. If it is necessary to describe any general tasks which cannot be assigned to a specific hardware or which are standard procedures for a group of items (eg, repair procedures for electrical wiring), non-hardware items must exist within the Product breakdown. These non-hardware items can become LSA candidates, too.

5.2

Classification of LSA candidates

It is recommended that categories of LSA candidates be established. This enables the differentiation between the needs and to reduce the effort to an adequate level for each category of LSA candidate. The LSA candidates can be classified as shown in [Table 6](#). For all candidate types, the exact meaning of every classification must be defined and harmonized with the customer, especially for the partial candidates, as it could be necessary to define more than one partial type within a certain project.

Table 6 LSA candidate categories

Candidate category	Description
Full candidate	Provide full scale of selected LSA information applicable to the related item.
Partial candidate	Provide a partial scale of selected LSA information applicable to the related item, (eg, only remove and install information because of the need to gain access to other items, but no repair information required because the item is a discard item).
Candidate family	Provide LSA information focused on specific requirements, (eg, for a family of items such as non-specific wirings, harnesses, pipes, lines, minor items of the same type such as clamps, harnesses, connectors) Those items could need only a summary analysis for each group.
Standard procedures candidate	Provide a partial scale of LSA information because relevant information for this item is already covered by the LSA information of other non-hardware BEI (eg, tasks concerning standard repair procedures of structure or electrical wiring).

For every LSA candidate type, a list of criteria must be established. Together with these criteria lists, a flowchart for the decision process should be developed to support the logisticians.

5.3

Selection process and criteria list

For LSA candidate selection, each breakdown item must be assessed concerning its significance for LSA purposes. If the agreed selection matrix indicates that an item is a candidate according to the rules for selection, the relevant item will be noted as an LSA candidate.

The LSA candidate selection process should be adapted to each project. [Para 5.3.1](#) thru [Para 5.3.4](#) give examples for typical selection criteria. Additionally, LSA candidate selection flowcharts are given in [Para 10.3](#). Depending on the project, some questions in the flowcharts can be modified or additional questions can appear in the process.

To perform the LSA candidate selection process, some preconditions concerning Product breakdown and the establishment of rules must be fulfilled.

5.3.1

Preconditions for the selection process - Product breakdown

LSA candidates will be derived from a hierarchical hardware breakdown of the System Under Analysis (SUA) from the Product level to sub-module level. Therefore, the most important precondition to be able to perform an LSA candidate selection is the availability of a Product breakdown. It is essential that the existing breakdown is applicable for this task by examination of the following criteria:

- Is the Product breakdown available in a sufficient depth and extent (refer to [Chap 4](#))?
- Is the Product breakdown a preliminary issue (eg, for responding to a request by a potential customer or for pre-contractual agreements with a potential customer)?

- Is the existing Product breakdown already agreed by the customer?

5.3.2 Preconditions for the selection process - Existing analysis results

Each existing analysis result (eg, from analysis activities in advance or directly from the manufacturer) should be used for candidate selection. Examples are:

- Existing experience for equipment already used in other Products
- Existing maintainability or reliability data
- Existing Failure Mode and Effects Analysis (FMEA) or Failure Mode and Effects Criticality Analysis (FMECA) results

5.3.3 Preconditions for the selection process - Candidate selection rules

The establishment of the rules for LSA candidate selection and the extent of analysis must be agreed to by the contractor and the customer as a part of the LSA GC. This means that the final CIL cannot be presented at the beginning of an LSA GC (only a preliminary CIL can be published). Some criteria that can be used as a basic guideline for potentially relevant aspects for an LSA candidate selection include but are not limited to:

- The item is maintenance significant concerning the related failure/defect frequency or the awaited workload
- The item needs special logistic requirements such as personnel, training, material or support equipment
- The item is subject to scheduled or preventive maintenance (eg, preventive change of life-limited items)
- The item is subject to special procedures, (eg, after special events like lightning-, bird-, hail-strikes, contact with obstacles)
- The item is subject to diagnostic and/or functional test tasks (eg, complete system tests linked to a high level BEI)
- The item is potentially endangered by damages due to the installation area and/or the design of an item
- The item is subject to the use of new technology
- The item contains user loadable software (including data)

The following items should be considered as a potential LSA candidate that needs to be assessed for a global point of view (for special interests):

- The item is a potential readiness driver
- The item is a potential cost driver (eg, expensive support equipment required)
- The item is a potential maintenance driver (eg, high workload expected)
- The item is subject to emphatically customer interest
- The item is subject to LSA related contractual fulfillment

Items that be considered as potential LSA candidates that need to be assessed only or mainly for standardization reasons are:

- The items require only removal and installation in order to gain and undo access to an LSA candidate. No true LSA activities are required concerning these items, but the involved logistic requirements should be standardized and documented.
- Documentation of general tasks within the LSA database. Those tasks need to be incorporated in the LSA database mainly for registration and documentation of the associated logistic requirements. Often, the general tasks are linked to non-hardware BEI.
- Groups of items that could require a summarized logistic analysis (eg, a family of non-specific wirings, harnesses, pipes, lines, minor items of the same type like clamps)

All the criteria above must be detailed using measurable threshold values. For example, the criteria indicating whether an item is maintenance significant concerning the related failure/defect frequency, must be specified with a clear threshold value such as the MTBF. The

criteria above fit for all types of candidates. The possible criteria for the differentiation of the candidate types are described in [Para 5.3.4](#)

5.3.4 Preconditions for the selection process - Candidate classification selection rules

In [Table 7](#) thru [Table 9](#), examples for criteria for the different candidate types are given to support this important task of the LSA candidate selection process. Flowcharts for candidate item selection are given in [Para 10.3](#).

Table 7 Classification criteria for full LSA candidates

Candidate classification	Classification criteria	Required answer
Full candidate	Is the item a newly developed or a major modified item (functional equipment, not valid for structure)?	Yes
	Is the item a Line Replaceable Unit (LRU)?	Yes
	Is the item a repairable item?	Yes
	Is the item of low reliability (must be defined in measurable way)?	Yes
	Is the item maintenance relevant? In other words, are there any dedicated maintenance tasks planned such as repair, servicing, lubrication or calibration (no general tasks such as simple cleaning or a simple replacement).	Yes
	Are dedicated maintenance tasks complex, very time consuming or personnel intensive?	Yes
	Is the required support equipment for dedicated maintenance tasks a non-standard or a non-existent item?	Yes
	Does the item need specific scheduled or preventive maintenance identified in an SMA?	Yes

If all of the questions from [Table 7](#) Classification criteria for full LSA candidates are answered with "No", the IUA will not be a full candidate. If any question is answered with "Yes", then this item is a potential full candidate and can be classified as such in the CIL.

Table 8 Classification criteria for partial LSA candidates

Candidate classification	Classification criteria	Required answer
Partial candidate	Must the item be removed to gain access?	Yes
	Is a removal for access frequent for the IUA?	Yes
	Is the item a system or a subsystem that has not been previously defined as a full candidate, for which a fault location and/or a test procedure or other general maintenance procedures will be described?	Yes
	Is the item a non-hardware item, for which general activities (eg, cleaning, storing, parking, mooring, general inspections) will be described?	Yes

If all of the questions from [Table 8](#) Classification criteria for partial LSA candidates are answered with "No", the IUA will not be a partial candidate. If any question is answered with "Yes", then this item is a potential partial candidate and can be classified as such in the CIL.

Table 9 Classification criteria for LSA candidate family

Candidate classification	Classification criteria	Required answer
Candidate family	Is the IUA installed within the system many times in the same or similar way?	Yes
	Is it possible to combine all equal or similar items to one family with common maintenance activities?	Yes

The LSA candidate family concept should be used when a large number of equal or similar items are installed within the SUA. For example, all electrical wiring with the same properties and the same or similar connectors can be summarized under one LSA candidate family. For this family, all maintenance relevant information (eg, a connector repair concept) is valid for all single cables of the whole family, and can be documented for a family BEI.

5.4

Influencing factors

The following factors should be taken into account for candidate item selection and/or establishing related rules:

- Project phase to which the LSA program belongs
- Complexity of the Product to be analyzed
- Price of the Product to be analyzed
- Budget available for the overall logistic support and budget specifically planned for the LSA process. The limitation of budget will always have an influence on the number of LSA candidates that can be analyzed. In these cases, a reduction in the real maintenance and cost drivers is required.
- Importance of the LSA results for the internal logistic disciplines and the information required by the customer and the industry management required to make decisions and to fulfill the contractual requirements of the LSA related items
- Items already in use under comparable conditions
 - Items already in use, but under not comparable conditions
 - COTS items usable with or without major modification
 - Items already available but requiring major modification
 - Newly developed items
- Information expected from the LSA process (to be harmonized with/agreed by the customer during the LSA GC)
- Identification of possible alternatives (for hardware, software, support concepts) and related consequences (eg, Maintenance Concept (MC), logistic support requirements)
- Proposal of recommended MC
- Identification of tasks associated with the intended MC
- Identification of related logistic support requirements
- Estimation of related logistic support costs (part of Life Cycle Costs (LCC))

5.5

Recommendations for LSA candidate selection

To give an overview to the analyst on how to handle LSA candidate selection criteria in detail, a list of recommendations concerning what is unquestionably required and what is nice to have. [Table 10](#) provides assistance in the selection process. From this table, the analyst can find

recommendations on how to determine whether it is mandatory, recommended, or voluntary for an IUA to be an LSA candidate.

Table 10 Recommendations for LSA candidate selection

Aspects	Description of criteria	Consideration as potential candidate item?
Risk and criticality	Must LSA candidates be integrated from separate LSA programs? LSA information to be integrated into one LSA program (developed outside) without any change (eg, LSA data for an engine derived from a separate contractor) or external LSA information to be integrated with adoption to an LSA program that requires changes to be harmonized with the original contractor.	Mandatory
	The IUA is subject to an SMA (S4000P, or RCM) and scheduled or preventive tasks are identified.	Mandatory
	The structural IUA is subject to an SMA (S4000P, or RCM) and scheduled or preventive tasks are identified.	Mandatory
	The IUA is subject to life limits.	Mandatory
	The IUA is subject to preventive maintenance, other than scheduled, or to special procedures (eg, after special events like lightning-, bird-, hail- strikes, contact with obstacles).	Voluntary
	The IUA is subject to the use of new technologies.	Mandatory
	The IUA is a potential cost driver (high value items, cost limits must be defined with the customer).	Mandatory
	The IUA is a potential readiness driver because of long repair times.	Mandatory
	The IUA is a potential maintenance driver (high workload)	Mandatory
	The IUA is of special customer interest (information values must be defined with the customer)	Mandatory
Item type	The IUA is subject to LSA related contractual fulfillment	Mandatory
	Item already in use under comparable conditions	Recommended
	Item already in use under non-comparable conditions	Mandatory
	Item already in use but requiring major modification	Recommended
	Item newly developed	Mandatory
	COTS items	Recommended
	Part of an maintenance significant "item family"	Voluntary
General attributes	Item containing user loadable SW and/or data	Mandatory
	Item is line replaceable	Mandatory
	Item is shop replaceable	Recommended

Aspects	Description of criteria	Consideration as potential candidate item?
Maintenance significance	Item is line repairable	Recommended
	Item is shop repairable	Recommended
	Item is repairable at customer depot or at industry level	Voluntary
	A FMEA/FMECA is available concerning the related item	Mandatory
	Item is potentially subject to LSA relevant servicing	Voluntary
	Item is potentially subject to (diagnostic/functional) test	Recommended
	Item is potentially endangered by damages	Voluntary
	Item is potentially subject to general tasks	Voluntary
	Item is potentially subject to summary analyses	Recommended
	Item is potentially subject to standard procedures	Voluntary
Item is subject only to gain/undo access to LSA candidates		Voluntary

5.6

Candidate item list (draft)

For LSA candidate selection purposes, a matrix must be used in order to structure and document the selection results. A first draft should be included in a recommendation list, prepared by the contractor, which reflects the proposed selection criteria. The draft of the CIL should be prepared by the contractor for the LSA GC. After harmonizing the proposed selection rules between the contractor and the customer, this draft can be reworked and published as the preliminary CIL.

5.7

Candidate item list as an output of LSA guidance conference

The CIL should be included in an obligatory list, reflecting the decisions that were made by the customer. This CIL must be a living document in order to document changes during the life cycles of the project. During the LSA GC (refer to [Para 4](#)) the Draft CIL must be can as one of the most important steps regarding contractual terms. In the CIL, a collection of all LSA candidates and the corresponding workload from the selected analysis activities must be managed. To support the LSA process management, the progress of the LSA and documentation work should be documented by status codes, which reflect the status for all the different contractual relevant analysis activities to be covered within the LSA process.

To increase effectiveness, it is recommended to integrate the CIL as a central management tool into the logistic database application so that it will be possible to hold all relevant information of the CIL in the breakdown of the system. An overview of the project can easily be given by the corresponding reports.

All changes in the CIL for whatever reasons (eg, technical, budgetary, redesign) must be discussed and harmonized between the customer and the contractor and documented. For the LSA managers and the ILS managers for both the customer and the contractor, the CIL is the valid document for control of the whole LSA process. The customer and the contractor must understand, at every stage of the project, who is the holder of the contractually valid issue of the CIL.

6 Logistic analysis activities

In the early life cycle, LSA must support cost relevant decisions by performance of adequate front end analysis (eg, Level of Repair Analysis (LORA)). Where necessary, LSA must influence design concerning logistic aspects from the beginning of the definition phase and this task must be continued during the lifetime of the Product. The inputs for the design come from different analysis sources such as reliability, maintainability, testability or SMA. The performance of analysis activities for LSA candidates can be a cost concern if no threshold is established to reduce the effort to a suitable level. In order to balance between analysis-related efforts and acquired knowledge which is mandatory to identify adequate logistic requirements (that satisfies the user's needs and logistic support cost constraints), a series of assessments is required. Related allocation criteria and rules must be identified by the contractor and proposed to the customer as general guidelines tailored to meet the specific requirements of a given project.

Note

This has a large cost influence on both the effort to be spent for the LSA program itself and on the generation of future support costs during the overall life cycle of the items under analysis. It is strongly recommended to perform this task on a preliminary basis as an initial assessment early in a project and to harmonize the derived strategy with the customer before signing a contract.

6.1

Principles for analysis activity selection

Only those analysis activities should be selected for an LSA candidate that are considered to improve the investigation result such a way that the expected benefit is greater than the effort to be spent for the intended analysis activity itself. Prior to the allocation of potential analysis activities to LSA candidates, they should be grouped into categories in order to identify the real need for analysis activities to reduce the effort to an adequate level. Refer to [Chap. 4](#).

Knowledge acquired by "lessons learned" should be taken into account by both the customer and the contractor side. This holds for including both, positive and negative experience. Based on that knowledge, LSA relevant decisions can be predetermined without further analysis activities.

New available analysis methods should be taken into account where feasible. For example, a simulation result is much more comprehensive than a traditional analysis.

Note

Based on the experience of Industry, powerful simulation software packages are available and are expected to become important for LSA purposes.

6.2

Potential analysis activities

The following LSA activities are a list of potential analysis which can be performed and documented in different ways depending on the kind of project. The LSA database must document the LSA activities and their results.

- Analysis for identification of general LSA needs
- Comparative analysis
- Human factor analysis
- Configuration assessment
- Reliability analysis assessment
- Maintainability analysis
- Testability analysis
- FMEA/FMECA
- Damage analysis
- Special event analysis
- SMA (eg, S4000P or RCM)
- LORA

- Maintenance Task Analysis (MTA)
- Software Support Analysis (SSA)
- Operations analysis
- Simulation operational scenarios
- Training Needs Analysis (TNA)

For each LSA activity, the contractor and the customer must agree whether, and how, the results of each analysis activity will be documented, delivered, commented and assessed. It is recommended that most of the results of the analysis activities be documented in an LSA database, but it is possible to document as a special analysis report or an official document. It must be clarified before the beginning of the analysis, the required results and the benefits from the collected data, and how the data will be evaluated and used for later logistic purposes.

6.2.1 Analysis for identification of general LSA needs

This starting point for any LSA activity is the basic precondition for applying any of the following tasks. Here, the justification for performing the analysis must be determined. Valuable resources should not be spent on analysis work without a clearly defined objective. The identification of general LSA needs is mainly related to contractual, customer and user interests. In order to ensure that the general LSA needs for the project are properly addressed and agreed with by the customer, aspect that must be addressed are:

- Consider the results of the establishment of Product usage data (ORD and CRD), intended support strategy and principles and alternative solutions. Clarify which scenarios must be analyzed, how and to what depth.
- Consider which contractual information must be demonstrated and validated by LSA results, how and to what depth
- Consider the definition of required reports concerning areas of special interest (eg, management reports, status overviews, performance measuring reports, verification reports)
- Consider trade off analysis results in order to guarantee the best return of investment. This means, for example, customer demands could be discussed to achieve the goal by some other method. The alternative solution might have significantly less effort, but it must lead to a comparative result which meets the requirements of the customer in a similar and acceptable level.

The result of these first considerations must be an agreement between the contractor and the customer on the purpose, goal, depth of analysis, and documentation method for each analysis type so each has a common view on the performance of LSA. The result of this analysis should be documented in the LSA program plan at the LSA GC.

6.2.2 Comparative analysis

This analysis will be performed upon special request only. As a precondition, comparative information of equipment, a system or an end item must be available and effective for decision making (limited to the initial LSA activities). If a comparative system can be defined, appropriate analyses must be performed in order to derive applicable comparative data. For the documentation of the results, special summary reports concerning comparative factors and data and/or support alternatives addressed to comparable LSA candidates must be identified.

6.2.3 Human factor analysis

This analysis may have influence on those LSA candidates for which an MTA is performed. Human factors can be a reason for special limitations (eg, concerning the applicability of a maintenance task which requires special abilities). Ergonomic aspects must be considered here, as well as the definition of rules for an appropriate man machine interface. Refer to [Chap 6](#). For example, some aspects that can have limiting consequences to LSA activities.

- Ability to lift and carry heavy loads
- Ability to move for a long time in special conditions

- Handling of dangerous materials
- Limitations of personnel employment by law (security restrictions)

6.2.4 Configuration assessment

In the case of significant deviating design configurations of any LSA candidate, configuration assessments are mandatory. Potential changes to validity, engineering changes, deviations or waivers must be examined. Configuration assessments must be performed as general assessments of the need of deviating maintenance concepts and/or logistic support requirements (eg, personnel, material) in order to re-assess the need for analysis activities on LSA candidates.

The customer and the contractor must clarify how configuration deviations of LSA candidates should be documented within a logistic database.

6.2.5 Assessment of reliability analysis

Reliability predictions (eg, MTBF) related to the intended usage scenario of the LSA candidates must be assessed regarding their use for LSA purposes. The determination of reliability values is a design and development related task, but the documentation of certain results of this analysis is of crucial importance for LSA purposes. Normally, a reliability analysis is mandatory for each hardware LSA candidate. Additionally, it is recommended to extend reliability predictions to all essential items of an LSA breakdown and to ensure consistency of these values over all breakdown cascades.

Note

The reliability values are directly linked to the results of a FMEA concerning ratios of different failures of an LSA candidate. These two areas in a logistic database must be consistent. Any LSA software package must be able to check this correlation between these two analysis areas automatically, otherwise the overview of reliability data consistency with FMEA data will be easily lost. Every LSA software package must be able to automatically check the consistency of reliability values over all breakdown cascades.

6.2.6 Maintainability analysis

For each LSA candidate, a maintainability analysis is strongly recommended in order to decide, from a technical point of view, whether an item is supportable or not (rules for related conditions and time frames should be established). Alternative repair solutions must be investigated, if applicable. Where applicable, a maintainability analysis must be applied that addresses:

- Assessment of maintainability features reflecting customer requirements
- Identification of maintenance tasks at different levels in order to support the maintenance concept for each applicable LSA candidate
- Investigation of supportability aspects such as modular design of end item, good accessibility, installation concept in special zones (the item with the worst MTBF must not be installed behind others)

In general, the maintainability analysis can be performed by following applications with different level of detail and effort:

- Best Engineering Judgment in the case of lowest level of available information
- Assessment with or without transformation of equipment supplier information. In the case of demanding information from suppliers, apply data sheet templates to ensure that every supplier will be able to deliver the required information correctly and completely. Harmonize these templates with the customer and the contractor (and contractor suppliers).
- Take into account supporting analyses such as LORA for comparison of alternatives and/or SMA methods, such as S4000P, or RCM, to build the maintenance concept for the IUA
- Use simulation tools for those Products performing missions according to operational profiles, focusing on maintenance and support while evaluating potential consequence from, for example, limited maintenance resources or changed operational profiles

Depending on the different types of items, the analyses given in [Table 11](#) must be applied as a minimum analysis:

Table 11 Depth of maintainability analysis depending of item type

Item	Maintainability analysis depth
Items currently used under comparable conditions	Moderate information and data transformation, a more detailed maintainability analysis is voluntary
Items currently in use, but not under comparable conditions	Careful data transformation is mandatory, a more detailed maintainability analysis under new conditions is recommended
COTS items without major modification	Non-compliances of logistic features and/or requirements should to be documented and agreed by the customer.
Items currently available requiring minor modification	Moderate information and data transformation, a more detailed maintainability analysis is voluntary
Items currently available requiring major modification	Careful data transformation is mandatory, a more detailed maintainability analysis under new conditions is strongly recommended
Newly developed items based on well-known technology	Detailed maintainability analysis is strongly recommended
Newly developed items based on new technology	Detailed maintainability analysis is mandatory

The results of the maintainability analysis must be carefully documented, therefore, maintainability reports should be defined and harmonized between the contractor and the customer. In addition to these reports, some results of the maintainability analysis will be documented within a logistic database within the maintenance concept reflected by the identified maintenance tasks.

6.2.7

Testability analysis

The main aspects for a testability analysis, which must be taken into consideration, are:

- Identification of maintainability strategy aspects, which must be observed
- Identification of overall test architecture and test principles
- Identification and verification of contractual testability requirements
- Evaluation of FMECA/FMEA documents concerning testability information
- Description of failure verification methods at equipment-, system- and end item level
- Description of functional check requirements at all involved levels
- Identification of related logistic resource requirements (eg, personnel, loadable software, test software)

The testability analysis is closely connected to the maintainability analysis. Results from testability directly influence the maintainability analysis. Any corrective maintenance concerning test capable items is dependent of the detectability of the failure to be corrected. If a failure cannot be verified by any test procedure, the corresponding repair task cannot be performed. In addition to these aspects, it is necessary to be able to perform a functionality test of an item after repair. The depth of failure detectability/verification is one of the most important influencing factors on an applicable maintenance concept.

Since it is nearly impossible to establish testability features after the design of an item has been finalized, testability requirements must be described and documented for each applicable item during the design and development phase. These requirements must be harmonized with the

customer during the LSA GC, however, testability capability should only be planned to the depth on which repair is intended.

For all items containing full Built in Test (BIT) capability, a testability analysis is mandatory. Related testability features must be identified and documented. A testability analysis report, as a summary of the analysis itself, is required. For equipment with reduced BIT capability items (eg, COTS equipment) that are somehow monitored within the overall test architecture, a testability analysis is recommended. Related testability features must be identified and documented.

The types of data concerning testability features must be documented in a logistic database or in special testability reports and must be established and harmonized with the customer. Rules for manufacturers for how to implement testability features must be established and verified (eg, by testability demonstrations/validations together with the contactor and/or the customer).

6.2.8 Analysis activities concerning event driven maintenance

There are events that can be the justifications for maintenance activities. An overview of the event driven maintenance is given in [Chap 12](#).

6.2.8.1 FMECA/FMEA

The aim of FMECA activities is to identify the potential failure conditions of a Product. If the criticality of failure conditions is not considered, FMEA is also used as a common term. A distinction is drawn between functional and technical failures. Functional failures are identified/analyzed by performing a System FMECA, technical failures are identified/analyzed by the Equipment FMECA approach.

Note

A detailed presentation of System FMECA and Equipment FMECA methodology can be found in SAE ARP5580.

6.2.8.1.1 System FMECA

A System FMECA analyzes the different systems of a Product using a top down approach concerning the potential functional failures, their functional failure effects with corresponding criticality and finally, the corresponding failure causes. Refer to S4000P.

Potential failure effects, which prove to be critical due to safety, legal or environmental integrity reasons must be avoided. Other failure effects, which are undesirable due to operational/mission related or economic reasons, should also be avoided. Analysis results are the baseline for the identification of preventive (scheduled) maintenance activities or even redesign requirements, if no scheduled task is applicable and/or effective. Refer to S4000P.

6.2.8.1.2 Equipment FMECA

An Equipment FMECA is performed to identify in detail the consequences and the probability of technical failures, which can occur to any equipment within a Product following a bottom up analysis approach.

Safety analysis activities (eg, fault tree analysis, FTA) are based on System and Equipment FMECA. Additional safety critical failure combinations can be identified in that way. Depending on probability values, redesign and/or preventive (scheduled) maintenance requirements must be defined (eg, must be harmonized with the results from the SMA).

Additionally, the Equipment FMECA results will be used to determine corrective (unscheduled) maintenance task like repair or complete replacement of equipment, including task frequencies. Refer to [Chap 7](#).

6.2.8.2 Damage analysis

Items susceptible to damage should be considered as at least partial LSA candidates. In general, damages are special events, for which a prediction of criticality or extent cannot be given for every case. Nevertheless, it is useful to identify classes of damages, which can be maintained in the same way or which can be detected in the same way. Analyses that at least address initial diagnostic tasks and simple repair procedures (eg, simple repair of damages such as scratches) are strongly recommended. The result of these analyses should be documented within general chapters in a logistic database, (eg the standard test and repair procedures for structural items or for electrical wiring is collected within a non-hardware chapter of the Product breakdown).

Because reliability values concerning damages are not normally available, general quantitative statements are generally not possible. Only with the help of statistical evaluations is it possible to get a prediction of the required effort resulting from damages. It is recommended to use such predictions with caution. Refer to [Chap 8](#).

6.2.8.3 Special event analysis

In addition to the occurrence of damages, which are a type of special event, other events can have an impact on the maintenance concept for an IUA or on the Product. It is strongly recommended to identify and to document probable special events that require special maintenance tasks in order to guarantee the proper functionality for the further usage. Also, the required maintenance tasks must be described and documented, information concerning required resources like personnel, material or software, within a logistic database. Refer to [Chap 8](#).

Some examples of special events that require maintenance tasks include:

- Exceeding temperature limitations
- Exceeding mechanical load limits (eg, over torque)
- Exceeding maximum allowed speed
- Operation in salt-laden atmosphere
- Operation in sand-laden atmosphere
- Lightning strike
- Hard landing of an aircraft
- Collision with external objects (eg, bird strike)

6.2.8.4 Scheduled maintenance analysis

A SMA is performed to ensure that safety relevant, highly economic or ecological failures are strictly avoided. This analysis is mandatory for safety relevant aspects and strongly recommended if significant economic or ecological disadvantages are to be expected. The SMA itself is an extensive analysis and the results have direct impact on the maintenance concept. Scheduled maintenance tasks, which are identified by the SMA from S4000P or RCM processes, must be documented in a logistic database. It is recommended that rules be established for how to harmonize SMA results with the logistic database documentation method.

Each item for which scheduled or preventive maintenance tasks are identified via an SMA process, should become a full LSA candidate. Refer to [Chap 10](#).

6.2.9 Level of repair analysis

Depending of the general support strategy, it could be necessary to decide for each item, the level at which maintenance and/or repair of the item will take place. To support this decision, a LORA can be performed by applying one of several methods, depending on the type of LSA candidate to be analyzed, the effort required and/or the information available. Depending on LSA GC agreements, different categories of LORA could be assigned, as shown in the examples in [Table 12](#) Classification of different depths of LORA.

Table 12 Classification of different depths of LORA

LORA classification	Description
Simplified LORA	Can be derived on the basis of Best Engineering Judgment by simply taking into account best information available. The results of this analysis must be documented and harmonized with the customer through, for example, a "Simplified LORA report".
Full LORA	A front end analysis using software packages available based on mathematical models of different complexity and accuracy. All required data concerning the item itself and the context of its use (eg, operational scenario, spare part provision, costs) must exist in order to perform such an intensive analysis.
Analysis via simulation	A simulation that considers very detailed information about the IUA and its deployment and operational scenario can produce the best LORA results. This requires a complete set of information/data in the required format for the simulation software package to be used. The preparation of this data can cause a lot of additional effort.

In any case, where applicable, a LORA analysis must be considered to be a mandatory analysis activity. This holds for LSA candidates with alternative repair and/or support solutions concerning personnel, material and/or user loadable software or data as well as for scheduled maintenance tasks derived from an SMA or equivalent and/or servicing tasks, if contracted. Refer to [Chap 11](#).

6.2.10 Maintenance task analysis

The MTA is one of the central analysis activities within the LSA process. Here, the identified maintenance tasks (both scheduled and unscheduled) are detailed with all required information available. The following list gives a limited overview, for more detailed information refer to [Chap 12](#).

- Documentation of general task information such as preconditions for task performance, training requirements or criticality information
- Assignment of maintenance tasks to the identified events (eg, failures, damages, special events, time limitations)
- Rough task description including the sequence of subtasks
- Identification of related logistic resource requirements (eg, personnel, support equipment, spares, facilities, software)
- Time estimations
- Calculation of task frequencies
- Consideration of required pre- and post- tasks (eg, test, fault location, gaining access)

6.2.11 Software support analysis

Each item containing user loadable data/software must be analyzed with regards to loading, de-loading, transportation and documentation of existing Software (SW) releases. This covers maintenance and servicing tasks applicable for:

- Maintenance of hardware containing user loadable data/SW
- Data and/or operational SW required for usage preparation
- Software support in order to update items with new user loadable SW releases

If an SSA program will manage SW changes, SW releases, and the maintenance of SW in case of SW failures (bug fixing), a process aligned with the LSA process for hardware should be established. Refer to [Chap 13](#).

6.2.12 Operations analysis

Operational requirements must be analyzed to identify tasks that are important for the general handling of the SUA. These tasks should also be documented in a logistic database. For logistic purposes, it is necessary to analyze such tasks to identify the requirements concerning personnel, support equipment or material. Refer to [Chap 9](#).

6.2.13 Simulation of operational scenarios

This analysis will be performed by special request only. Simulation represents a very powerful ability to evaluate operational scenarios in combination with logistic scenarios. Today, simulation SW packages offer the opportunity to analyze and optimize the planned usage of a system under particular logistic conditions. Only the combination of these two aspects, planned usage scenario and planned logistic scenario, makes it possible to generate meaningful analysis results. The better the scenario can be described, the better the results will be.

Simulation analysis needs a lot of input data with a certain depth. This causes additional effort, which must be considered. If simulation is selected as a required analysis activity, it is strongly recommended to start early with the examination of the data needs of the simulation SW package. The decision to perform simulation can influence the depth and the requirements for the quality of any logistic data (eg, the detailed description of a logistic scenario can cause additional effort within an MTA). Also, the creation of an operational scenario can cause additional effort for the customer.

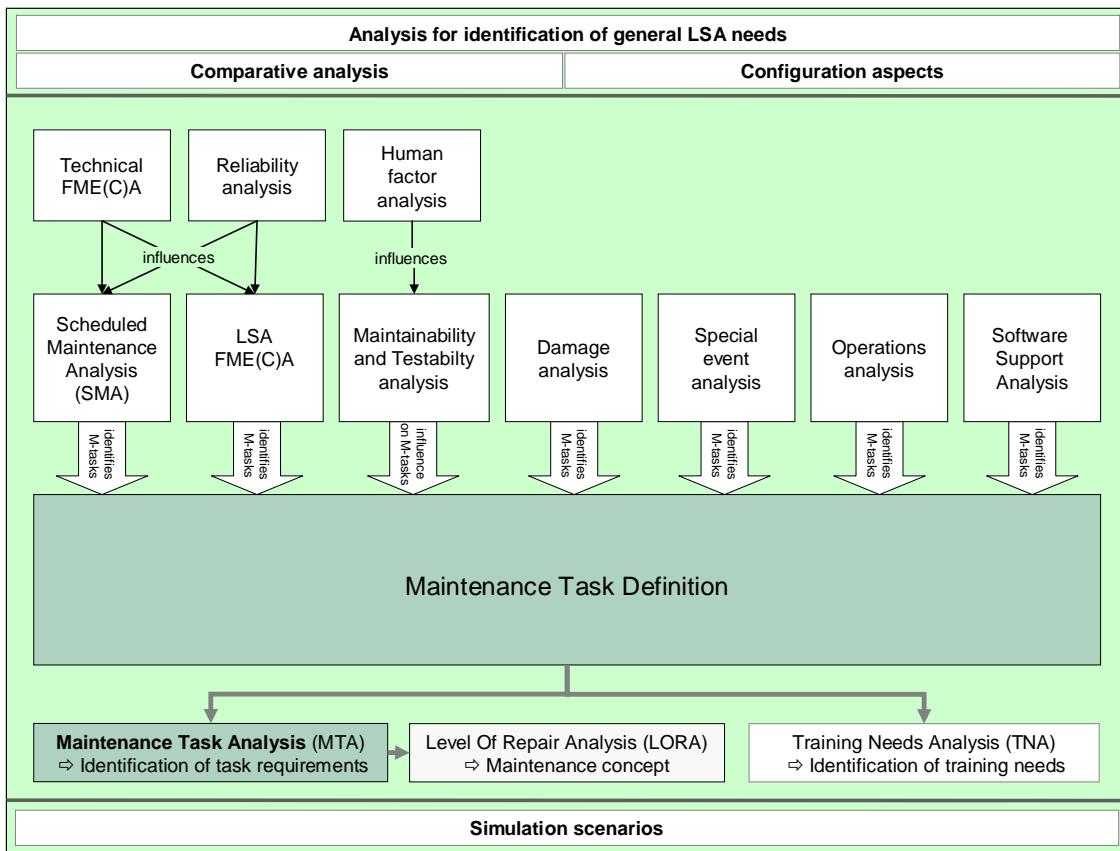
6.2.14 Training needs analysis

Within this analysis, decided decision must be made as to whether a task requires special training or not. If training is required, how the training can be applied most effectively must be determined. This process can be supported with the help of the content of the LSA database concerning the identified tasks.

The criteria for a training requirement should be discussed and harmonized between the customer and the contractor during the LSA GC. It is recommended that an IT supported process for the TNA be established. The criteria should be "translated" to the LSA database system and each task should be examined against these criteria. The result can be a preliminary TNA decision based on existing data concerning tasks within an LSA database.

6.3 Analysis relations and general overview

All the analysis types described above are interconnected and influence each other. Some analysis types are a basic precondition for other ongoing analysis processes. Some analysis types are of a general significance for all other analysis procedures. [Fig 6](#) shows a graphical depiction of the relations between the different logistic analyses.



ICN-B6865-S3000L0018-001-01

Fig 6 Analysis activities relations and overview

6.4

Analysis activities selection criteria

The aspects given in [Table 13](#) must be considered in order to select and allocate LSA relevant analysis activities to LSA candidates that are applicable and effective depending on the type of items, candidates and required information.

The selection criteria and aspects must be tailored for each project by taking into account any specific circumstances and they must be harmonized between the contractor/industry partner companies and the customer within the LSA GC.

Table 13 Analysis activities selection criteria

Criteria group	Criteria	Recommendation
Items with existing experience	Items already in use under comparable conditions	Moderate transformation of existing analysis data
	Items already in use but under non-comparable conditions	Careful transformation of existing analysis data
	COTS items usable without major modification	Moderate transformation of existing analysis data
	Items currently available that would require (minor/major) modification	Careful transformation of existing analysis data or performance of new analysis
	Newly developed items based on well-known technology	Performance of analysis activities in required depth strongly recommended
	Newly developed items based on new technology	Performance of analysis activities in required depth mandatory
Items that need to be assessed for global point of view or for special interests	Potential readiness drivers and/or cost drivers	Criteria for these items must be detailed within the LSA GC
	Subject to emphatic customer interest	Items must be detailed within the LSA GC
	Subject to LSA related contractual fulfillment	Items must be detailed within the LSA GC
Items that need to be assessed for inherent reasons or due to the installation area	Item is maintenance significant concerning the related failure/defect frequency, workload, special logistic requirements (personnel and/or material)	Criteria for these items must be detailed within the LSA GC
	Item is subject to scheduled maintenance or to life limits	Criteria for SMA must be detailed within the LSA GC
	Item is subject to preventive maintenance resulting from special events	Criteria for special event analysis must be detailed within the LSA GC
	Item is potentially susceptible to damages due to its installation area	Criteria for damage analysis must be detailed within the LSA GC
Type of LSA candidates	Full candidate	Subject to applicable analysis, results documented within LSA
	Partial candidate	Rudimentary information to be documented in LSA
	Candidate families	A group of items considered as one LSA candidate, subject to applicable analysis, results documented within LSA

Criteria group	Criteria	Recommendation
	Items subject to standard procedures	Rudimentary information to be documented in LSA
Subject to request for information by the customer	Information expected from the LSA process	To be harmonized with and agreed to by the customer during the LSA GC
	Proposal of recommended logistic support principles	To be harmonized with and agreed to by the customer during the LSA GC
	Identification of possible alternatives (for hardware, software, support concepts) and related consequences (eg, MC, logistic support requirements)	To be harmonized with and agreed to by the customer during the LSA GC
	Proposal of recommended maintenance concepts including corresponding tasks	To be harmonized with and agreed to by the customer during the LSA GC
	Estimation of related logistic support costs	Identification of related logistic support requirements Estimation of related logistic support costs Information for early system/project decisions (significant costs influence)

6.5

6.5.1

Checklist for analysis activity recommendation

Recommendation and decision sheet

For analysis activities selection purposes, a selection matrix must be used in order to structure and document the selection and decision results.

6.5.1.1

Analysis activities recommendation sheet:

This matrix should be prepared by the contractor for the LSA GC, identifying the LSA candidates, the task selection criteria and the derived results with recommendations. For all general tasks not driven by the LSA candidates, recommendations must be documented in independent documents. This applies:

- Analysis for identification of general LSA needs
- Configuration assessment/aspects
- Availability of a FMECA/FMEA
- Human factor analysis

6.5.1.2

Analysis activities decision sheet:

This matrix will be harmonized and finalized at the LSA GC and reflects the decisions made by the customer. This contractually relevant document should be established as a living document in order to reflect changes, to incorporate "lessons learned" results and enable traceability during the life cycles of the project. As a recommendation, a preliminary list could become part of a commercial proposal in order to introduce a general guideline for LSA task allocation to the customer for risk reduction purposes.

6.5.2

Selection matrix example and rating values

In order to identify the relative importance of the allocated analysis activities, "ratings" should be agreed to concerning both the importance of the selected LSA candidate itself as well as the

importance of the potential analysis activity. The result can be used to rank LSA candidates and analysis activities from mandatory down to voluntary. This ranking could be used to refine the selection of candidates/analysis activities in case of eg limitations of budget or time constraints. The results of this selection procedure would be the initial issue of the LSA CIL as well as the range of LSA activities considered as relevant and effective for the LSA candidates. Appropriate rules must be identified and harmonized within the industry partner companies and agreed to by the customer during the LSA GC.

CANDIDATE ITEM LIST (CIL)			0 = not applicable 1 = voluntary 2 = recommended 3 = strongly recommended 4 = mandatory																			
BEI	Candidate Type	Element Name	General LSA Needs	Comparative Analysis	Human Factor Analysis	Configuration Assessment	Reliability Analysis available?	Maintainability Analysis	Testability Analysis	LSA FMEA	Damage Analysis	Special Event Analysis	Scheduled Maint. Analysis	Level of Repair Analysis	Maintenance Task Analysis	Software/Data Loading	Software Support Analysis	Operations Analysis	Simulation Scenarios	Training Needs Analysis	Ranking	
HEN	enditem	Vehicle	4	1	2	3	n	0	0	0	0	0	0	0	0	0	0	3	0	0	13	
HEN28	sys	Fuel System	0	1	0	3	n	0	0	0	0	0	0	0	0	0	0	0	3	0	0	7
HEN281	subsys	Fuel Storage	0	0	0	3	n	0	0	0	0	0	0	0	0	0	0	0	3	0	0	6
HEN2812	subsubsys	Internal Tanks	0	0	0	3	n	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
HEN281201	partCand	Fuel Tank 01 Assy	0	0	0	3	y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
HEN281201001	fullCand	Rubber Tank 01	0	0	0	3	y	4	4	4	4	4	4	4	3	4	0	0	0	0	4	38
HEN281201002	partCand	Cover Left	0	0	0	3	y	0	0	0	2	0	2	0	0	0	0	0	0	0	0	7
HEN281201003	partCand	Cover Assy Right	0	0	0	3	y	0	0	0	2	0	2	0	0	0	0	0	0	0	0	7
HEN281201003001	nonCand	Cover Right	0	0	0	3	n	0	0	0	2	0	2	0	0	0	0	0	0	0	0	7
HEN281201003002	nonCand	Socket Plate Assy	0	0	0	3	n	0	0	0	2	0	2	0	0	0	0	0	0	0	0	7
HEN281201003003	nonCand	Bracket Assy	0	0	0	3	n	0	0	0	2	0	2	0	0	0	0	0	0	0	0	7
HEN281201003004	nonCand	Safety Rope	0	0	0	3	n	0	0	0	2	0	2	0	0	0	0	0	0	0	0	7
HEN281201004	fullCand	Drain Valve Left	0	0	0	3	y	4	4	4	4	4	4	4	2	4	0	0	0	0	4	37
HEN281201005	fullCand	Drain Valve Right	0	0	0	3	y	4	4	4	4	4	4	4	2	4	0	0	0	0	4	37
HEN281201006	nonCand	Blanking Plug	0	0	0	3	n	0	0	0	2	0	0	0	0	0	0	0	0	0	0	5
HEN281202	nonCand	Fuel Tank 02 Assy	0	0	0	3	n	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
HEN281202001	partCand	Rubber Tank 02 EQ	0	0	0	3	y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
HEN281202001001	fullCand	Rubber Tank 02	0	0	0	3	y	4	4	4	4	4	4	4	3	4	0	0	0	0	4	38
HEN281202001002	nonCand	Tank Flange Assy L1	0	0	0	3	y	0	0	0	2	0	1	0	0	0	0	0	0	0	0	6
HEN281202001003	nonCand	Tank Flange Assy R1	0	0	0	3	y	0	0	0	2	0	1	0	0	0	0	0	0	0	0	6
HEN281202001004	nonCand	Tank Flange Assy L2	0	0	0	3	y	0	0	0	2	0	1	0	0	0	0	0	0	0	0	6
HEN281202001005	nonCand	Tank Flange Assy R2	0	0	0	3	y	0	0	0	2	0	1	0	0	0	0	0	0	0	0	6
HEN281202001006	nonCand	Intercon Tubes R	0	0	0	3	y	0	0	0	2	0	1	0	0	0	0	0	0	0	0	6
HEN281202001007	nonCand	Intercon Tubes L	0	0	0	3	y	0	0	0	2	0	1	0	0	0	0	0	0	0	0	6
HEN281202002	partCand	Cover Left	0	0	0	3	y	0	0	0	2	0	1	0	0	0	0	0	0	0	0	6
HEN281202003	partCand	Cover Right	0	0	0	3	y	0	0	0	2	0	1	0	0	0	0	0	0	0	0	6
.....																						

ICN-B6865-S3000L0019-002-01

Fig 7 Example of a simple analysis activity recommendation sheet

Fig 7 shows a simplified example of a recommendation matrix for the selection of analysis activities. For an actual project, it could be necessary to go into further detail, for example, the performance of a LORA can be divided up in more categories such as simplified LORA or full LORA with the help of a special software package. Additionally, the rating values can be more detailed.

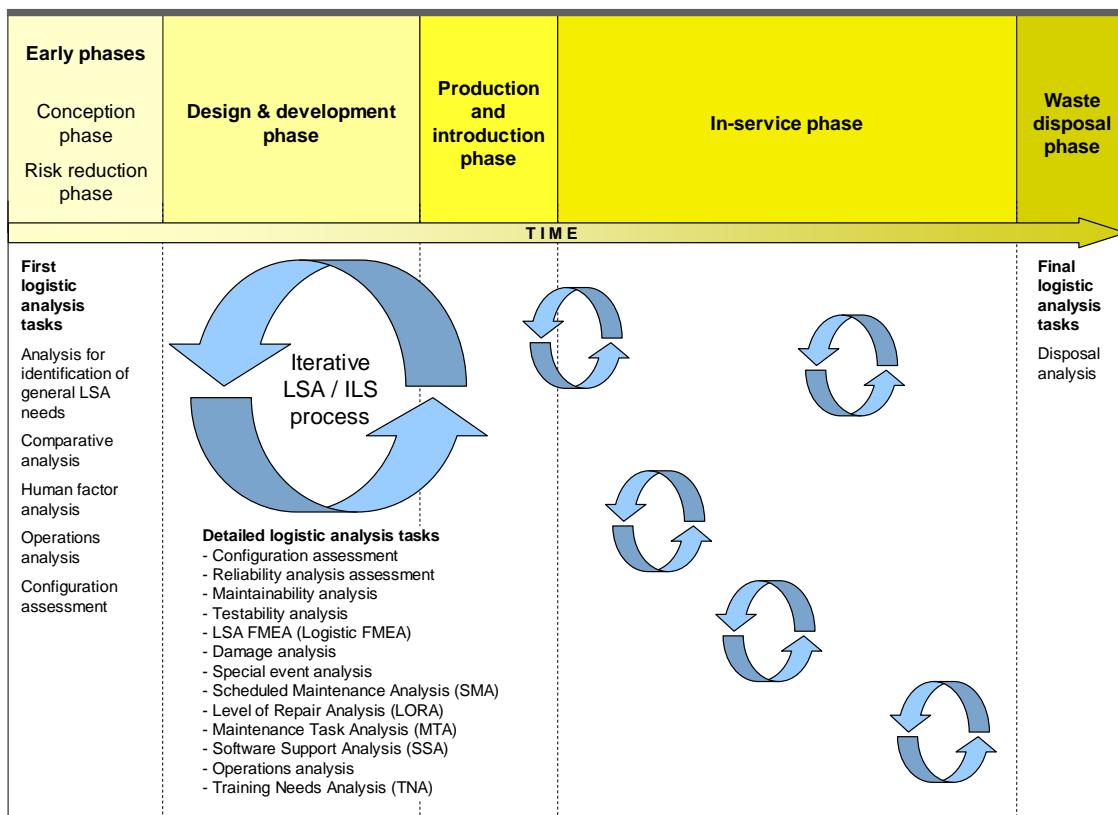
The creation of the rating method should only be done by very experienced personnel in order to ensure that the result of the recommendation sheet for analysis activities selection is reliable and sensible.

6.6

Analysis workflow processes

In the different project phases, different sets of information are available for the execution of the logistic analysis activities. Some of the tasks described can typically be addressed during an early phase of the project. Some of the tasks can be executed at a stage in which detailed and final information concerning the design are available (eg, an extensive SMA can be carried out

during a late stage of design, because a lot of detailed information is required for this type of analysis). Refer to Fig 8.



ICN-B6865-S3000L0020-002-01

Fig 8 Position of LSA in the overall project schedule

Logistic analysis requirements do not end at the end of a design & development phase. Within the next phase (production and introduction), logistic analysis can be required multiple times, depending on the modifications being made to the Product.

This is especially relevant for the in-service phase, which is normally the longest phase. There will be extensive modifications during the in-service phase and, depending on the modification, logistic analysis will be necessary again in corresponding depth and so the iterative LSA/ILS process will reappear repeatedly. Therefore, it is recommended that the documentation of logistic data and logistic decisions is continued throughout the entire project, ending with the disposal phase. Even for this final phase, logistic aspects must be considered.

6.6.1 Early phase analysis activities

In the early phase of a project, the logistic analysis activities can be limited to the ones that only require a low level of information. Before the LSA GC, information about the operational and customer requirements is normally made available and drafts for basic rules of the LSA process are prepared.

Detailed information of the SUA or the IUA is not normally available at this stage. Only basic conditions and rules for the application of the process should be available in the early stages, nevertheless, some analysis activities can, or even should, start in the early stages of the LSA process:

- Analysis for identification of general LSA needs
 - Comparative analysis

- Human factor analysis
- Operations analysis
- Configuration assessment

6.6.2 Design freeze analysis activities

During the Design & Development phase, the LSA process should accompany the design process. This applies to almost all LSA activities with the exception of those activities that should be performed in a very early phase or after a design freeze. The activities that should accompany the design & development phase are:

- Configuration assessment
- Reliability analysis assessment
- Maintainability analysis
- Testability analysis
- FMECA/FMEA
- Damage analysis
- Special event analysis
- SMA (S4000P, or RCM)
- LORA
- MTA
- Software and data uploading/downloading/transportation analysis
- SSA (software design and software maintenance)
- Operations analysis

After design freeze, the collection of logistic relevant data should be mostly complete. Therefore, analysis activities that need a lot of detailed data should be performed at a late stage of the design process in order to guarantee a fixed design and to reduce the risk of repeating costly analysis activities. This applies for:

- Simulation operational scenarios
- TNA

6.6.3 In-service phase analysis activities

During the in-service phase, the logistic analysis activities will repeat, but to a lesser extent. If the Product has technical upgrades or design changes, a new analysis concerning logistic requirements should be performed. The depth and extent of this analysis will depend on the type of system and the scale of change.

6.6.4 Summary of activities over the life cycle phases

[Table 14](#) shows a summary of the potential analysis activities over the entire life cycle phases (eg, data and information collection, management, technical/logistic analysis and preparation tasks). In this table, the Design & Development phase additionally contains typical phase review gates like a Preliminary Design Review (PDR) and a Critical Design Review (CDR).

Table 14 Summary of LSA activities during the Life Cycle Phases

Early phases	Design & Development Phase			Production Intro-duction phase	In-Service Phase	Waste Disposal phase
	Feasibility study	Preliminary design (PDR)	Critical design (CDR)			
Performance Product data (CRD)	Performance Product data updates (CRD)					
Product usage data (ORD)	Product usage data updates (ORD)					
	M-Studies	M-Analysis	M-Analysis updates		Ongoing optimization of support concept based on feedback data ⇒ In service LSA	
	R-Studies	R-Analysis	R-Analysis updates			
Comparative studies	Comparative studies	Comparative studies updates				
		Planning of development of ILS products	Development of ILS products	Update of ILS products	Update of analysis results and ILS products based on configuration changes	
					Planning of disposal	

7 Customer involvement

In order to ensure that the industry interpretations and approaches to the LSA business process are in-line with the customer's, the customer must be invited to be involved to an appropriate depth during the whole LSA program. The customer involvement with regards to the selection of LSA candidates, relevant analysis activities and resulting analysis data must be covered as well as the principles of information exchange, documentation of results and related management aspects.

7.1 Customer assessment of candidate items and recommended analysis activities

The selection of LSA candidates in conjunction with the allocation of appropriate analysis activities must be considered as the most cost influencing for an LSA process. For that reason, these selections must be carried out and assessed with great care in an early project stage. In order to reduce risk, an initial selection should be carried out prior to the signing of the contract

In general, the contractor must select the LSA candidates and related LSA relevant analysis activities based on a Product breakdown as detailed in [Para 5](#) for LSA candidate selection and [Para 6](#) for recommended analysis activities. The result must be released to the customer for assessment.

Note

Since a list that only reflects the proposed LSA CIL would not provide a clear picture of what it is really intended, it is recommended that the assessment of both the proposed Candidate Items (CI) and the related analysis activities be carried out in order to find a well-balanced decision.

7.1.1 Establishing LSA assessment rules

The assessment of proposals must be carried out with the customer's agreement, and some general rules concerning principles for assessment should be established, as a minimum, to include but not be limited to:

- Only aspects that concern rules agreed to in the LSA GC or rules that can be agreed to between the customer and the contractor within the subsequent LSA process (eg, in LSA reviews), must be considered as assessment relevant
- If an assessment has been performed and has reached a successful final clarification, it must not be reassessed unless new assessment relevant aspects occur
- Only rules that have been established within the area of the LSA community must be considered as assessment relevant
- Other communities (eg, technical documentation, materiel support, and training) are not allowed to establish or define subjects with relevance for the LSA assessment
- A change in the customer's staff is not to be considered as assessment relevant and must not be accepted by the contractor as a reason for any reassessment
- Rules concerning time limits between the release of LSA deliveries and an adequate response should be established and agreed upon, along with the consequences of non-fulfillment
- Both the invitation for assessment of details within the contractor's LSA deliverables as well as the assessment result should be indicated using applicable codes within a status code
- The structure, details of information and individual codes to be used are the responsibility of the contractor but must be coordinated with the customer
- The LSA review structure should be organized in line with the information group represented by the status code
- An appropriate data element within the logistic database should be established for status code. Refer to [Para 8.4](#).
- The status code concerning each LSA candidate must be kept updated as applicable in the logistic database

7.1.2 Initial assessment and re-assessments

7.1.2.1 Initial assessment

During the initial assessment, primarily considerations must be given to principles such as:

- Is the Product breakdown considered to be in line with the rules established during the LSA GC, eg especially concerning its structure, content and depth?
- Are rules for selection of LSA candidates and for non-selection or non-recommendation available and sufficient?
- Is the selection of LSA candidates and relevant analysis activities selection in line with rules established within the LSA GC?
- Assessment results must be documented and passed to the contractor according to the established assessment procedure and followed until a final clarification status is reached
- Once assessed and commented upon and final clarification is reached between the customer and the contractor, those principles must not re-assessed or re-discussed for later assessments

7.1.2.2 Re-assessments

Assessments subsequent to the initial assessment must concentrate on aspects such as:

- Items released by the contractor designated for assessment

- Assessment of contractor deliveries concerning general questions or comments (new aspects)
- Assessment of contractor deliveries concerning detailed questions or comments (observations)
- Assessment of contractor deliveries concerning details such as like discrepancies with prior agreements
- Assessment results must be documented and passed to the contractor according to the established detailed assessment procedure and followed until a final clarification status is reached

7.1.3

Establishing an assessment procedure

A detailed assessment procedure (including an appropriate commenting process) must be established that is tailored to the specific requirements of the individual LSA program and comprises the above elements. This procedure should be harmonized between the customer and contractor.

7.2

Information exchange between customer and contractor

The customer should inform the contractor about the individual assessment results that are relevant, in order to proceed with the LSA process, mainly on agreements/disagreements together with the reasons for any disagreement. In addition, general comments covering global aspects and/or general questions could occur. Those comments and questions should be kept separate from the other comments as they usually need a special response such as a dedicated explanation or request for special training.

This is not limited to LSA candidates and related analysis activities, but it also applies to detailed results of the analysis and to a response on any official LSA report.

7.2.1

Commenting process

A structured commenting process must be established in order to ease the management of comments and to enable traceability.

- Any LSA delivery from the contractor to the customer released for assessment must be registered and officially commented upon
- Any customer agreement must be documented carefully
- Any customer disagreement must be documented along with the related reasons. Affected items must be monitored until final clarification has been reached and the clarification is documented.

In order to address these subjects, the commenting process must cover the registration of LSA deliverables, comments and LSA status.

7.2.1.1

Registration of LSA deliverables

Any LSA delivery that needs assessment must be registered according to rules and files that have been agreed to and established as part of the LSA/ILS management structure.

7.2.1.2

Registration of comments

For each LSA delivery released for assessment, the customer response must be documented. For that reason, a status must be registered for each applicable BE depending on the consequences of the customer response.

7.2.1.3

Registration of the LSA status

The LSA status code should be structured in order to belong to different information requirements per released BE (refer to [Para 8.4](#)).

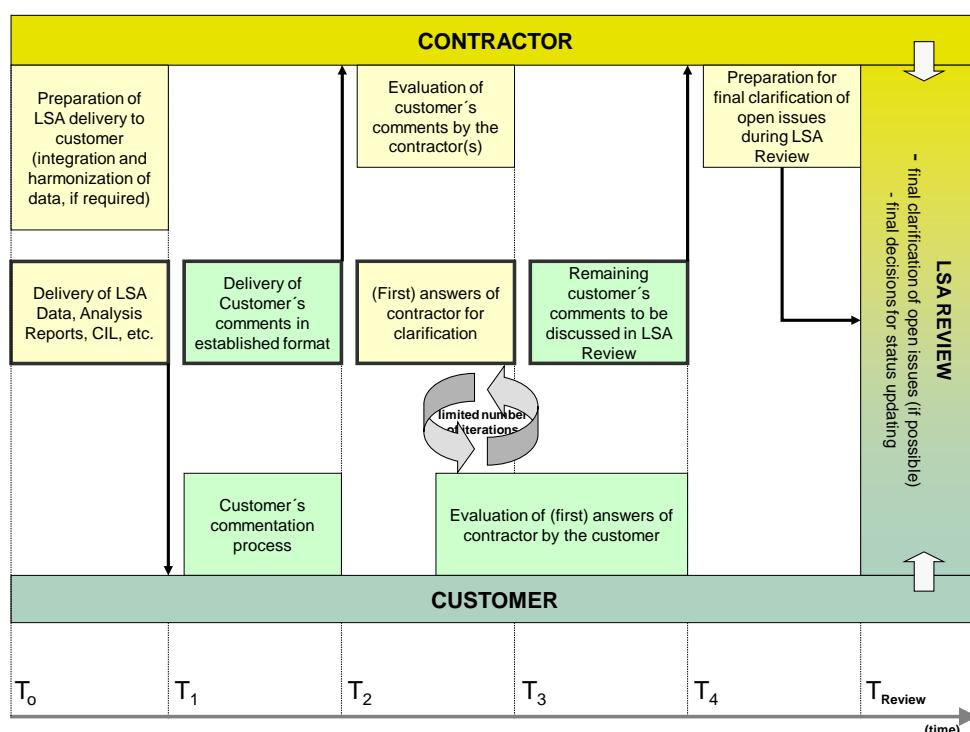
7.2.2

Informing the contractor by the customer

The customer informs the contractor as agreed in the established commenting process about:

- Agreements concerning released LSA deliverables:
This will be documented by the contractor according to established management rules. In the case of a consortium, all companies involved in the contractor network must be informed according to related rules.
- Disagreements concerning released LSA deliverables:
In these cases, the response will be investigated by the contractor (or the involved company) in order to identify an adequate response (eg, further explanation, proposed rework by industry). The contractor's response will be forwarded to the customer in order to reach an agreement in an iterative manner.
- When issues remain open and cannot be resolved by iterative commenting, an adequate solution must be identified in order to reach a final clarification.

[Fig 9](#) shows an example of the process for information exchange between the customer and the contractor. [Table 15](#) gives an explanation of the time intervals.



ICN-B6865-S3000L0021-002-01

Fig 9 Process of information exchange between customer and contractor (example)

Table 15 Explanation of time schedule of the commenting process

Time	Actions
T ₀	The start of the preparation of the LSA delivery items by the contractor. This preparation can include the integration and harmonization of data between one or more contractor in case of a contractor consortium.
T ₁	Delivery of the contractual LSA delivery items to the customer in the agreed format.
T ₂	Between T ₂ and T ₁ the commenting of the customer on the delivered items (eg, LSA data, LSA reports) will take place. At the agreed point of time T ₂ the delivery of the comments to the contractor will take place.

Time	Actions
T ₃	To reduce the effort of the LSA review, any type of comments that can be answered easily by the contractor should be handled before the LSA review, if possible. Nevertheless, any decision concerning the customer's comments must be documented carefully. During this time, more than one question-answer loop can clarify special questions in detail, however, the number of loops should be limited since more extensive questions must be clarified (eg, at the LSA review or by a longer clarification process between the contractor and customer).
T ₄	Taking into account the clarification loops between the customer and the contractor in the time between T ₂ and T ₄ , the remaining customer comments will be delivered to the contractor. These comments will be the basis for any discussions and clarifications in the LSA review.
T _{Review}	Time of LSA review. It must be ensured that every decision in the LSA review is documented in an update of corresponding status information.

7.3 Final clarification on open issues

For any remaining open issues, an adequate solution must be identified. If this is not possible during a regular review session, a meeting of dedicated specialists could be required to reach a satisfactory solution.

7.4 Customer decision (status influencing)

Normally, distribution of an LSA delivery for assessment to different recipients (eg, different authorities in different nations and, since LSA is to be understood as inter-disciplinary, to all involved logistics communities) is required. Subsequently, individual comments must be collected and (in case of deviating responses) be harmonized in order to provide one official customer response.

Note

It is important for the contractor to receive only one official customer response in order to proceed with the LSA process in a manageable way. It is recommended to receive the customer's decisions as a status code change (following the rules established for status code allocation).

The status code allocated by the customer remains unchanged in the related customer release. Any related explanation (eg, reason for disagreement) also must be documented as part of the contractor response and remain untouched for traceability reasons.

For LSA program traceability, both, the LSA delivered as proposed for assessment, as well as the customer's response (if applicable including final clarification results), can be documented as:

- LSA recommendation document delivered by the contractor
- LSA decision document responded to by the customer

Particularly in the case of essential documents such as LORA or SMA results, the resulting recommendation/decision documents could ease some clarification requirements.

Within the contractor's LSA database, corresponding status codes concerning the involved BEs must be updated accordingly. However, contractor's information can supersede the status code as allocated by the customer (eg, on-going work status, required deletions).

7.5 Exchange of analysis data with the customer

The steps that must be applied include but are not limited to:

- Establishment of appropriate rules for data and document exchange. Refer to [Para 7.5.1](#)
- LSA data and document collection by the contractor. Refer to [Para 7.5.2](#)
- Delivery of LSA data/documents by the contractor. Refer to [Para 7.5.3](#)
- Customer assessment and distribution of assessment results. Refer to [Para 7.5.4](#)
- Distribution of customer response by the contractor. Refer to [Para 7.5.5](#)
- Distribution of the contractor response concerning disagreed comments. Refer to [Para 7.5.6](#)

7.5.1 Establishment of appropriate rules for data and document exchange

Appropriate agreements and rules must be documented and agreed to, by the customer and the contractor in the LSA GC and the subsequent LSA agreements. For example:

- Software to be used
- Data and report formats (eg, DEX-format)
- Periodicity
- Other reasons for delivery
- Distribution partners and sequences to be obliged

7.5.2 LSA data and document collection by the contractor

Appropriate agreements must be documented and agreed to, by the customer and the contractor in the LSA GC and the subsequent LSA agreements. This includes but is not limited to:

- Establishment of an appropriate data/document exchange and file network within industry
- Collection of data/documents for intended delivery between associated companies and LSA related disciplines
- Performance of industry internal quality checks, harmonization and agreement for delivery

7.5.3 Delivery of LSA data/documents by the contractor

Appropriate agreements must be documented and agreed to, by the customer and the contractor in the LSA GC and the subsequent LSA agreements. This includes but is not limited to:

- Distribution of the LSA deliverable (directly or via a designated management unit) by taking into account the agreed distribution rules including the due date, if applicable
- Contractor internal documentation and distribution of the official LSA delivery

7.5.4 Customer assessment and distribution of assessment results

Appropriate agreements must be documented and agreed to, by the customer and the contractor in the LSA GC and the subsequent LSA agreements. This includes but is not limited to:

- Distribution of the contractor's LSA delivery, as required
- Harmonization of the individual responses to the official customer response
- Distribution of the customer response to the contractor

7.5.5 Distribution of customer response by the contractor

Appropriate agreements must be documented and agreed to, by the customer and the contractor in the LSA GC and the subsequent LSA agreements. This includes but is not limited to:

- Registration of the official customer response
- Industry internal distribution
- Identify, distribute and harmonize the contractor investigation results

- Update of the LSA database according to the customer response details (as much as possible)
- Preparation of harmonized contractor response to general comments and disagreements

7.5.6 Distribution of the contractor response concerning disagreed comments

Steps concerning iterated analysis data from the contractor as described in [Para 7.5.3](#).

8 LSA review conference

A LSA Review Conference (LSA RC) should be planned with the participation of contractor and customer personnel in order to:

- reach final clarification of open issues
- reach customer acceptance on open issues
- monitor the status of LSA candidate items, eg concerning completion and quality
- assess logistic support aspects related to each phase of the process
- reach additional agreements between the customer and the contractor concerning LSA aspects in order to improve the LSA process, as required

8.1 LSA review process overview

The LSA RC must be conducted at LSA candidate level. Prior to the conference, the contractor must inform the customer about the LSA candidates to be discussed.

During the LSA RC, the related LSA deliveries and the relevant data elements must be assessed and decided upon by taking into account established acceptance criteria. Any decisions must be documented.

Complex LSA CI could require a sequence of analysis activities, which can be distributed over a long period of time, depending on the type of information that is available within different project stages. On the other hand, some decisions must be made in a very early project stage based on limited information (eg, establishment of an preliminary maintenance concept) while other decisions require a stable design that is not typically available until a very late stage (eg, identification of scheduled maintenance tasks based on time consuming, excessive analysis such as S4000P). Therefore, different review steps must be established in order to enable sequenced decisions in line with the structured analysis process. The overall review process must also be divided into different review steps, each supported by coherent information. These review steps must be identified by review step indicators (refer to [Para 8.3](#)). Those data elements that must be available for, or that must be assessed during, the individual review steps must be defined. Each LSA RC should cover any steps that belong to the individual LSA candidates being requested for assessment which will be indicated by according status codes.

Due to the iterative nature of any LSA process, any approval given during an LSA RC must be considered as a temporary assessment but it must allow the contractor to continue with the LSA process. As a consequence of the iterative nature of LSA, the data can be re-examined due to changes in the design, the updating of technical data and/or the support or operational environment.

The final results must be documented in the LSA documents and/or within the LSA database, eg at the end of each project phase, must be considered as frozen. Related acceptance criteria must be agreed to during the LSA GC.

The LSA RC should be held in accordance with the evolving design and the progress of the intended analysis activities. These meetings should take place regularly within an agreed period in time or depending on the amount of information to be assessed. Minutes of each LSA RC must be prepared to record the results and to give evidence of the necessary actions for the customer and the contractor. Updating related LSA documents and/or the LSA database in accordance with the results of an LSA RC must be considered a required iterative action.

8.2

Subject for review

In general, any contractual item agreed to during an LSA GC, or by subsequent agreements between the customer and the contractor must be reviewed during an LSA RC. In addition, any aspect that can be identified as essential for the LSA process or that could influence other logistic processes with interfaces to the established LSA program can be discussed during an LSA RC in order to agree on modified or additional LSA activities. Occurrences within the LSA process that require common assessment in order to correct any essential problem should be coordinated between the customer and the contractor.

Introductions into procedures and/or principles of LSA relevant analysis required should be presented by the contractor to the customer in order to reach a common understanding concerning the goal of analysis activities, related restrictions and essential LSA results including the introduction into global analysis work flowcharts. Refer to [Fig 3](#) and [Fig 4](#).

Progress reports and data quality assessment summaries should also be provided by the contractor, as well as summary reports reflecting items of special interest for the customer. Some examples include:

- Maintainability values such as mean man-hours per operating hour
- Logistic performance parameter such as Mean Elapsed Time (MET) for maintenance tasks that occurred within specified limits, along with the related percentage

8.3

Examples for review structuring

A typical review process could be structured as a series of steps. Different aspects and/or types of information at different stages of the overall analysis process can be covered in this way. The steps can be arranged as required by a project to realize LSA data release and acceptance within the LSA review process:

- LSA review step - CIL and maintenance analysis allocation. Refer to [Para 8.3.1](#)
- LSA review step - LORA results and maintenance concept information. Refer to [Para 8.3.2](#)
- LSA review step - FMEA and SMA results. Refer to [Para 8.3.3](#)
- LSA review step - Maintenance Task Analysis (MTA). Refer to [Para 8.3.4](#)

8.3.1

LSA review step - CIL and maintenance analysis allocation

This step forms the basis for all subsequent LSA activities documented within the LSA database and consists of:

- BEI and LSA CI selection including grouping candidates, where applicable
- Identification of the type of candidate and related attributes, such as:
 - Identification of LRUs
 - Potential cost drivers
 - Potential maintenance drivers
 - Potential readiness drivers
- Allocation of LSA relevant analysis activities to be performed for each selected candidate
- Status code reflecting details
- Any updates regarding changes in the design configuration at the CI level must be managed in the LSA database

8.3.2

LSA review step - LORA results and maintenance concept information

In this step the preliminary maintenance concept will be prepared by allocating maintenance tasks to their related maintenance levels and performance sites. This task typically includes:

- Maintenance concept recommendation proposed by the contractor.
- Identification of maintenance tasks relevant for equipment integration into the Product, such as:

- Gain and undo access
- Remove and install equipment
- Performance of functional tests at equipment and system level, as required
- Substantiation concerning the proposed maintenance concept as required (eg, LORA results, applied method or procedure, main influencing aspects to be considered, other analysis being applied or assessed for LORA constitution)
- Customer decision on the maintenance concept which could be rejection or alternative solution

8.3.3 LSA review step - FMEA and SMA results

In this step, the maintenance concept is be finalized by taking into account the FMEA (eg, constituted from Equipment FMECA by the grouping of failures focused on the intended maintenance tasks) for the identification of unscheduled maintenance tasks. Results from SMA will identify scheduled maintenance tasks.

8.3.4 LSA review step - Maintenance Task Analysis (MTA)

In this step, the identified maintenance tasks will be analyzed concerning its requirements and its execution:

- Description of the required maintenance tasks in applicable detail and depth
- Duration of each task step
- Identification of related logistic resources required to support the maintenance activities (eg, personnel, support equipment, spare parts, consumables, facilities, data and software)

8.4 Status code examples

In order to reflect the status of each LSA candidate in detail concerning the identified review steps, a status code should be used, which can consist of sub codes reflecting:

- The responsibility of the companies involved. For each task group, the responsible company should be identified. This also reflects the agreed work share details concerning this subject.
- The type of LSA candidate (eg, full, partial)
- Important aspects (eg, contains user loadable SW and/or data)
- Review step indicator
Here, the analysis status of the addressed LSA candidates must be indicated concerning the content of each review step (eg, as described in [Para 8.3](#))

[Table 16](#) gives examples for potential codes and the corresponding meaning (for each LSA candidate review step indicator):

Table 16 Examples for status codes

Code	Description
X	Review step content is in "working condition", assessment not requested
N	Review step is "not applicable" for the related LSA candidate
R	Review step content is requested for assessment and decision in the next LSA RC
A	Review step content is (temporarily) agreed to by the customer
B	Review step content was assessed and in principle (temporarily) agreed to by the customer. However, minor rework is expected prior to final acceptance. In terms of contractual status, a "B" can be treated like an "A".

Code	Description
O	Review step content was not agreed to by the customer and remains an open issue

8.5

Depth of status code allocation

As a minimum requirement for each LSA candidate, relevant status codes should be allocated. In addition, the subject of assessment can be addressed in more detail (eg, the specific task of the LSA candidate that was established/modified and needs to be assessed). This decision must be made during the LSA GC.

8.6

Functions of the status code

The status code reflects the review process structure and address all contractual issues that must be documented within the LSA database. It can be used for overall status summary reports concerning the content of the LSA database, eg, after each LSA RC to reflect the progress in the area of LSA.

An LSA CIL filtered by status code can be used to focus the customer's attention to this specific subject.

The status code could also be used internally by the contractor in order to indicate to logistic disciplines the "stop" or "go", status to indicate any risk when creating ILS products.

8.7

Status code definition at the LSA guidance conference

The intended review steps, as well as the related status code structure and applicable status codes, must be proposed by the contractor and agreed to by the customer during the LSA GC along with the depth and level of detail of status code allocation.

9

Starting point and management of the creation of the logistic products

9.1

Starting point recommendations

The starting point for the creation of the logistic products is dependent on several factors, including technical documentation and illustrated parts data availability. Changes of design or maintenance concepts can have also extensive influence on these products. In this context, ILS products to be considered include:

- Technical documentation
- Materiel support (illustrated parts data)
- General and special support equipment
- Training
- The facility construction is not considered a typical ILS product since the decisions for facilities are made in an early phase of a project, because of the long qualifying time

9.1.1

Technical documentation

In general, technical documentation consists of two main parts:

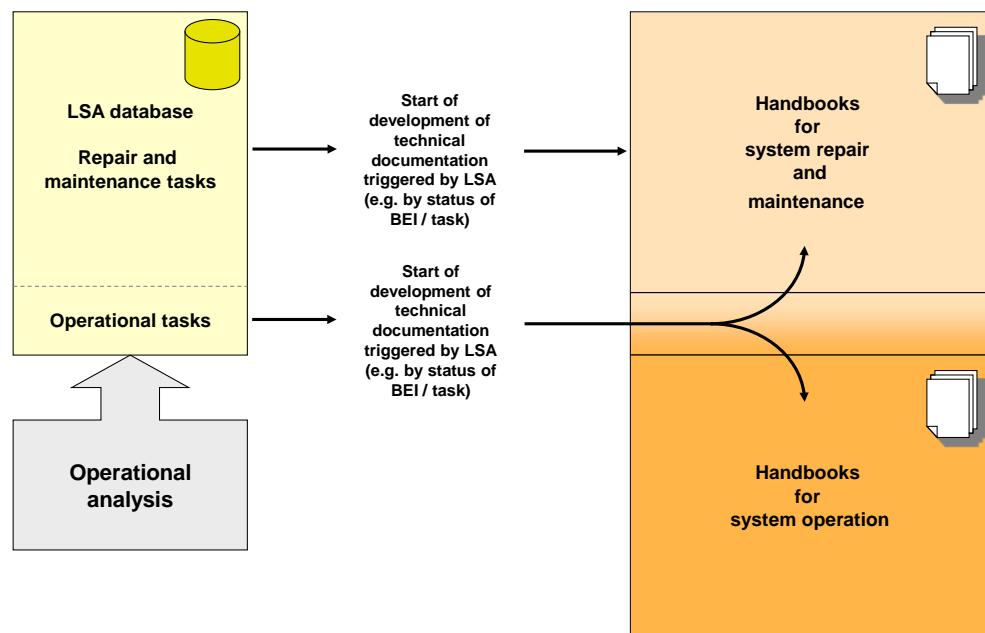
- Handbooks for system repair and maintenance
- Handbooks for system operation (user documentation)

During a design and development process, the supporting logistic analysis activities are performed. Depending on the progress of these analysis activities, which maintenance concept will be the basis, which personnel, support equipment and spares are required and how the task will be executed (task description), become more well defined. Therefore, the development of technical documentation concerning system repair and maintenance should commence at a

later point. The better the information from the logistic analysis activities is, the less the risk for inaccurate technical documentation, which could lead to rework.

In some cases however the development of technical documentation must start early because of time limitations and contractual preconditions. The basis for the development of technical documentation detailing system repair and maintenance is the maintenance task, which is documented in the LSA database. It is recommended that a robust solution to trigger the technical documentation development to support the Product be implemented. This process must be harmonized between the contractor's support engineering and the technical documentation departments. A good solution is to use status information for LSA candidates, or even for maintenance tasks, within the LSA database and mechanisms for drafting technical documentation described in S1000D.

The correlation between LSA and technical documentation is shown in [Fig 10](#).



ICN-B6865-S3000L0022-003-01

Fig 10 Correlation between LSA and technical documentation

Generally, the development of the technical documentation for the operation of the Product is an independent activity from the LSA process. However, it is recommended that this documentation (eg, a prototype of a technical system safety for testing and qualification) be made available as early as possible. A particular aspect is the consideration of information from the operational analysis activities. The tasks described and documented in the LSA database in this area can be part of the technical documentation for both Product repair and maintenance, and Product operation. In this case, the recommendations for the repair and maintenance documentation are also valid for the results of operational analysis.

Each repair or maintenance task, identified and documented in the LSA database, must be reflected by the technical documentation for Product repair and maintenance.

However, the technical documentation for system repair and maintenance can contain information beyond the documented LSA tasks because sometimes it is not possible to include each single maintenance action within the LSA database (eg, because of budgetary reasons). Not every piece part will be an LSA candidate. The purpose of LSA should be to analyze, in depth, the true maintenance and cost drivers. The remaining equipment or piece parts that require technical documentation must also be considered. The depth and the extent of this

additional technical documentation should be clarified between the contractor and the customer during the GC.

To guarantee a meaningful information flow between LSA and technical documentation, it is recommended to establish a process to exchange the status of work of LSA to the technical documentation department on a regular basis. In this way, technical documentation can keep pace with all relevant information needed to start the development of the technical documentation. Aspects that should be covered, include but are not limited to:

- Is the relevant maintenance task ready for technical documentation?
- If the status of the LSA task indicates: "Not ready for technical documentation", what risk could be taken to start the production of technical documentation anyway?
- Is there a blocking status in LSA, which permits the start the production of technical documentation?
- Which additional activities must be documented within technical documentation, which are not a part of the LSA database (no trigger from LSA for those aspects)?

9.1.2 Materiel support

The identification of relevant spare parts is a main task within the LSA process. The depth of the breakdown and the extent of analysis activities will be reflected in the identification of spare parts. Depending on the maintenance concept, identification of spare parts can differ drastically. For example, in a simple 2-level maintenance scenario, complete components will be replaced and no repair activities will be performed by the operator of the system. In this case, only a small number of spare parts will be identified. However, additional spare parts, such as standard parts, could be identified that are additionally required by the identified complete components.

The decisive question is the depth of the breakdown within the LSA. Theoretically, a breakdown down to the last piece part is possible, but the effort would be huge and therefore unacceptable. Therefore, in practice, the identification of spare parts within the LSA process will stop at a certain level. Typically, the maintenance drivers and the cost drivers should be identified within the LSA process. This information is documented within the requirements of maintenance tasks in the LSA database. A similar process as described for technical documentation should also be established for materiel support. Depending on the progress of the logistic analysis activities, the start of the creation of the logistic products concerning materiel support should be triggered by LSA relevant maintenance drivers and cost drivers.

The same applies to technical documentation in that, in many cases, not all required spare parts will be identified by LSA. However, all spare parts identified by an LSA task must be considered within materiel support. [Table 17](#) gives an overview of the different types of spares and their consideration in the identification process.

Table 17 List of different spare part types identified by LSA

Spare part type	Description	Correlation between LSA and materiel support
Complete piece of equipment	<ul style="list-style-type: none"> - Maintenance driver - Cost driver <p>Required spare parts for replace tasks (scheduled or unscheduled) at operator site (replacement of complete equipment without any repair at operator site)</p>	Identification of the equipment as a required spare part should be approved by a maintenance task defined within the LSA.
Dedicated spare parts	Required spare parts for equipment repair or maintenance tasks at operator site (repair of the equipment will be performed at operator site)	Identification of this component as a required spare part should be approved by a maintenance task defined within the LSA.

Spare part type	Description	Correlation between LSA and materiel support
Standard parts	Required standard parts (eg, attaching parts or seals) for repair or maintenance tasks at operator site (repair of the equipment will be performed at operator site)	Identification of this component as a required spare part should be approved by materiel support because of missing depth of breakdown within the LSA.
Consumables	Required consumables such as liquids, grease, special chemical products, source material, glue.	Identification of this component as a required spare part can be approved by materiel support or by LSA.

The process to begin the preparation of illustrated parts data within materiel support is shown in [Fig 11](#):

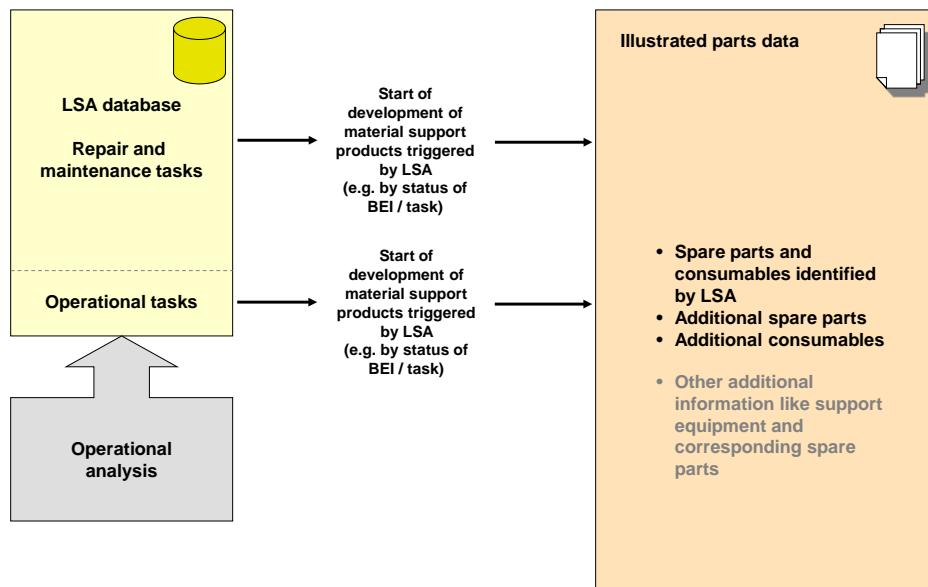


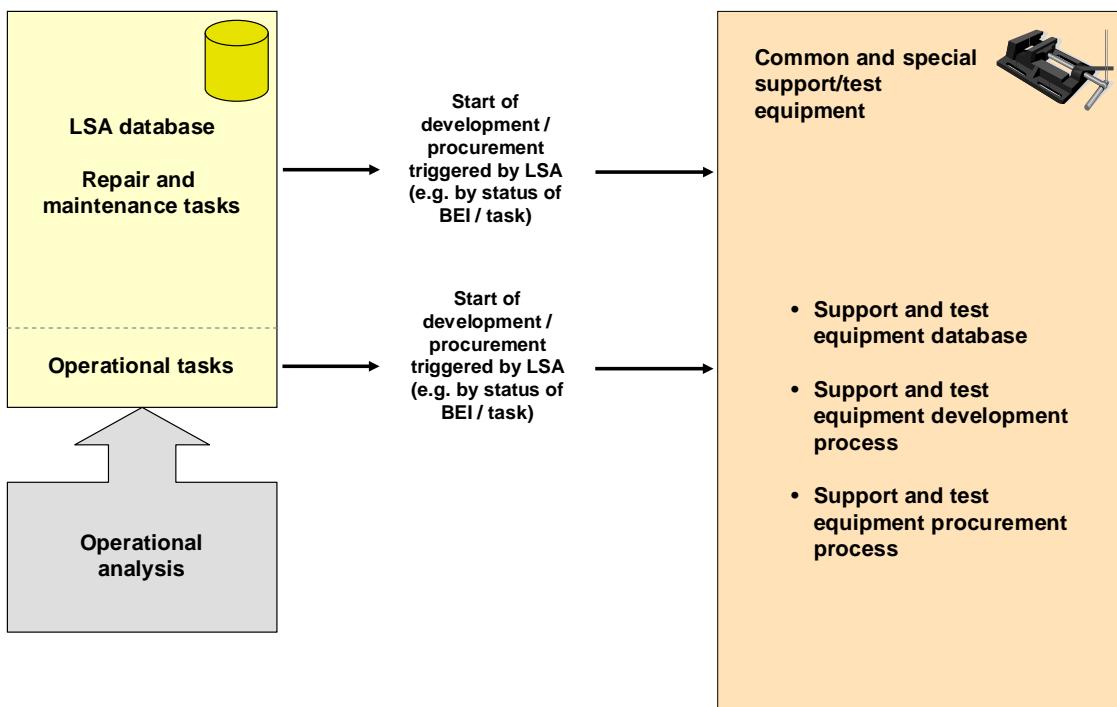
Fig 11. Correlation between LSA and materiel support

9.1.3

Common and special support/test equipment

The identification of relevant support and test equipment is a main task within the LSA process. The timely provisioning with required support/test equipment is of crucial importance and it should be distinguished between common and special support/test equipment. Particularly, the requirements for special support/test equipment should be identified by the LSA process. After the identification, a development or procuring process should begin. Refer to Fig 12.

Each item of support and test equipment that is identified by the LSA as a requirement must be a part of a development or procurement process. The LSA status information can be used to initiate activities within the department responsible for support and test equipment. However, in some cases, the requirement for support/test equipment can also be derived from sources other than the documented LSA tasks typically for common tools, (eg, a simple screwdriver does not need to be identified by a special LSA task).



ICN-B6865-S3000L0024-001-01

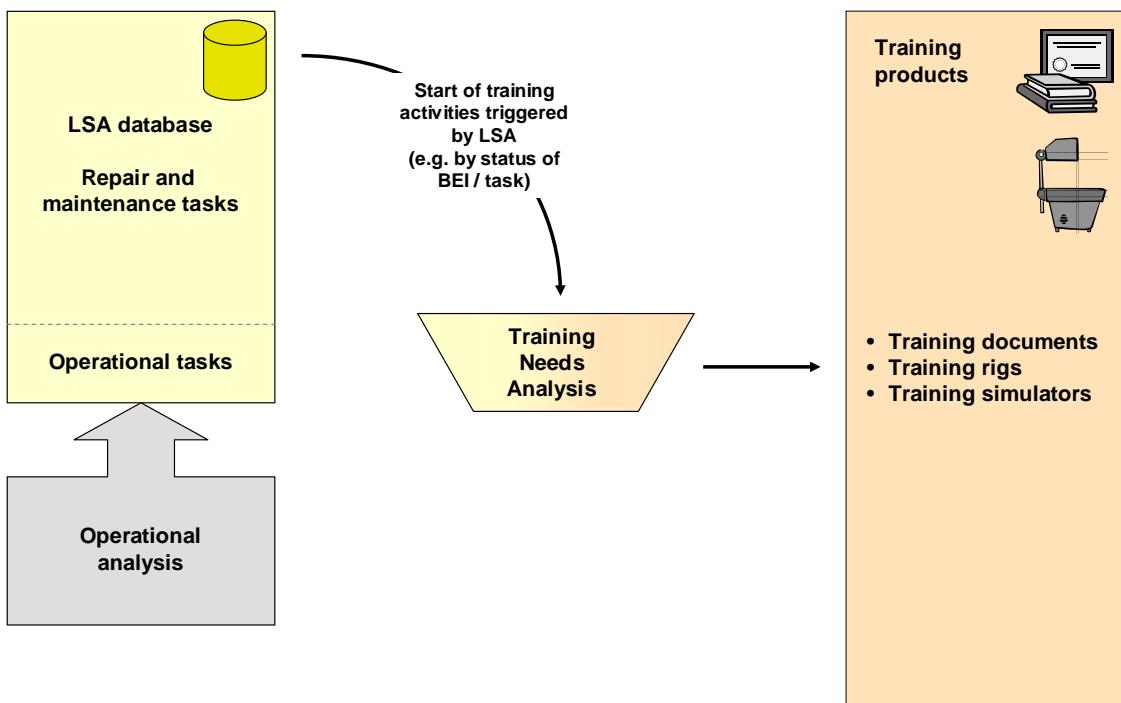
Fig 12 Correlation between LSA and support/test equipment

9.1.4 Training

Particular attention must be applied to training. Training needs concerning maintenance activities cannot be identified until complete information concerning the required maintenance tasks is available (eg, details of performance, required support equipment, difficulty and criticality, duration, number of working steps, required personnel). The first step in identifying training requirements should be a TNA. This can be based on the maintenance tasks identified by the LSA. To help identify the best starting point for training activities, status information should be introduced that can be exchanged between the TNA and the LSA. Additional information should be taken from the technical documentation in order to support the identification of training needs that are not documented in LSA, and to support the development of training content. Refer to [Fig 13](#).

The process to identify and develop training content should be agreed to by the customer taking input from LSA and/or technical documentation that is available. The development of training equipment can be very cost intensive (eg, production of training videos, training rigs, training simulators). Therefore, decisions concerning training should be based on reliable information.

For the best possible support, it is recommended that training information, that is within the LSA database, be documented. The LSA database can contain simple markers such as "training required", skill levels for personnel or more detailed information, if available. It is recommended that the LSA database be designed to include queries that support the extraction of training requirements.



ICN-B6865-S3000L0025-001-01

Fig 13 Correlation between LSA and training

9.2

Management of the development of the logistic products

One central challenge within the development or introduction of a new Product is the coordination of the product support activities. For the development of new Products, the development of the ILS products can be complex. Therefore, the management of this development process must be supported by a robust solution. It is recommended that, ILS managers be provided with the means to use LSA information and the LSA database as a central management tool for the entire ILS process. The contractor must consider:

- Timely development of the ILS products must be supported by LSA status information. Triggering the logistic disciplines should be generated by the support engineering department to ensure a proper start.
- Creating unnecessary effort in any logistic discipline must be avoided, such as:
 - Creation of technical documentation for maintenance actions that are never performed at the customer operational site.
 - Documentation or provisioning of spare parts or consumables that are never required at the customer operational site.
 - Beginning the development or procurement of support/test equipment that are never required at the customer operational site.
 - Planning training for maintenance tasks which are never performed at customer operational site.

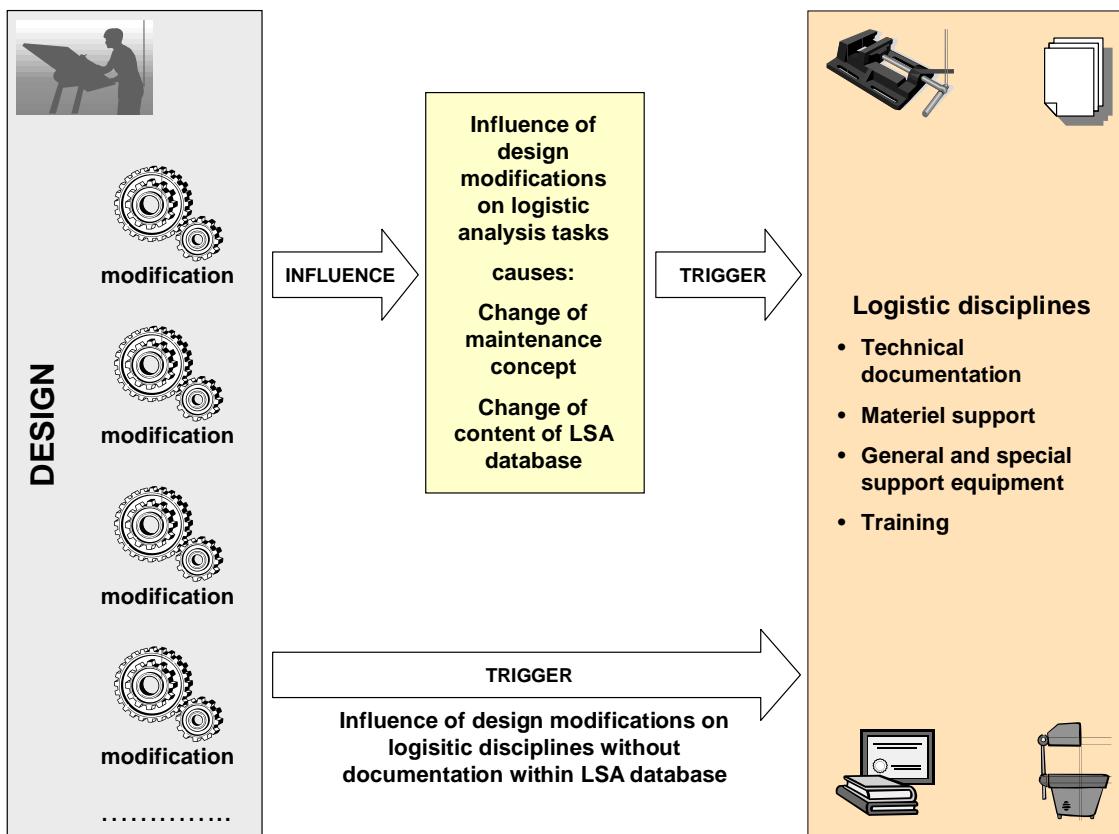
To avoid negative impacts of these aspects, it is recommended that a permanent quality check of the content of the logistic disciplines against the content of the LSA database be carried out.

9.3

Influence of modifications on logistic products

Design modifications that are relevant for any logistic analysis activity and are relevant for the ILS disciplines must be considered carefully. The commencement of analysis for any of the logistic disciplines must be triggered in case of changes in the LSA database and design changes with no connected information within the LSA database. Good management of the

entire ILS process, as a common process from the design to the logistic products, is essential. It is recommended that a strategy for managing modifications during every project phase, be established. Even in the design and development phase, many of modifications will have an effect on the entire project. Also, in later phases of the project, modification management for the logistic support products is of vital importance for both the customer and the contractor. [Fig 14](#) illustrates a typical triggering process.



ICN-B6865-S3000L0026-001-01

Fig 14 Influence of design modifications on logistic disciplines in general

10 Checklists

10.1

Detailed checklist for the creation of an ORD

To support logisticians in developing a complete ORD, a checklist with a number of detailed questions should be used (refer to [Table 18](#)). However, there will be project specific aspects that should also be taken into account.

Table 18 Checklist of detailed questions to support the ORD creation

Detailed ORD questions	Answer/Action
General usage scenario	
Which usage areas or which mission areas are expected?	Describe usage areas
Is the normal usage of the Product the only scenario?	Yes/No?
Are there special usage scenarios, which must be taken into consideration?	Yes/No? If yes, describe special usage scenarios in detail

Detailed ORD questions	Answer/Action
Is a permanent usage expected?	Yes/No? If yes, describe in detail
Is an out of area usage under rough conditions expected?	Yes/No? If yes, describe in detail
Is there any influence on the environment expected from noise from the usage or the maintenance of the Product?	Yes/No? If yes, describe in detail
Is there any influence on the environment expected by pollution from the usage or the maintenance of the Product?	Yes/No? If yes, describe in detail
Is there any other influence on the environment expected from the usage or the maintenance of the Product?	Yes/No? If yes, describe in detail
Are there any actions necessary to avoid negative influence on the environment such as noise or pollution?	Yes/No? If yes, describe in detail
Are there any problems with the currently used Product concerning performance?	Yes/No? If yes, describe problems
Are there any problems with the currently used Product concerning maintenance?	Yes/No? If yes, describe problems
Are there any problems with the currently used Product concerning supportability?	Yes/No? If yes, describe problems
Is the system used in combination with other existing Products (generally or only sometimes)?	Yes/No? If yes, describe interaction
Is the usage of the Product dependent on other existing Products?	Yes/No? If yes, describe dependence
Is the Product itself transportable?	Yes/No?
How long does it take to prepare the Product for transportation?	Duration
Does transportation require a special conservation and packaging?	Yes/No? If yes, describe in detail
Define transportability requirements (mode, type, quantity, distance, duration of transport)	Define details
Which requirements concerning support equipment and personnel exist for the preparation of the Product for transport?	Define details
What is the maximum time to reset the Product to full capability after transportation?	Define duration
Which requirements concerning support equipment and personnel exist for the rest of the Product to full capability after transportation?	Define details
What else must be transported (eg, support equipment, spares, personnel) for the proper operation of the Product at the destination and what are the requirements for these additional transportation actions.	Define details

Deployment of locations and special conditions of every location

How many operating locations exist? Number

Applicable to: All

S3000L-A-03-00-0000-00A-040A-A

Chap 3

Detailed ORD questions	Answer/Action
Is there an international or even an intercontinental distribution of the operational locations?	Yes/No?
What are the distances between the operating locations?	Map, distances
What are the distances between the operating locations and the required industry and/or contractor facilities?	Map, distances
Define the type of each location:	Define details
Is the location land based?	
Is the location ship based or sea based?	
Is the location based in a mountain region?	
Is the location a depot with maintenance abilities?	
Is the location a home base or an offshore location for the Product?	
Is the location a training base equipped with training facilities?	
Define the special conditions at each location:	Define details
Are there extreme sandy or dusty conditions?	
Is there a salty atmosphere because of location?	
Are there extreme hot or cold weather conditions?	
Are there other extreme stress conditions at special locations?	
Are there limitations due to special regulations/laws at special locations?	
Are there special storage requirements due to extreme weather conditions at special locations?	
In a wartime scenario, are there special locations that would preclude contractor maintenance? (eg, maintenance and repairs in wartime environment cannot easily be performed by contractors)	Yes/No?
Are there any special situations involving threat that might require a special emergency maintenance concept limited to a minimum of intervention?	Yes/No?
Is there sufficient infrastructure available to reach all locations (eg, to guarantee proper supply of spare parts or consumables)	Define details
Quality and existence of roads	
Airport (local or international) nearby	
Connection to railway stations	
Connection to inland or sea ports	
Are there special infrastructural requirements for reaching a location?	Yes/No? If yes, define details
What are the existing capabilities of each location concerning support equipment?	Define details
What are the existing capabilities of each location concerning infrastructure and facilities?	Define details
What are the existing capabilities of each location concerning personnel?	Define details
What are the planned capabilities of each location concerning support equipment?	Define details
What are the planned capabilities of each location concerning infrastructure and facilities?	Define details

Detailed ORD questions	Answer/Action
What are the planned capabilities of each location concerning personnel?	Define details
Will there be plans to establish a repair station at special locations?	Yes/No?
Will there be plans to establish a supply depot at special locations?	Yes/No?
Will there be plans to share abilities between different locations (exchange of support equipment or personnel)	Yes/No?
Product support and deployment	
How many supported Products are planned per location?	Define details
Deployment of Products (end items) per location	Define details
Will there be plans to share Products (end items) between different locations?	Define details
Product usage overview	
What will be the normal usage of the Product at every location (key usage)?	Define details
What is the planned availability of the Product at each operating location?	Define details
When the Product is used in missions, what are the planned mission success rates for each operating location?	Define details
Are there any periods of special usage of the Products at a special location? <ul style="list-style-type: none"> – low payload period – temporary peak loads – temporary storage of the Product – out of area usage under special or rough conditions – wartime usage 	Define details
In the case of permanent operational conditions, what are the possible down times for maintenance (maintenance windows)?	Define time periods for maintenance
Define additional Product performance parameters in measurable quantifiable terms such as: <ul style="list-style-type: none"> – range of usage – required accuracy – payload values – required temperatures – required speed – required distances 	Define detailed values
What is the key measurement base of usage per unit of time? Examples are: <p>Operational hours for general Products Flight hours for airborne Products (aircrafts, helicopters) Kilometers or miles for land based vehicles Rounds or cycles for periodic processes in a Product Tons for transportation systems</p>	Define detailed values

Detailed ORD questions	Answer/Action
What operational profile is expected at each location?	Define a operational profile as detailed as possible
How many operating hours or missions are planned per day?	
Are there different typical usage days, define the operating profile for each "day type".	
What are the typical operational profiles for longer ranges such as weeks, months or entire years?	
How many operating days/operating hours are expected for a year or for any other basic time period?	Define detailed values
What is the average duration of each different usage (eg, operation, mission, trip, flight, dive)	Define detailed values

10.2 Detailed checklist for the creation of an CRD

To support logisticians to develop a complete CRD, a checklist with a number of detailed questions should be used (refer to [Table 19](#)). However, there will be project specific aspects that should also be taken into account

Table 19 Checklist of detailed questions to support the CRD creation

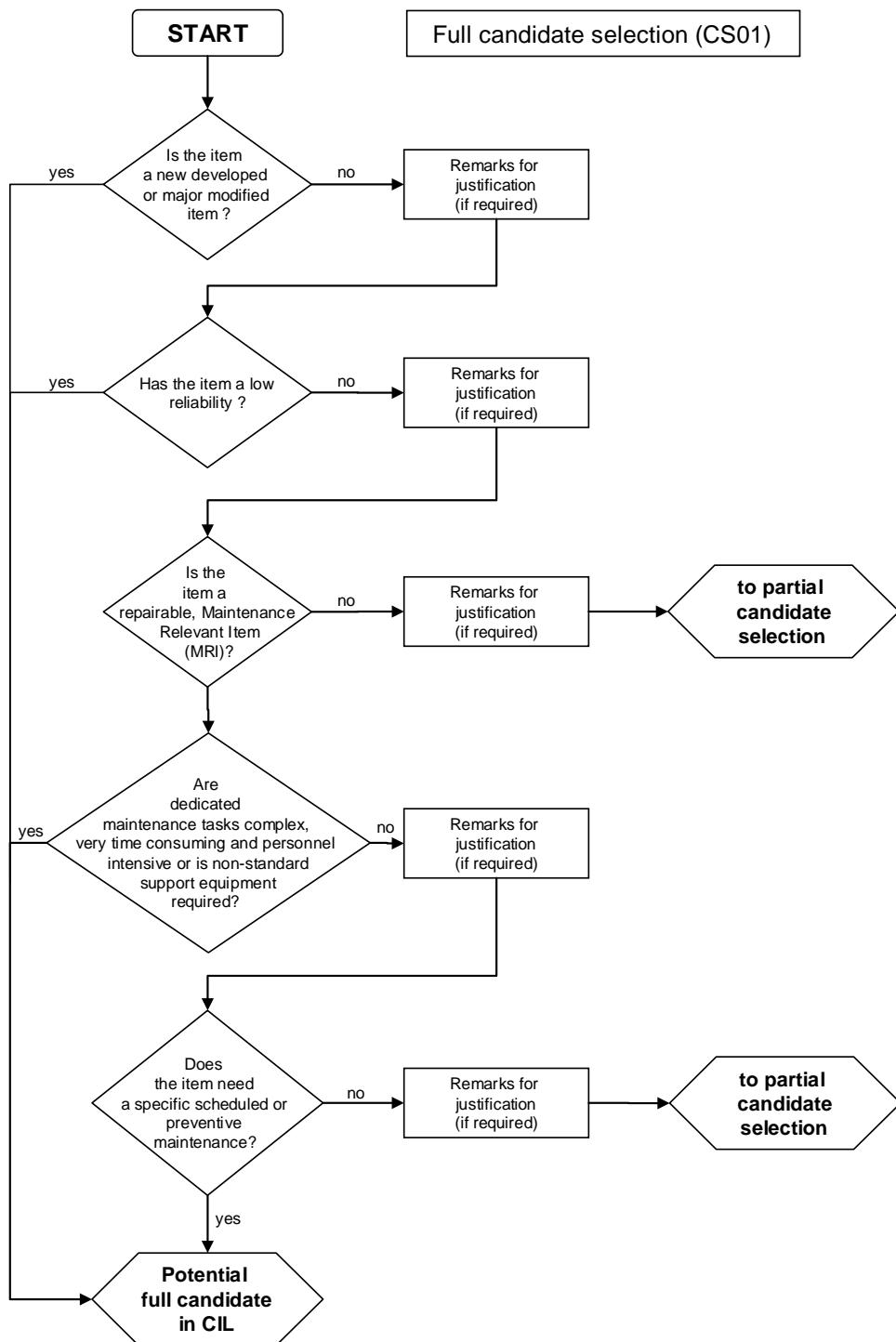
Detailed CRD questions	Answer/Action
Supply concept	
Are central depots expected, and how is the deployment planned?	Define details
Are depots planned directly at the operational locations?	Yes/No?
Is a material exchange between different locations expected?	Yes/No?
Is a selection process for suppliers defined and how is it planned to integrate suppliers into the expected supply concept?	Define details
Is it possible for a production line to provide spare part support (especially at early stages)?	Yes/No?
Is an outsourcing concept via a support agency expected?	Yes/No?
What are the limitations for spare part availability and lead times?	Define details
What distribution systems and IT-system will be used for spare part supply?	Define details
Which concept will be used for initial provisioning?	Define details
Is it expected to use an optimization or a simulation tool?	
Is the deployment schedule properly supported by the initial spares delivery?	
Will spare part delivery impact the production schedule?	Yes/No?
Will the industrial base be obligated to provide spares support in the out-years for items that remain in the inventory?	Yes/No?
Support equipment concept	
How can it be ensured that all support equipment is available in time?	Define details

Detailed CRD questions	Answer/Action
What is the identification and realization process for standard and special support equipment?	Define details
How effectively is automated test equipment being utilized to support the Product?	Define details
What is planned for maintenance and support of the support equipment itself?	Define details
Repair and overhaul of support equipment	
Calibration of support equipment	
Personnel integration	
What are the general requirements and provisions for manpower and personnel?	Define details
Are the Products operated by the staff of the customer or by staff of the contractor?	Define details
Should cooperative models be considered?	Yes/No?
How can it be ensured that the required operator training is verified on time?	Define details
How can it be ensured that the required maintenance training is verified on time?	Define details
What training processes should be developed to ensure adequate operational and maintenance support at all levels during the entire life of the system? The possible rapid turn-over of personnel should especially be considered.	Define details
Facilities	
Which services must be available at the required facility (eg, electrical power, hydraulic power, compressed air, special working environment)	Define details
Are existing facilities at operational locations appropriate to the requirements of a new Product (operational and maintenance facilities)?	Yes/No?
Can existing facilities at operational locations be adapted to the requirements of a new Product (operational and maintenance facilities)?	Yes/No?
Is there a realistic time schedule available for the building of required new facilities?	Yes/No? Define details
Is there a plan to identify and reduce risks caused by the delay of the building of facilities?	Define details
IT and communication resources	
Which IT architectures exists at every operational location concerning the following aspects:	Define details
<ul style="list-style-type: none"> - Network capabilities - Data storage capabilities - Computer capabilities - Communication resources 	

Detailed CRD questions	Answer/Action
Are the existing IT and communication capabilities appropriate for the new Products to be introduced?	Yes/No? Define details
Are there risks in connection with future changes in IT or communication architecture concerning the operation of the new Products?	Define details
Are there concepts for an initial provision with data for required IT systems?	Define details
Who is the holder of the required data?	
Are there contractual aspects to be considered?	
Are there any problems concerning data security requirements?	
Are there plans for a first installation of required new IT systems and for the appropriate service of IT systems during the life of the Product to be supported?	Yes/No?
Are there service contracts for hardware and software (planned and existing)? Fast reaction to downtime must be ensured. Upgrade of hardware and software in case of technical progress should be included in the agreements of a service contract.	Yes/No?
Are different IT systems at different operational locations compatible?	Yes/No?
Is there a plan to ensure the existence of all required interfaces to other IT systems?	Yes/No?
New organizational structures	
Will any organizational structures be changed at operational locations and what are the risks of these changes?	Define details
What structure changing opportunities exist with the introduction of a new Product (eg, reduction of personnel)	Define details
Schedule consideration	
What are the plans for an appropriate participation of logistic disciplines in all phases of the project?	Define details
Are the logisticians involved in the project in a very early phase of the project so they can influence basic decisions concerning design?	Yes/No?
Additional aspects	
What additional aspects specific to the project are of high significance for the logisticians?	Define details

10.3 LSA candidate selection flowchart examples

10.3.1 Non structural items

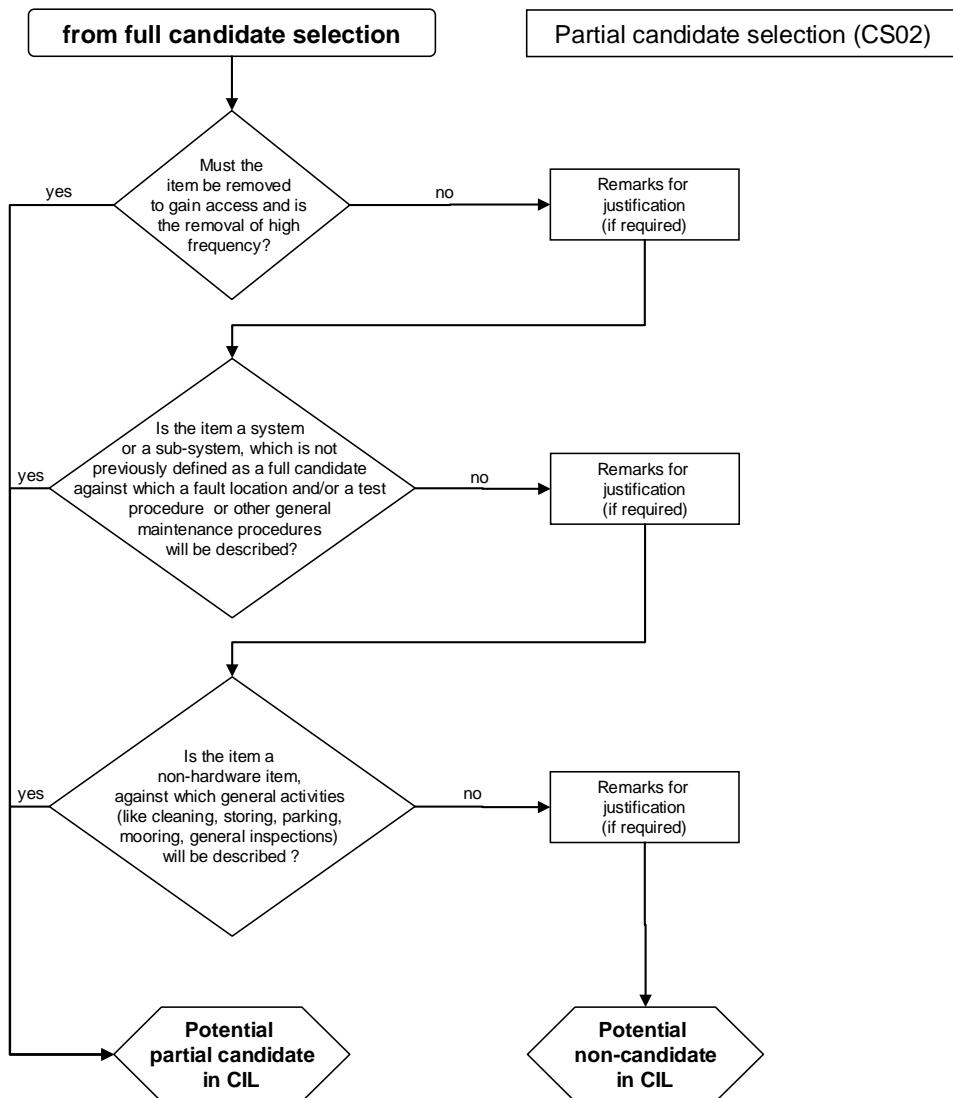


ICN-B6865-S3000L0036-001-01

Fig 15 Full LSA candidate selection flowchart

The selection flowchart from Fig 15 shows a typical process for the determination of full LSA candidates from an existing breakdown. The flowchart must be applied to each breakdown element. If the selection flowchart for full LSA candidates leads to the IUA not being a full LSA

candidate, the second LSA selection flowchart for partial candidates must be applied to each of those BE's. Refer to [Fig 16](#).



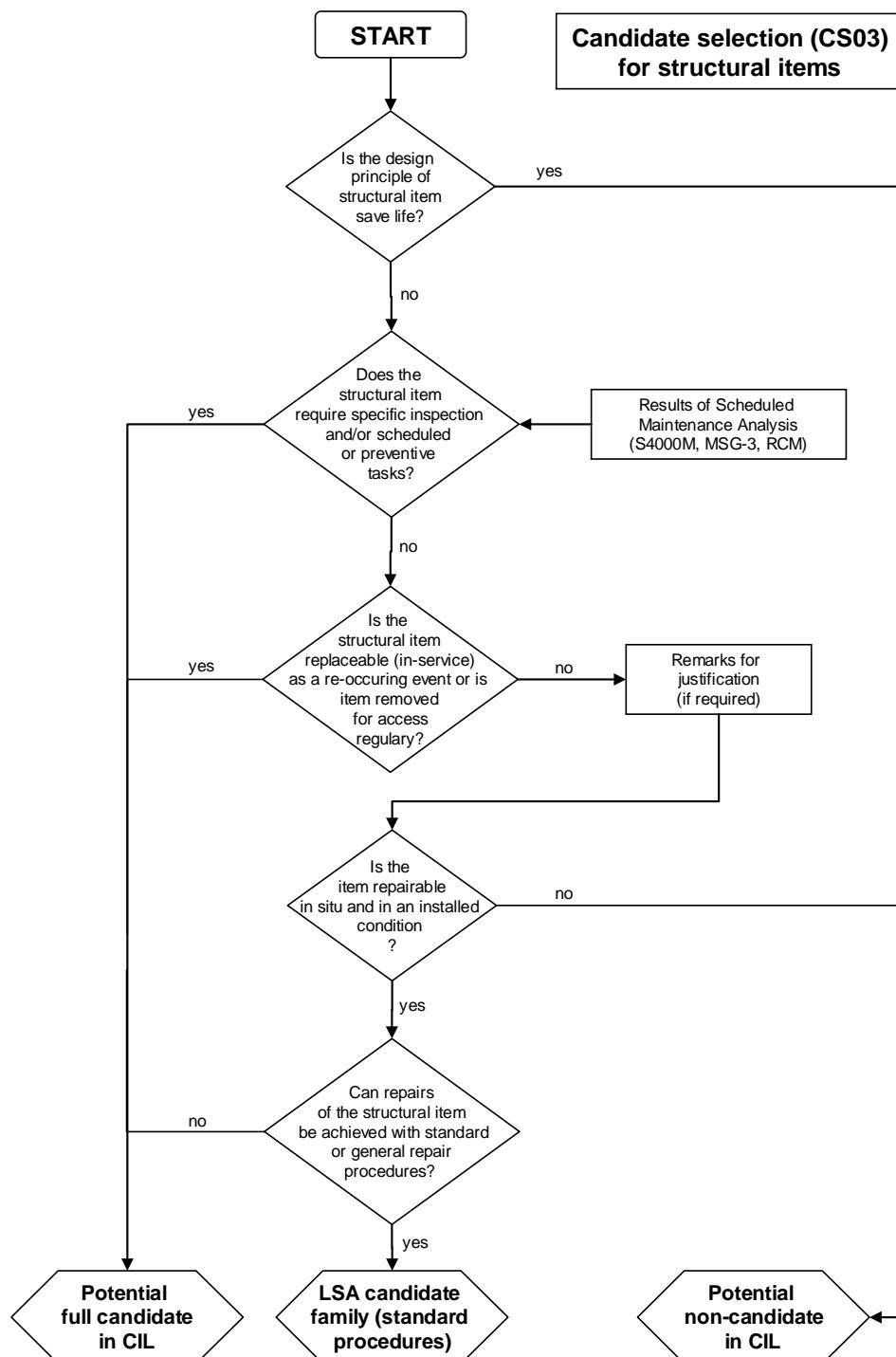
ICN-B6865-S3000L0037-001-01

Fig 16 Partial LSA candidate selection flowchart

Depending on the project, the selection criteria must be adapted since additional criteria could be required. In general, the analyst must have a guideline for the LSA candidate selection. Any required change of flowchart logic within a project must be harmonized between the contractor and the customer and agreed to the customer during the LSA GC.

10.3.2 Structural items

In the case of structural components, the selection of LSA candidates differs from the selection process for non-structural items. Refer to [Fig 17](#).



ICN-B6865-S3000L0038-001-01

Fig 17 LSA candidate selection flowchart for structural items

Chapter 4

Configuration management in LSA

Table of contents

	Page
Configuration management in LSA	1
References.....	2
1 General	3
1.1 Introduction	3
1.2 Objective	3
1.3 Scope.....	3
2 General considerations on configuration management.....	3
2.1 Configuration management rules	3
2.2 Configuration control	4
3 Configuration management in the LSA process	5
3.1 Establishment of breakdown according to agreed rules	5
3.1.1 Definitions of general terms.....	5
3.1.2 Types of product breakdown	7
3.1.3 Background for different approaches to establish a Product breakdown.....	11
3.1.4 Usage of breakdown element identifier	14
3.1.5 Breakdown element identifier examples.....	17
3.1.6 Breakdown element identifier and related codes	19
3.1.7 Breakdown element identifier requirements concerning software.....	24
3.2 Breakdown element identifier - structure examples	24
3.2.1 Functional breakdown	24
3.2.2 Mixture of functional and physical breakdown.....	25
3.2.3 Separation of physical and functional breakdown with cross referencing (hardware only)	26
3.2.4 Physical breakdown with explicit parent-child relationship.....	28
3.2.5 Integration of software as part of a single physical hardware item	29
3.2.6 Integration of software as part of more than one physical hardware item.....	29
3.2.7 Integration of software in a functional breakdown	30
4 Configuration changes in LSA	31
4.1 Evaluation of the impact of configuration changes	31
4.2 Sources/drivers for configuration changes	32
4.2.1 Customer requirements for change	32
4.2.2 Design change proposals	32
4.2.3 Manufacturing deviation from design (concessions)	33
4.2.4 Supplier changes	34
4.2.5 In-service changes (service bulletins)	34
4.3 Traceability between the source of changes and consequential LSA changes	34
4.4 Processing of configuration changes during the LSA process	34
4.4.1 General aspects.....	34
4.4.2 LSA compilation by change	34
4.4.3 Extended applicability management.....	35

List of tables

1 References	2
2 List of configuration identification rules (steps)	4
3 Common terms for replaceable items.....	6

4	Basic for explicit breakdown documentation	17
5	BEI readability aspects	18
6	Examples of usage of BEI and applicability from Fig 12	22
7	Aspect of replaceability.....	23
8	Aspects of reparability	23
9	Listing of functional breakdown, simple syntax with separators.....	24
10	Listing of functional breakdown, extended syntax with separators	25
11	Listing of mixed breakdown, extended syntax with separators	26
12	Listing of breakdown with parent-child methodology.....	28
13	Types of affectation	35

List of figures

1	Simple functional breakdown.....	8
2	Simple physical breakdown	8
3	Breakdown methodology - Simple parent-child philosophy	9
4	Breakdown methodology - Extended parent-child philosophy	9
5	Breakdown methodology - mixture of functional and physical methodology.....	10
6	Example for a spare part grouping concept	11
7	Product breakdown - traditional LSA approach versus PDM approach	12
8	System of systems breakdown.....	14
9	Product breakdown without structured BEI syntax	18
10	Example for simple BEI syntax (reduced to numbering system).....	19
11	Example for extended BEI syntax	19
12	Usage of BEI and applicability	21
13	Usage of BEI and applicability for traditional breakdown	22
14	Functional breakdown, simple syntax with separators	24
15	Functional breakdown, extended syntax with separators	25
16	Mixed breakdown, extended syntax with separators.....	26
17	Parallel usage of functional and physical breakdown.....	27
18	Cross referencing between functional and physical breakdown	27
19	Breakdown with parent-child methodology.....	28
20	Assigning of software BEI in a physical breakdown	29
21	Distributed software in a physical breakdown	30
22	Functional BEI structure concerning SW	31
23	MSN usage example	36
24	Fleet example	36
25	FSN (fleet) versus MSN.....	37
26	Applicability calculation.....	38

References

Table 1 References

Chap No./Document No.	Title
Chap 3	LSA Business process
Chap 13	Software support analysis

Applicable to: All

S3000L-A-04-00-0000-00A-040A-A

Chap 4

Chap No./Document No.	Title
Chap 19	Data model
S1000D	International specification for technical publications using a common source database
S2000M	S2000M - International specification for material management

1 General

1.1 Introduction

Configuration Management (CM) is a discipline that ensures the correct identification of different Product configurations, controls changes, and records the change implementation status of the physical and functional characteristics of the Product's structure, systems, subsystems, equipment and components. CM efforts enable the complete audit traceability of decisions and design modifications.

CM is performed starting at the lowest level of the Product breakdown. It identifies what was/is required, designed, produced, supported and evaluates changes, including impact on technical and operational performance and Integrated Logistic Support (ILS) activities. LSA as part of ILS is involved in CM processes during all project phases.

1.2 Objective

The goal of this chapter is to present LSA involvement in the CM process to allow configuration of products with the generation of the LSA structure through the Product breakdown. The Product breakdown is the basis of the Product structure for the other ILS disciplines. Examples are given to allow efficient CM of the operational Product as well as the support system. The chapter can be tailored to specific needs of a project.

1.3 Scope

This chapter is directed at ILS and LSA managers and is recommended also be read by other ILS discipline managers. It is focused on LSA CM, but since LSA is a core ILS process, it has many relationships with CM activities in other ILS disciplines. Most of the guidelines and rules can be applied to other ILS disciplines because there is a strong relation to other ASD specifications.

2 General considerations on configuration management

CM ensures correct and harmonized Product breakdown structure along all project areas (eg design, production, ILS). CM sets and maintains the baselines which define the Product. These baselines are further developed and adapted as required during the entire life cycle.

CM starts at the beginning of a project and continues through all phases of Product life cycle. ILS must be involved in the CM process from the beginning to participate in decisions and in the establishment of CM rules.

2.1 Configuration management rules

Clear CM rules must be established at project level in the early project phases to allow correct configuration traceability along all project areas.

A milestone plan must be established, which must follow the project and design milestones including project time schedules for the necessary contractor/customer documentation deliveries. Milestones and documentation deliveries must be under configuration control.

Effective configuration identification is a pre-requisite for all other CM activities. If configuration identification and their associated configuration documentation are not properly identified, it is impossible to control the changes to the Product's configuration, to establish accurate records and reports, or to validate the configuration through audits.

Inaccurate or incomplete configuration documentation can result in defective Products, schedule delays, and higher maintenance costs after delivery. Main configuration identification rules are listed in [Table 2](#).

Table 2 List of configuration identification rules (steps)

Rule	Description
Identification of configuration items (functional and physical)	The identification of an item must follow the design configuration and be compatible with other ASD specifications such as S1000D or S2000M. For example, the Standard Numbering System (SNS) from S1000D can be adopted for a functional configuration or a physical breakdown. Refer to S1000D.
Relationship between functional and physical configuration	Functional and physical configurations are two different points of view of the same item. Therefore, requirements for the management and link between these configurations must be established.
Relationship between design and LSA identification	Rules to guarantee the traceability between design configuration and LSA configuration must be established.
Zoning and access panel identification	Structuring the Product by zones to include identification of access panels is an additional approach to the configuration of a Product. Examples how a Product can be structured into zones are given in S1000D.
Identification variants and alternate items	Rules for the identification of variants and alternate items must be defined at the beginning of the project. This is important to identify different technical solutions for the same function.
Top level configuration identification	At the beginning of a project, the type of top level CM must be established. Top level CM serves to identify the Product variants or any other kind of grouping of Products.
Required documentation for proper configuration identification	The documentation necessary to properly identify the configuration, eg ASD breakdown document, Illustrated Parts Data (IPD), must be defined at the beginning of a project.

2.2

Configuration control

Configuration control includes the evaluation of all change requests, change proposals, and their subsequent approval or disapproval. This ensures that no change will be implemented without due consideration of its effect on the baselines, including logistics impact, costs, schedules, performance, or interface with any associate companies.

The authority required to make a decision on a change varies with the magnitude and nature of the change concerned. It involves the appropriate levels within the organization of the customer and the developer for the project. The decision making process is an integral part of the overall management of the "project management system".

3

Configuration management in the LSA process

LSA must be involved in the CM process from the beginning of the project. A fundamental part of the configuration is the generation of the Product breakdown. Rules must be established in order to configure the Product. These rules must ensure traceability of Product structure through all project areas (eg engineering, ILS).

According to the established rules, LSA must develop a Product breakdown capable to maintain multiple Product configurations. Establishing this Product breakdown as a basis for all ILS areas is essential.

3.1

Establishment of breakdown according to agreed rules

A systematic Product breakdown is essential to the LSA process and provides:

- A clear understanding of how the Product is structured with respect to systems, subsystems, functions, hardware and software components. This also applies to peripheral equipment that can be included in the LSA process (eg complex training equipment and support equipment).
- A clear relationship between the included hardware elements and their possible realizations in terms of manufactured hardware parts
- A clear relationship between the included software elements and their possible realizations in terms of actual software releases
- Enable the allocation of unique or unified identifiers. This enables the establishment of interfaces between design departments including configuration sources, LSA and the logistics disciplines such as technical documentation, materiel support, training and support equipment.
- Enable an appropriate level of detail for the selection of LSA candidates
- Identification of Product variants and Product configurations
- Enable CM with respect to how a specific revision of the Product breakdown relates to the correct revisions of its corresponding ILS end products. It also enables the CM of source data on which the Product breakdown and its ILS products is based, including documents, design information and Product usage data.

Different breakdowns of the Product can be defined for different purposes. In many projects it is common to have a design breakdown of the Product being defined in some kind of Product Data Management (PDM) system. The same Product often also has an LSA breakdown, as well as a separate breakdown for recording operational and maintenance data feedback. Having different breakdowns for different purposes often results in mismatches in between the different breakdowns, and complicates to communicate Product related data in between the different disciplines.

This chapter describes both, the situation where is need to define a separate breakdown of the Product from an LSA point of view, as well as the situation where the LSA process can reuse the breakdown as defined by design.

3.1.1

Definitions of general terms

3.1.1.1

Product/end item

The Product (or end item) is a final combination of systems, subsystems, component parts/materials and software (eg ship, vehicle, machines, aircraft or other complex technical systems). The Product represents the top level (root) of any hierarchical Product breakdown.

3.1.1.2

Product variant identifier

The Product variant identifier is a code, which uniquely identifies the Product variant. It is recommended to use the Product variant identifier in conjunction with other identifiers within the entire ILS process as eg the System Difference Code (SDC) from S1000D.

3.1.1.3 Product breakdown

A Product breakdown consists of the Product itself, and the hierarchical listing of, for example assemblies, subassemblies, and components that can be disassembled, reassembled or be replaced. Depending on the breakdown philosophy, a Product breakdown can also consist of systems, subsystems, functions and zones. Examples of types of Product breakdown are described in [Para 3.1.2](#).

The required depth of a Product breakdown in an LSA database normally will be different from, for example, the one being required for materiel support or production. Nevertheless, it is recommended to harmonize breakdown rules concerning the syntax of the Breakdown Element Identifier (BEI) within a project.

3.1.1.4 Breakdown element and breakdown element identifier

The BEI identifies the individual Breakdown Element (BE) that is defined within a Product breakdown. A BE can represent functional or physical BEs. In case of a physical BE typically an installation location is defined, which is “realized” by the installation of a concrete hardware item like a piece of equipment.

It is recommended to establish the BEs and their corresponding BEIs harmonized with the other logistic disciplines such as technical documentation or materiel support. Additionally, a commonality as far as possible between the design/engineering view on the Product and the ILS view on the Product is recommended.

The BEI enables to uniquely identify an item that is a part of any logistic analysis activity.

[Para 3.1.2](#) describes the different types of systematic Product breakdowns. Depending on the breakdown type, different rules for the establishment of the BEI can be determined. Refer to [Para 3.2](#).

3.1.1.5 Breakdown element revision

A Breakdown Element Revision (BER) defines a new release of a BE. BER captures the development progress (often referred to as design iterations). The use of BER supports CM and especially the synchronization between changes that are being introduced from Product design, and the impact on the LSA results.

Note

The BER must not be mixed with methods to document alternate components within a Product breakdown as, for example, by Alternate Logistic Control Number (ALC) from MIL-STD 1388-2B or the disassembly code variant from S1000D. The situation, that an item which is installed at a certain position within the Product breakdown has two different realizations (eg two different pieces of equipment from different manufacturers which have different maintenance concepts) is covered in S3000L by the means of related realizations in terms of parts (hardware or software).

3.1.1.6 Replaceability aspects

Some common terminology has been established in the area of logistics concerning the replacement of equipment/components. Refer to [Table 3](#):

Table 3 Common terms for replaceable items

Term	Full name	Description
LRU	Line Replaceable Unit	Directly replaceable on Product level
SRU	Shop Replaceable Unit	Replace on LRU or SRU level, but not directly replaceable on Product level under normal circumstances. This means, the next higher component (LRU or SRU) can need to be removed before the SRU can be replaced.

Note

Instead of LRU and SRU, the abbreviations LRI (Line Replaceable Item) and SRI (Shop Replaceable Item) are also used.

3.1.1.7 Parts

Parts are the actual realization of BEs defined in a Product breakdown, and can be either hardware or software items.

3.1.1.8 Product related applicability - Management of Product variants and configurations

Product related applicability (often also referred to as effectivity) is used to reflect the configuration information of a BE with respect to its allowed installation in different Product variants. It covers the need to limit the applicability of any item in a Product breakdown to special Product variants. An example for the usage of BE and applicability is given in [Para 3.1.6.2](#).

Note

It is recommended that CM aspects between customer and contractor and the significance of CM must be clarified by both the customer and the contractor.

It is recommended not to use the LSA database as a CM tool for the as-built situation (eg to enter the configuration of each end item delivered to the customer, into the LSA database).

3.1.2 Types of product breakdown

A Product breakdown can contain both, functional and physical aspects. These two approaches are illustrated by examples in [Fig 1](#) and [Fig 2](#). An LSA database must be able to document both methods and combinations thereof (so called hybrid Product breakdown).

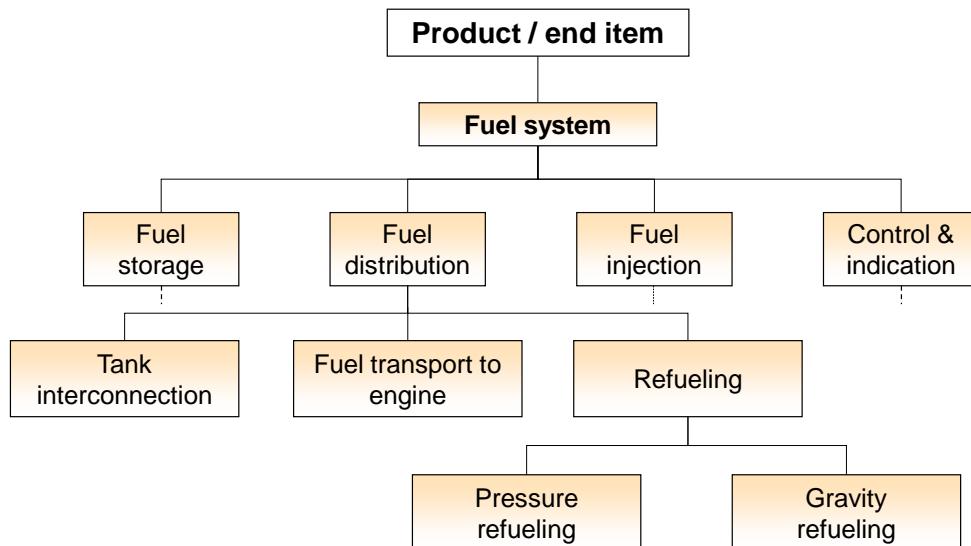
The interrelationship of an item with other items at the same indenture level and the next higher indenture, as well as its breakdown items (eg attaching parts), can be reflected by the BEI and BE realizations or via explicit parent child relationships, respectively. To reflect a Product breakdown the following methods are recommended, depending on the requirements of the project.

- A pure functional breakdown
- A pure physical breakdown
- A mixed breakdown, functional and physical in one breakdown tree (often referred to as hybrid breakdown)
- A pure functional and pure physical breakdown in parallel, interrelated by cross-mapping

Special attention must be given to software. In modern state-of-the-art Products, software is an integral part. This must be considered in any logistic analysis. It can be required to integrate loadable software packages into the Product breakdown, if the installation/de-installation of software is maintenance relevant. The assignment of software to hardware must be clear within the Product breakdown. Configuration changes due to software modification are another aspect of the breakdown. If hardware is not modified, but software integrated in the hardware changes, this can be necessary to be documented in the Product breakdown.

3.1.2.1 Functional breakdown

A functional breakdown typically starts on Product level as the root of a breakdown tree. The different Product functions are documented from the main functions down to the sub-functions to the required depth. Hardware elements cannot be found within a pure functional breakdown. For a simple example, refer to [Fig 1](#).

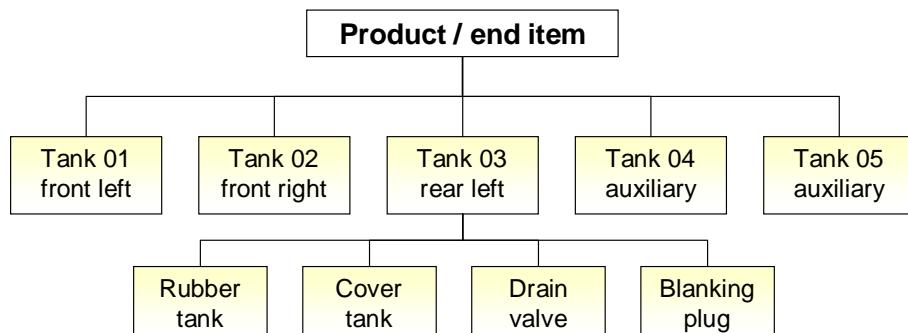


ICN-B6865-S3000L0008-002-01

Fig 1 Simple functional breakdown

3.1.2.2 Physical breakdown

A physical breakdown also typically starts on Product level as the root of the breakdown tree. As a general principle, in each physical breakdown it must be possible to structure the breakdown by the usage of assemblies. As shown in [Fig 2](#), this is realized on the second indenture level by introducing tank assemblies, which are broken down to the single components rubber tank, cover tank, drain valve and blanking plug.



ICN-B6865-S3000L0009-001-01

Fig 2 Simple physical breakdown

Below the Product itself, the hardware elements are documented. Functional elements cannot be found within a pure physical Product breakdown.

3.1.2.3 Zonal breakdown aspects

A zonal breakdown can be accomplished by sequencing all the zones defined for the Product. The required depth and structure of a zonal breakdown established in an LSA database can be different from, for example, a Product breakdown being defined for design or production purposes. Nevertheless, it is recommended to harmonize Product breakdown rules within a project.

Zones as BEs are typically required for the documentation of maintenance activities which are related to the physical area represented by a zonal BE (eg for scheduled zonal inspection). To

describe the content of a zone, it must be possible to establish relationships between zonal BEs and physical BEs within an LSA database.

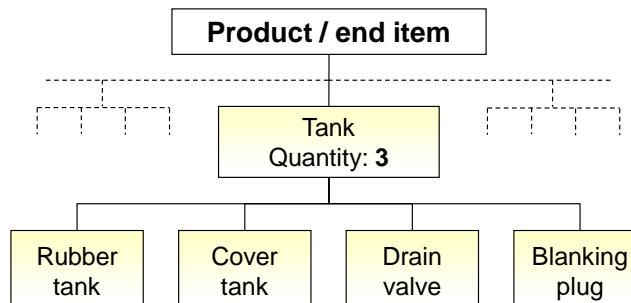
3.1.2.4

Installation location of equipment/components and multiple installation

In general, the installation location is the primary information within physical Product breakdown. Multiple installations of equipment/components can be considered in two different ways:

- One BE with a quantity information:

In the example the same type of tank is installed three times within the system, refer to [Fig 3](#). The breakdown below the tank is always the same. In some cases, it can be sufficient to have only one BE for the three tanks within the Product breakdown including the information of the corresponding quantity.

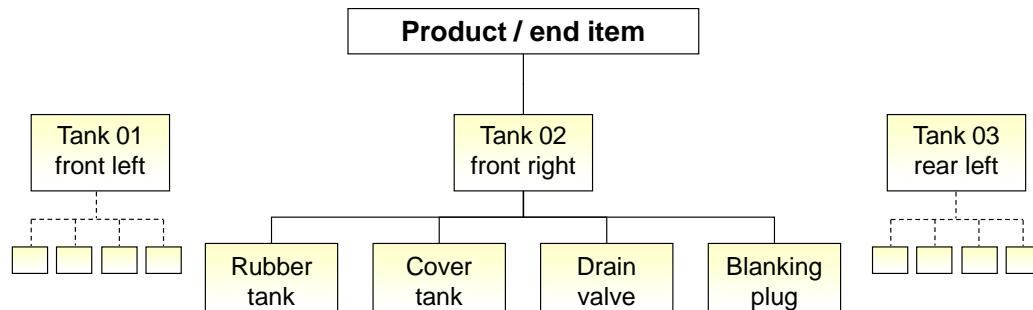


ICN-B6865-S3000L0010-002-01

Fig 3 Breakdown methodology - Simple parent-child philosophy

- Several breakdown elements:

In the example each tank is considered as a separate BE, refer to [Fig 4](#). Using this method, the focus is on the installation location (eg the installation or the removal can be different depending on the installation location).



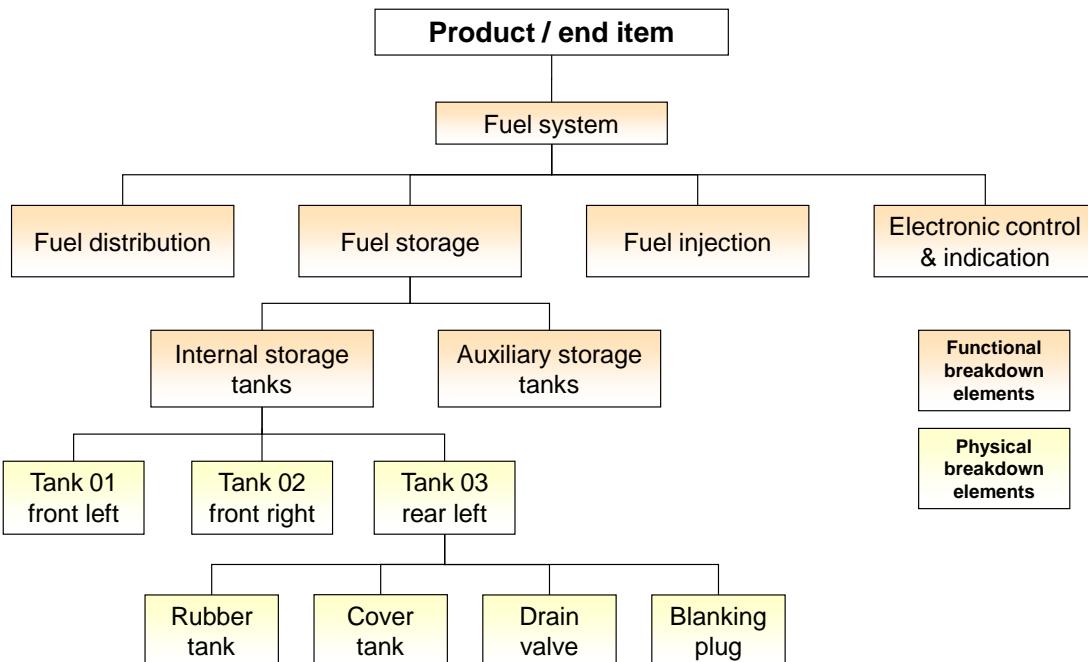
ICN-B6865-S3000L0011-002-01

Fig 4 Breakdown methodology - Extended parent-child philosophy

3.1.2.5

Mixture of functional and physical breakdown methodology

A widely-used Product breakdown approach is a mixture of functional and physical aspects within one breakdown tree, as shown in [Fig 5](#). Typically systems and subsystems are of functional character, equipment and components are physical BEs. Nevertheless, a system need not represent one concrete function but more a collection of components that have a set of common characteristics (eg an electrical system). In this case, a system is built from a set of functions/components, but there is still a system BE which is neither a concrete function nor a physical item.



ICN-B6865-S3000L0012-002-01

Fig 5 Breakdown methodology - mixture of functional and physical methodology

If a mixture of functional and physical breakdown methodology is used, it can be useful to first generate a functional structure of the Product by dividing it into several basic systems and subsystems (eg propulsion, electric system, structure, hydraulic system, fuel system).

Depending on the magnitude of a system, the indenture level where the documentation of real hardware BEs begins can vary. The typical breakdown path, system-subsystem-equipment-component is not always fixed, so the indenture level at which LSA candidates are located can also be different from system to system.

3.1.2.6

Parallel usage of functional and physical breakdown methodology

If establishing both functional and physical breakdowns in parallel is required for different analysis purposes, each breakdown tree is established as separate breakdown structures. The decisive factor is the generation of an effective interconnection between these two breakdown trees. Physical BEs must be linked to functional BEs.

This methodology can be a good basis for troubleshooting analysis. Based on a functional breakdown, troubleshooting must be able to identify potential justifications that certain functions are not available. For this reason, the analyst must know which hardware elements can be responsible for the missing function. The parallel usage of functional and physical breakdowns also allows assigning hardware to functions. For example, the function "fuel distribution" shown in [Fig.5](#) can contain typical elements such as fuel pipes, but also electric power supply elements. However, even though electric power supply is documented in another area, it must be assigned to the function "fuel distribution".

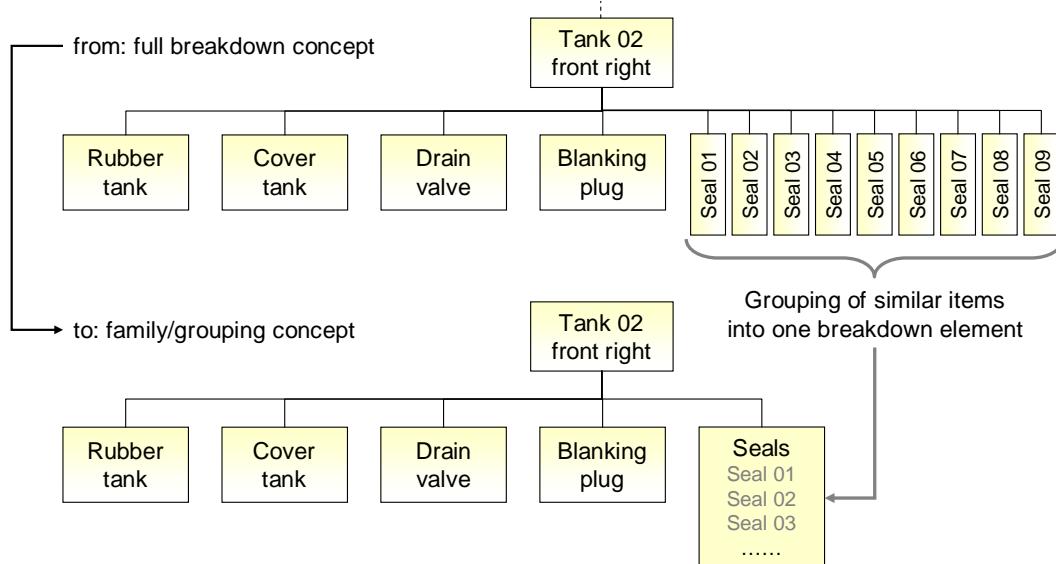
The cross functional interconnection between different Product breakdown methodologies must be considered in an S3000L based logistic database structure.

3.1.2.7

Family concepts in physical breakdown methodology

To reduce efforts in establishment a physical Product breakdown, family concepts should be used whenever possible. Factors that can be considered when deciding whether or not to use a family concept include:

- Similar items with common maintenance concepts can be grouped by family BEI concepts (eg harnesses of the same type, pipes of the same type, and structural items of the same type)
- Standard spare parts can be grouped by one BEI family. Refer to [Fig 6](#)



ICN-B6865-S3000L0013-001-01

Fig 6 Example for a spare part grouping concept

However, grouping of items by one BEI can lead to the need to document the single group elements in a certain location within the LSA database. The required logistic information is documented within the group BE.

3.1.3

Background for different approaches to establish a Product breakdown

As mentioned in [Para 3.1.2](#), a Product breakdown can be defined for different purposes. The respective breakdown is traditionally often defined and used by a specific discipline. However, the correlation between these breakdowns is often non-existent, which means that there is a problem exchanging data and sharing experiences between different disciplines. Examples are:

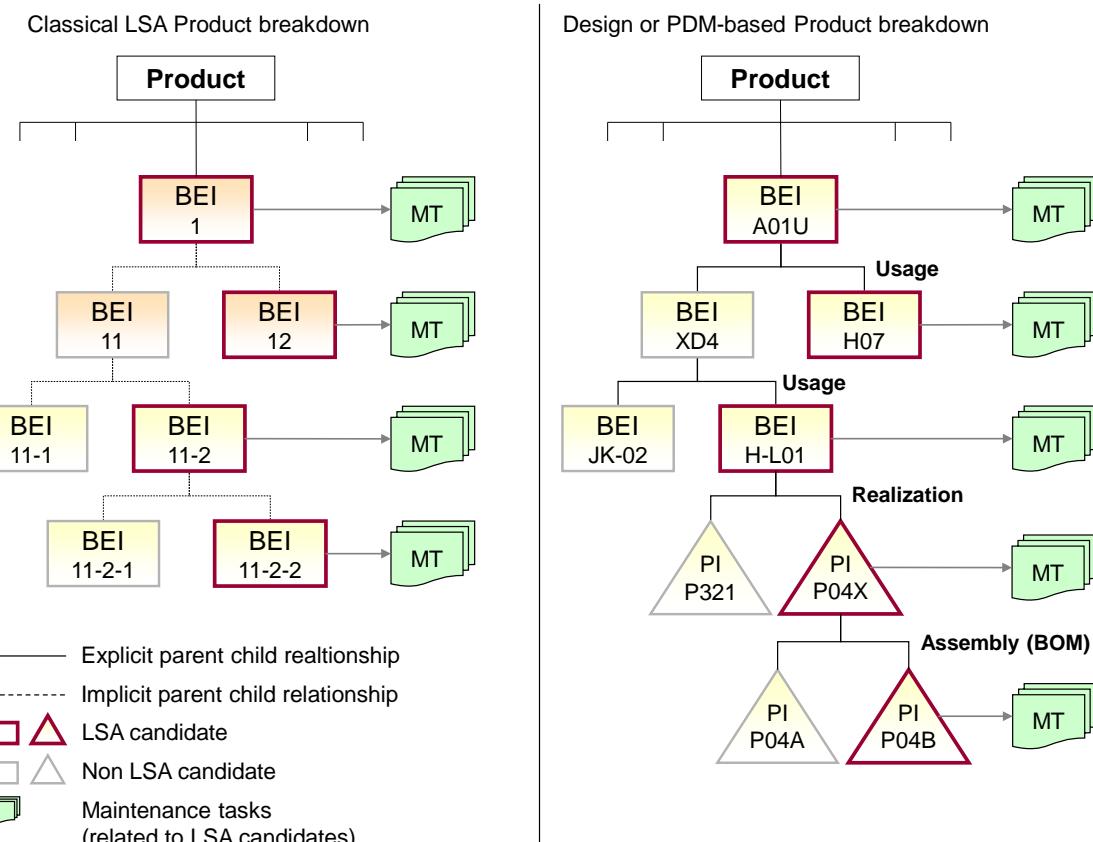
- BEs that represent a position in a Product are often identified in different ways within different breakdowns
- The breakdown philosophy (methodology) is often different from each other, which makes it hard to identify similarities and differences between the different breakdowns

Product breakdown approaches are driven by different requirements. The working area of design and development needs a different Product breakdown than, for example, support engineering for technical/logistic analysis activities. For that reason there are specific views on how to structure a Product breakdown. The intended implementation of S3000L concerning this subject must be as clear as possible but also be flexible if required. The integration of the two main approaches to Product breakdown into one data model, which supports both alternative solutions, is a main goal of S3000L. These are:

- Approach 1:
Traditional LSA Product breakdown by means of structured BEI (eg similar to the SNS structures in S1000D or to LCN/ALC structures as described in MIL-STD 1388-2B or DEF-STAN 0060). Functional and physical areas are integrated into one common Product breakdown. Indenture levels are defined in an implicit way by the syntax of the specific identifier (BEI).

- Approach 2:
Design oriented or PDM based Product breakdown by means of explicit parent-child relationships. These breakdowns are driven by relations of BEs and/or hardware components. The identifiers of the components are not structured by syntax to document indenture levels as in approach 1. The identifiers can be BEIs or at lower levels a PI (Part Identifier) which represents a part as a realization of a breakdown element.

A graphical overview of the main differences of these two basic Product breakdown approaches is given in [Fig 7](#):



ICN-B6865-S3000L0082-001-01

Fig 7 Product breakdown - traditional LSA approach versus PDM approach

3.1.3.1 Traditional LSA Product breakdown

A traditional LSA related Product breakdown uses an implicit breakdown approach. An implicit breakdown approach defines the position of the BE by the syntax of its BEI. This methodology can be established by using existing standards such as MIL-STD 1388-2B or DEF-STAN 0060 respectively. In these basic standards the breakdown elements are represented by a BEI known as the LCN. Another approach can be by using the S1000D Standard Numbering System (SNS).

The consequence for LSA process is that the design breakdown must be quite stable before the LSA work can start. One reason is that once the LSA Product breakdown has been established, it can cause a high level of effort to introduce intermediate levels into the breakdown.

Another consequence of using structured BEIs to organize breakdown hierarchies is that a structured BEI cannot be reused in between different Products in a simple way.

Incorporating breakdowns from subcontractors enforces the subcontractor to adopt the primary contractors BEI philosophy, and implement the BEI philosophy into its process and

documentation. This increases the cost for updating and maintaining Product breakdowns that already exists.

Note

In traditional Product breakdown concepts there is no mechanism to document revisions of a BE based on design progress. For that reason it is hard to synchronize changes originating from design since there is no way to track the revision of a BE based on development status.

3.1.3.2 PDM-oriented product breakdown

Product breakdowns in modern PDM systems define explicit parent-child relationships between its constituent breakdown elements. This means, that a breakdown can be established without having a structured method for the identification of the respective BE. Using explicit parent-child relationships requires that each BE documents:

- Relationships to its child BEs
- Each parent-child relationship can include information about the quantity of child elements and installation location

Note:

The unique BEI for BEs as defined by design are often generated by the PDM system as a random code.

BEs are often defined as being a specifications for a system, subsystem, hardware or software. Therefore, the breakdown stops at a level where a BE can be realized by an assembly (hardware part) or a software package (software part). Assemblies and software packages can in turn have a defined bill of material or a defined assembly structure.

The advantage of using explicit breakdown structures is that all BEs and parts can be reused in many installation locations and in multiple breakdowns (eg for different Products) without having to define new elements, and new intermediate levels can be introduced at any time during the project, without causing additional effort.

Another advantage is that there is no need to define alternative BEs, since alternatives are just different realizations of a BE by different parts.

The incorporation of breakdowns and parts from subcontractors is simpler, because there is only a need to define a parent-child relationship between a main contractor's leaf element and the subcontractors root element. This is much more cost effective than enforcing a complete identification system which the subcontractor needs to maintain.

3.1.3.3 Combining the two product breakdown approaches in S3000L

Even though the S3000L data model supports both approaches as such, a "combination" of the two approaches is recommended. This is achieved by:

During the early phases of LSA:

- Use the Product breakdown, BEs and their BEIs as defined by design (including the identification of revisions in order to be able to identify and track design changes)
- If there is a need to introduce BEs required by LSA, then:
 - First, try to influence the breakdown structure as defined by design
 - Second, if there is possibility to add this to the design, then create a separate branch in the breakdown. Do not enter intermediate BEs, since that complicates the management of breakdown updates coming from design.
- If there is a need to add BEs to the design breakdown, incorporate Product breakdowns from subcontractors using the subcontractor BEIs, if possible
- Identify assemblies, subassemblies and components by part numbers and not by their position in a breakdown

At delivery of the LSA results:

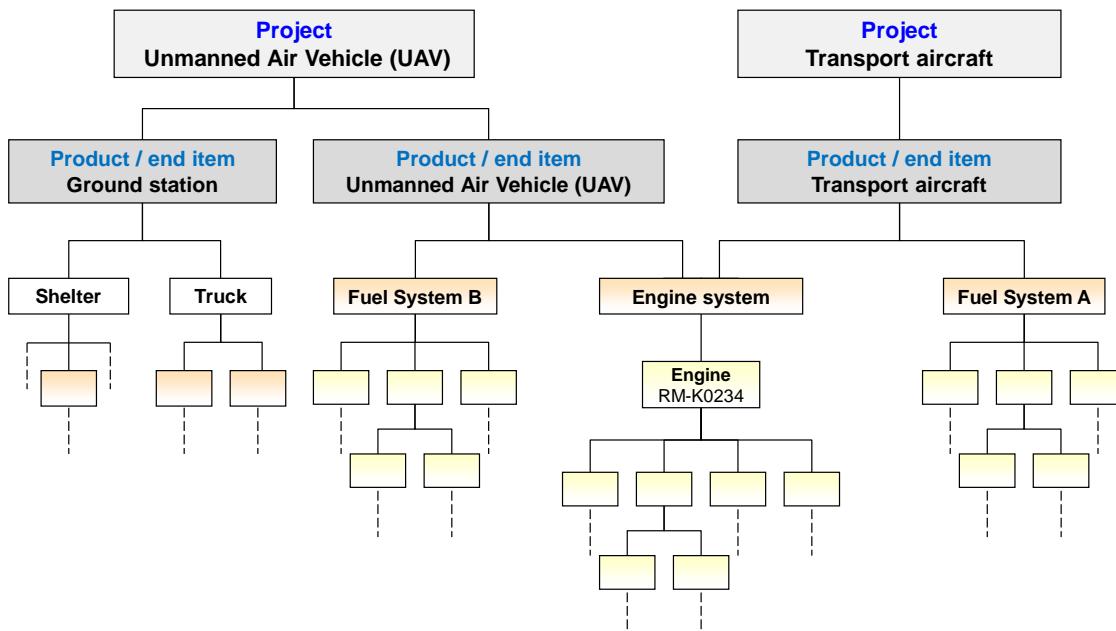
- Apply a systematic identification system to the BEs from the Product itself down to a sufficient breakdown level. One way is to stop at the level where a breakdown element (specification) can be realized by a hardware or software part.
 - Do not enforce BEI to parts (eg to actual realizations)

This approach will support both the flexibility required during the LSA process, and provide a cross-discipline solution.

3.1.3.4 System of Systems breakdown

The system of systems approach includes multiple, dispersed and independent Products as part of a more complex superordinate project. In this context, the same Product can be used in different projects (eg the same engine can be used in two different types of aircraft).

The example in Fig. 8 shows a typical situation. Two different projects include the same Product, in this case an engine for an aircraft. One and the same existing Product breakdown can be integrated into both projects.



ICN-B6865-S3000L0108-001-01

Fig 8 System of systems breakdown

3.1.4 Usage of breakdown element identifier

The data element to structure Product breakdowns is the BEI. The BEI uniquely identifies the BE. The relation of an item with other items at lower or higher indenture levels can be reflected by BEI syntax (traditional approach) or by parent-child relationship (PDM approach).

3.1.4.1 Breakdown element identifier syntax

For a traditional Product breakdown approach, the BEI syntax defines the relationship between the different breakdown indenture levels. The number and the type of characters assigned to each indenture level must be defined. The BEI syntax must be developed as early as possible and it must be ensured that the syntax is appropriate from the beginning to all phases of the project. It is recommended that using more characters than required for each indenture level be avoided. Ensure that the BEI syntax is capable, consistent, and broad enough to address the entire project structure. The limitations of LSA IT applications as they relate to BEI length and syntax must be taken into account.

Note

For projects using the traditional LSA Product breakdown approach, any change to the BEI syntax can cause extended rework within different LSA IT systems and associated ILS disciplines. There can be a risk that an indenture level requires more digits than originally planned for. In these cases, using an additional digit from the beginning can avoid BEI syntax problems at a later stage in the project.

For a PDM Product breakdown approach the definition of a specific BEI structure is not required. In this case, the parent-child relations of the breakdown elements are given by explicit relationships within a database.

3.1.4.2

Alternative breakdown element

Projects using the traditional LSA Product breakdown will meet the requirement to be able to define alternative BEs similar to the former usage of ALC from MIL-STD 1388-2B or Disassembly Code Variant in S1000D for example.

Since this kind of breakdown does not distinguish between the function being performed and the component that realizes the function, there is a need to introduce so called alternative breakdown elements (LCN/ALC in MIL-STD 1388-2B and Disassembly Code Variants in S1000D). An alternative BE within MIL-STD 1388-2B uses the same LCN, but different ALC to document alternative BE realizations. The same LCN indicates that the alternate BE occupies the same installation location in the Product breakdown.

The requirement for alternative BEs can be the result of, for example, different manufacturers that are selected for an item at a specific installation location, but the respective manufactured item requires different logistic considerations (in form, fit, function and interface, they are the same). In S3000L, the usage of different BEI to document this situation is not mandatory.

Alternate items to be installed at the same installation location within a Product breakdown can be different realizations by different hardware (or software) at the same installation location. The representation of these alternate realizations can be done in two different ways within S3000L:

- Introduction of additional BEs with own BEIs. Relation to different Product variants can be realized by using Product related applicability.
- Introduction of different realizations of a BE by different parts (eg from different manufacturers). These parts can also become LSA candidates on a part level for the documentation of different logistic considerations, if required.

Note

For the documentation of alternative parts to be possibly installed at one and the same installation location, which are exactly the same in form, fit, function, interface and logistic considerations it is sufficient to document the logistic data against the corresponding BEI without making the alternate part an LSA candidate, too.

3.1.4.3

Documentation of peripheral items

Besides the Product/system that is subjected to an LSA (prime LSA item), analysis of peripheral items can also be required. Those peripheral items (which include special tasks) have to be addressed by unique BEI and have their own Product breakdown composed of eg BEs and parts. However it is necessary for the BEI to follow the BEI logic that is defined for the prime LSA item.

Examples of peripheral items (enabling Products) are:

- LSA for complex support equipment (eg test equipment, working platforms)
- LSA for complex training equipment (eg simulators, training rigs)

3.1.4.4 Identification of item families

Similar items can be grouped together for analysis purposes. This family concept can minimize the efforts and can avoid the repetition of similar analysis activities. The following list shows typical examples for items which can be summarized in a family concept.

- Wiring of the same type (eg standard copper, coaxial, fiber-optical) within harnesses can be grouped together into families. Within these families, standardized tasks for cable maintenance can be identified.
- Structural parts of the same type can be grouped together. Within these families, standard structure repair procedures can be defined.
- Grouping by manufacturer. All parts of a similar type coming from one manufacturer, can be grouped, because the maintenance of these parts is covered by a maintenance concept supported by the manufacturer itself (eg with corresponding repair kits).

3.1.4.5 Integration requirements

It must be ensured that each BE can be identified uniquely by each logistic discipline by both the customer and the contractor. It is recommended to consider the following aspects:

- Harmonization of terminology

Use item names (eg derived from early design documents) unchanged and commonly within the associated logistic disciplines in order to avoid confusion by the usage of different names for the same item.

- Harmonization of key addresses between the logistic disciplines

Keep the IT key addresses as harmonized as possible in order to ease communication or interaction in between different disciplines. For that reason, a common identification of BEs and parts is recommended to a maximum extent within all logistic disciplines and also with design department and CM.

- Breakdown element identification adopted to a standardized numbering system

It is recommended that BEIs correspond to an SNS as used, for example, by technical documentation or design. It is recommended that numbering systems of LSA and other technical/logistic disciplines are harmonized.

3.1.4.6 Establishment of an exchange system for data and documents

By using the BEI and part numbers as a common key, the following requirements can be supported:

- To register and exchanged related ILS elements, internally and between partner companies and with customers
- To establish interfaces between the logistics disciplines, design and configuration much easier
- To enable the comparison of data between logistic disciplines in order to guarantee commonality and data quality
- To identify functional and/or physical positions of all LSA candidate items
- To ease trouble-shooting
- To address reports derived from or received by the LSA database (eg for logistic disciplines, management)
- To address data exchange with the customer, partner companies or suppliers
- To address LSA status information to breakdown elements
- To structure documents that are relevant for LSA and/or other disciplines (eg technical specifications, handbooks, general documents, drawings and document numbering systems)
- To enable structuring of other LSA documents such as the CIL

3.1.5 Breakdown element identifier examples

Within BEI logic, some special aspects of information can be incorporated. The main purpose of the BEI is the structuring of a Product into hierarchical levels from the top level down to the required breakdown depth, in order to document the relevant equipment and components. The establishment of rules for the syntax of the BEI is important. These rules are agreed to at the LSA GC.

3.1.5.1 Breakdown without homogeneous BEI logic

A Product breakdown can be established without having a structured BEI syntax. Each BE only needs to identify its children. Each association with the respective child BE contains information about the quantity of child elements and possible description of installation location.

Note

Quantity and installation location is data being associated with the relationship between the parent and child BEs.

Table 4 Basic table for explicit breakdown documentation

Item name	BEI	Part number	Item type	Next higher item	Quantity
Product/end item	-	-	Product	-	-
Fuel tank	TF001	-	Equipment	Product	3
Auxiliary tank	TF004	-	Equipment	Product	2
Rubber tank	TRT001	-	Component	TF001	1
Tank cover	TC001	-	Component	TF001	2
Blanking plug	TBP001	-	Component	TF001	1
Drain valve	TV001	-	Component	TF001	1
Valve1	-	45-90F	Component	TV001	1
Valve2	-	45-90H	Alternate component	TV001	1
Filter1	-	27-12	Component	45-90F	1
Body	-	45-BB	Component	45-90F 45-90H	1
Filter2	-	27-27	Component	45-90H	1
...

The breakdown structure example from [Table 4](#) is visualized in [Fig 9](#).

Note

Normally, any functional information is missing in this type of explicit breakdown methodology.

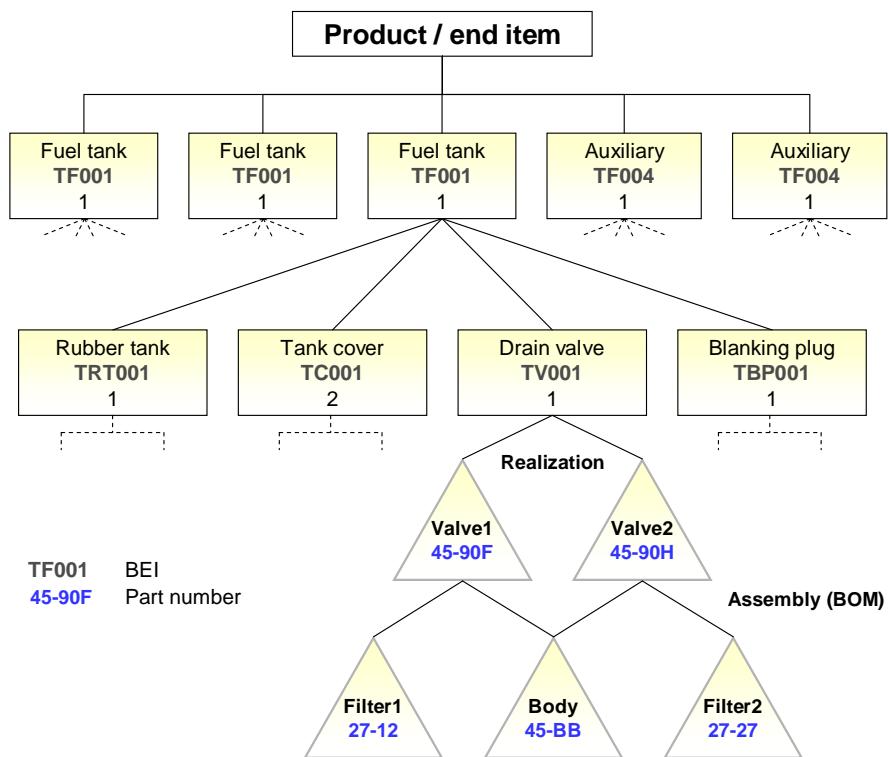


Fig 9 Product breakdown without structured BEI syntax

3.1.5.2

Basic syntax of breakdown element identifier - establishment of breakdown levels

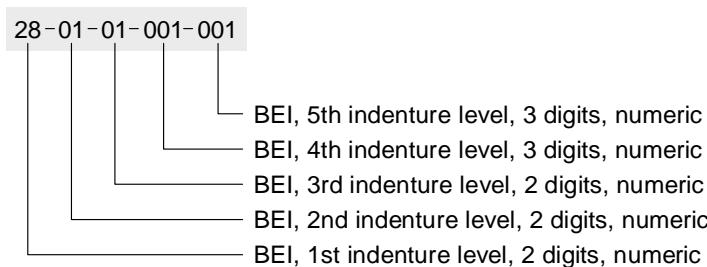
The implementation of a BEI syntax must consider the definition of the different indenture levels and how to make the BEI easily comprehensible. Both the number of digits per indenture level and the proper usage of letters, numbers and special characters must be defined carefully for each project. Each aspect concerning the required depth and range, readability and possible future requirements must be considered. Readability can be improved by taking into account the following aspects, refer to [Table 5](#):

- The usage of separators between different levels is recommended
- A decision made on whether to always use the full range of digits in the breakdown syntax or to allow a BEI termination at the relevant indenture level.

Table 5 BEI readability aspects

BEI		BEI (for better clarity)
28000000000AAA	⇒ better readable by usage of a separating character	⇒ 28-000-000-000-A-AA
28000000A	⇒ better readable by usage of a separating blank	⇒ 28 000 000 A
28-000-000-000-000		28
28-001-000-000-000		28-001
28-001-001-000-000	⇒ better readable by termination after the last relevant breakdown level	28-001-001
28-001-001-002-000		28-001-001-002
28-001-001-002-001		28-001-001-002-001

A typical example for a simple BEI syntax is given in [Fig 10](#):

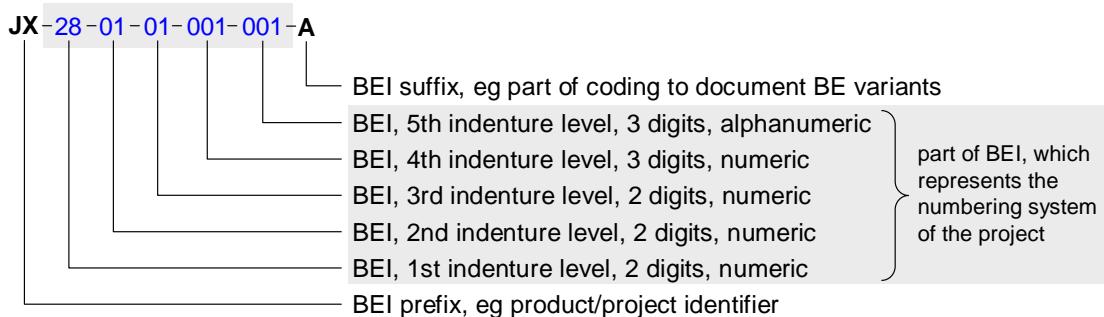


ICN-B6865-S3000L0015-001-01

Fig 10 Example for simple BEI syntax (reduced to numbering system)

In addition to readability aspects, other information can be incorporated within the BEI, refer to [Fig 11](#). Examples are:

- BE variant coding
- Coding of the project/Product, eg a Product identifier



ICN-B6865-S3000L0016-001-01

Fig 11 Example for extended BEI syntax

Note

The definition of the BEI syntax as a code is not restricted in S3000L. The BEI can be used as a simple representation of the numbering system as shown in [Fig 10](#). To incorporate additional information, the BEI can be extended eg by a prefix or a suffix as shown in [Fig 11](#). In this way the documentation of different BE variants can be realized.

JX-28-01-01-001-001-A and JX-28-01-01-001-001-B can be two different variants of breakdown elements being installed at the same installation location. The installation location is documented by the part of the BEI representing the numbering system of the Product breakdown, the variant is identified by the suffix.

3.1.6

Breakdown element identifier and related codes

Along with the coding of the functional/physical breakdown element with the help of a BEI, additional information is important for the unique identification of any Product component. The usage of this information along with the BEI will provide a complete idea of the structure and assembly of the analyzed Product.

3.1.6.1

Purpose of breakdown element revision

The Breakdown Element Revision (BER) documents different stages of development. Especially in a design & development phase it can be necessary to document different development steps concerning influence on design and principal changes which have significant influence on technical/logistic analysis results and/or on the support solutions. Revisions of BEs have not been considered in traditional Product breakdown methodologies, but is being

introduced in S3000L in order to enable a better integration with Product design and logistic disciplines.

Note

The BER is not designated to be the code to distinguish between for example, alternate hardware components (eg from different manufacturers) which can be installed at the same installation location (also refer to [Para 3.1.4.2](#)).

- 3.1.6.2 Applicability
- Often different equipment (eg from different manufacturers) or equipment variants (eg from the same manufacturer, but different releases) can be installed at the same installation position within a Product. The usage of applicability for different BEs and BE realizations by parts enables for the distinction between variants of equipment/components documented by BEI or part identifier which can be installed within different variants of Products. Concerning variants of components or parts respectively it is possible to distinguish between 2 different situations:
- Situation 1:
Simple alternative components (equal in form, fit, function, interface and logistics considerations, eg interchangeable components from different manufacturers).
 - Situation 2:
Modified alternative components (equal in form, fit, function and interface, but not equal concerning logistics, eg equipment/parts of the same type from different manufacturers, which have different maintenance concepts).

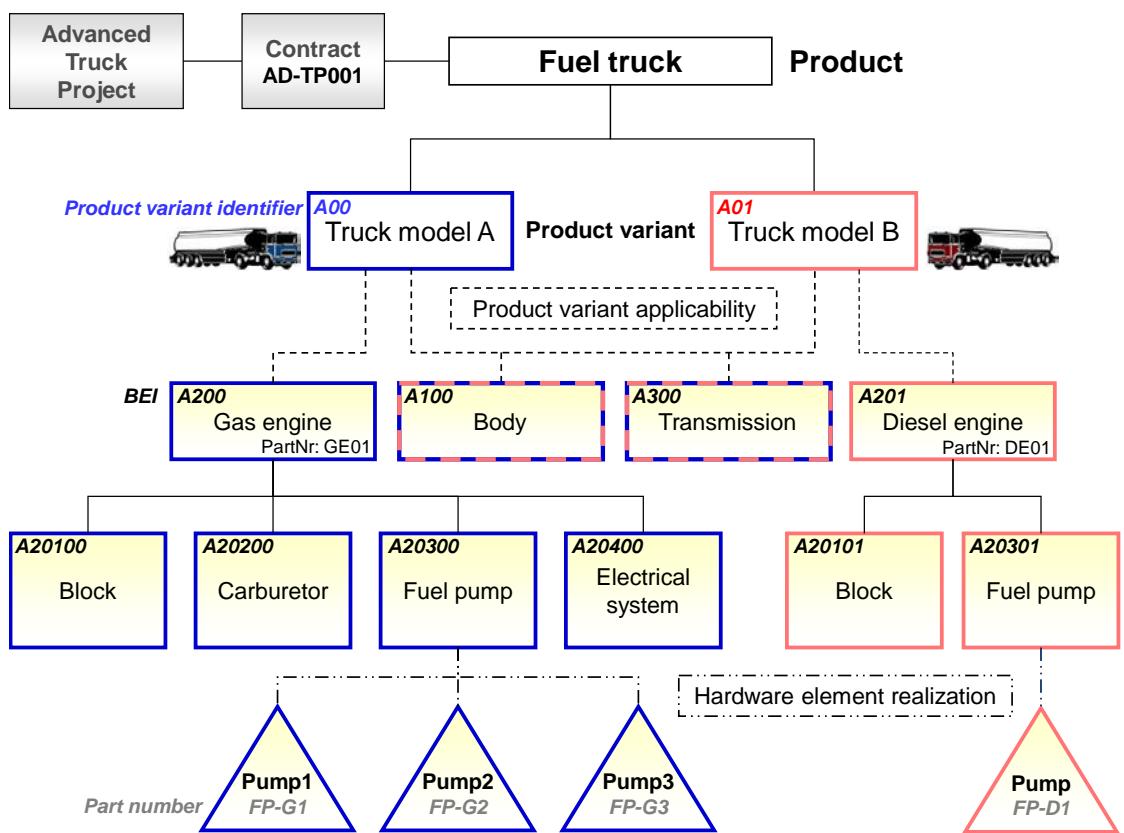
In general, situation 2 has to be documented carefully by the introduction of different items within the Product breakdown. These items can be different BEs represented by different BEI, or, alternative, different realizations for one and the same BE by different parts. In both cases it is possible to document the different logistic information against the LSA candidate, which is represented in the first case by the BEI and in the second case by the BE realization (installation of different parts at the same installation location, eg identified by part number).

Additional to the requirement to be able to distinguish between variants of components or parts respectively, the Product as the superior item can also be realized by different Product variants (eg for different customers). A proper applicability method must be able to allocate each component or part variant to the corresponding Product variant.

In the following S3000L Product breakdown example the principle of assigning applicability is visualized. The breakdown is represented by PDM-orientated explicit parent-child relationships and applicability is realized by relationship to the Product variant applicability object within the Product breakdown, refer to . The same principle can be used when a traditional Product breakdown approach is realized as shown in [Fig 12](#).

Note

In comparison to the traditional approach by MIL-STD 1388-2B the function of the data elements ALC and UOC is now realized by applicability and hardware or software realization.



ICN-B6865-S3000L0017-003-01

Fig 12 Usage of BEI and applicability

The example shows two different contracted variants of the Product, in this case a fuel truck. The main difference is the installation of different engines. For the truck variant represented by the Product variant identifier A00, a gas engine represented by the BEI A200 is installed, for the truck variant represented by the Product variant identifier A01, a diesel engine is installed. The breakdown elements *Body* and *Transmission* are applicable for both truck variants. The usage of applicability as represented in Fig 12 easily enables a complete representation of each Product variant, also refer to Table 6.

For the representation of different variants of components/parts, applicability is realized by hardware (or software) element realization. The three different fuel pumps which can be installed into the gas engine are variants of the fuel pump, represented by the BE of the fuel pump with the BEI A20300.

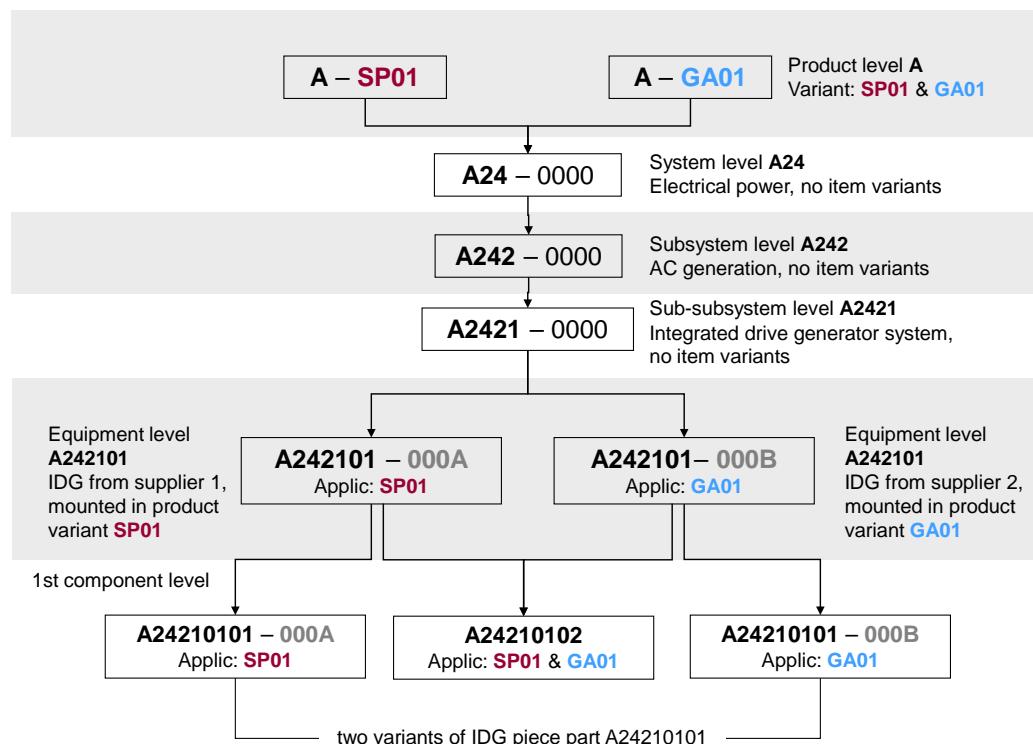
Note

Each BE and each BE realization by concrete hardware or software parts can be related to the corresponding Product variant. This ensures the documentation of each Product variant with all components which are installed. The relation to the Product variant can be established from each indenture level of the breakdown. Physical and functional BEs can be related as well as parts and its components down to the piece parts, if required. In a Product breakdown, it can be sufficient (refer to Fig 12) to relate only the gas engine to the corresponding Product variant. In this case, all items below the BE of the gas engine are also related to the corresponding Product variant (heritage method).

Table 6 Examples of usage of BEI and applicability from Fig 12

Product variant ID	BEI	PNR	Name	Applicability
A00	-	-	Truck model A	Product variant
A01	-	-	Truck model B	Product variant
	A200	GE01	Gas engine	to A00
	A201	DE01	Diesel engine	to A01
	A100	-	Body	to A00 and A01
	A300	-	Transmission	to A00 and A01
	A20300	-	Fuel pump	to A00
		FP-G1, FP-G2, FP-G3	Fuel pump	to A00
	A20301	-	Fuel pump	to A01
		FP-D1	Fuel pump	to A01

The next example for a traditional Product breakdown also shows the principle of assigning applicability. The breakdown is represented by traditional BEI syntax for systems, subsystem, and equipment relationships. Applicability is realized by relationship to the Product variant applicability object within the Product breakdown, refer to [Fig 13](#).



ICN-B6865-S3000L0100-001-01

Fig 13 Usage of BEI and applicability for traditional breakdown

It is strongly recommended that the identification of variants (both, Product/end item and its components/parts) and the usage of applicability is clarified between contractor and customer at the LSA GC.

3.1.6.3 Breakdown element type

In the S3000L Product breakdown philosophy there are BEs of different types. These are represented by specializations of a BE (refer to [Chap 19](#)):

- BE realized by a hardware part or a software part
- BE realized by a zone
- BE realized by an aggregated BE (eg for the documentation of systems, subsystems, item families)

3.1.6.4 Equipment type

Other important information which can be assigned to a BE are the aspects of replaceability and reparability of the corresponding item. This includes discard information as well as accessibility information. In [Table 7](#) and [Table 8](#), a number of general terms are introduced as a guideline to how corresponding terminology can be used within a project. However, it is recommended that a common understanding of typical logistic terms eg within a glossary, is agreed at the LSA GC.

Table 7 Aspect of replaceability

Terminology	Description
Directly replaceable	Item that is directly replaceable on the Product level (an LRU as already described in Para 3.1.1.6 is a typical item, which is directly replaceable).
Not directly replaceable	Item that is not directly replaceable on the Product level (installed in a superior item). The superior item must be removed before replacement, eg the compressor of an engine can only be replaced after engine removal (an SRU as described in Para 3.1.1.6 is a typical item, which is not directly replaceable).
Non replaceable	Item that is intrinsically tied to another and cannot be replaced separately.

Additional to the replaceability it is necessary to know whether an equipment or a component can be repaired or not. This aspect is summarized in the following table.

Table 8 Aspects of reparability

Terminology	Description
Full repairable	Item that is fully repairable.
Partial repairable	Item that is partial repairable (depends on failure).
Non repairable (discard part)	Item that is not repairable.

Normally, the reparability information is related to a component/part and will be documented against a realization (eg part). In some cases it can be an attribute of a BE, when the installation situation prevents a repair of the installed component.

The combination of replaceability and reparability fully describe the equipment within the context of the decision on whether an item will be replaced and/or repaired.

3.1.7

Breakdown element identifier requirements concerning software

In general, it is possible to include SW items within a Product breakdown in the same way as hardware items. However, software as a part of a Product breakdown need some special attention. Refer to [Para 3.2.5](#), [Para 3.2.6](#) and [Para 3.2.7](#) for examples concerning SW elements in physical and functional Product breakdowns.

3.2

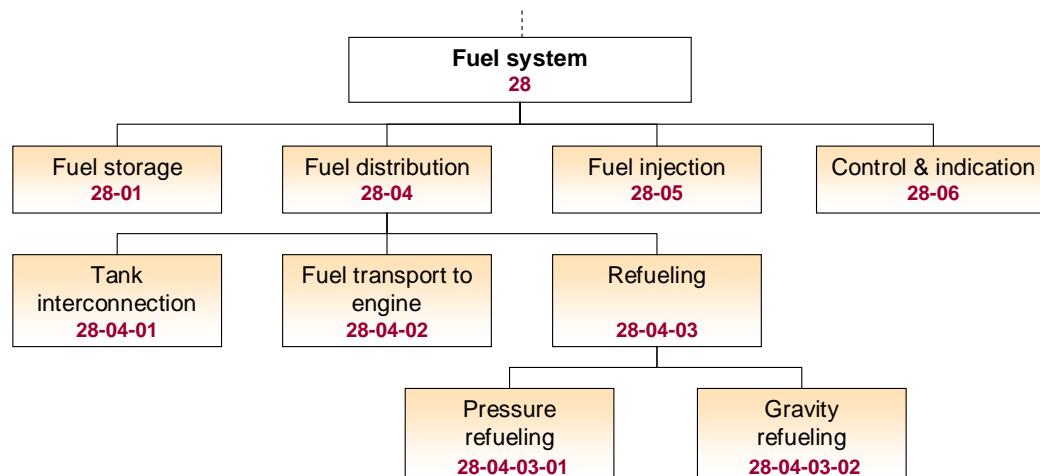
Breakdown element identifier - structure examples

The following examples show how an effective Product breakdown can be established. For this purpose, examples from [Para 3.1.2](#) are used and filled with the missing BEIs for each breakdown level and for each breakdown element. Additionally, examples of how to integrate software items are given.

3.2.1

Functional breakdown

The first example (refer to [Fig 14](#) and [Table 9](#)) shows a functional breakdown following a simple BEI syntax. The top level item (root) can be a Product containing a fuel system.



ICN-B6865-S3000L0027-001-01

Fig 14 Functional breakdown, simple syntax with separators

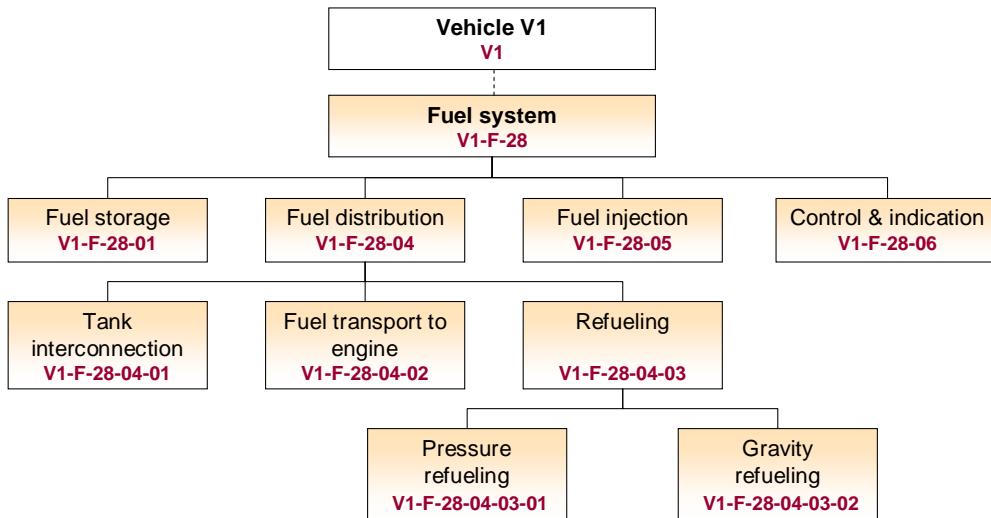
Table 9 Listing of functional breakdown, simple syntax with separators

BEI	Item name	Item type
28	Fuel system	Main system
28-01	Fuel storage	Function
28-04	Fuel distribution	Function
28-04-01	Tank interconnection	Subfunction
28-04-02	Fuel transport to engine	Subfunction
28-04-03	Refueling	Subfunction
28-04-03-01	Pressure refueling	Sub subfunction
28-04-03-02	Gravity refueling	Sub subfunction
28-05	Fuel injection	Function
28-06	Control & indication	Function

The second example (refer to [Fig 15](#) and [Table 10](#)) shows a functional breakdown with an extended BEI syntax structure. The top level item (root) can be a Product containing a fuel system (Vehicle V1). The first BE is the fuel system itself. The BEI includes information about the Product (V1) and about the breakdown type (F=functional). In this case, the fuel system is the first indenture level of the breakdown tree.

Note

The Product itself can be considered as the typical “root” BEI at an indenture level zero. The next level down from the root is the first Product breakdown level.



ICN-B6865-S3000L0028-001-01

Fig 15 Functional breakdown, extended syntax with separators

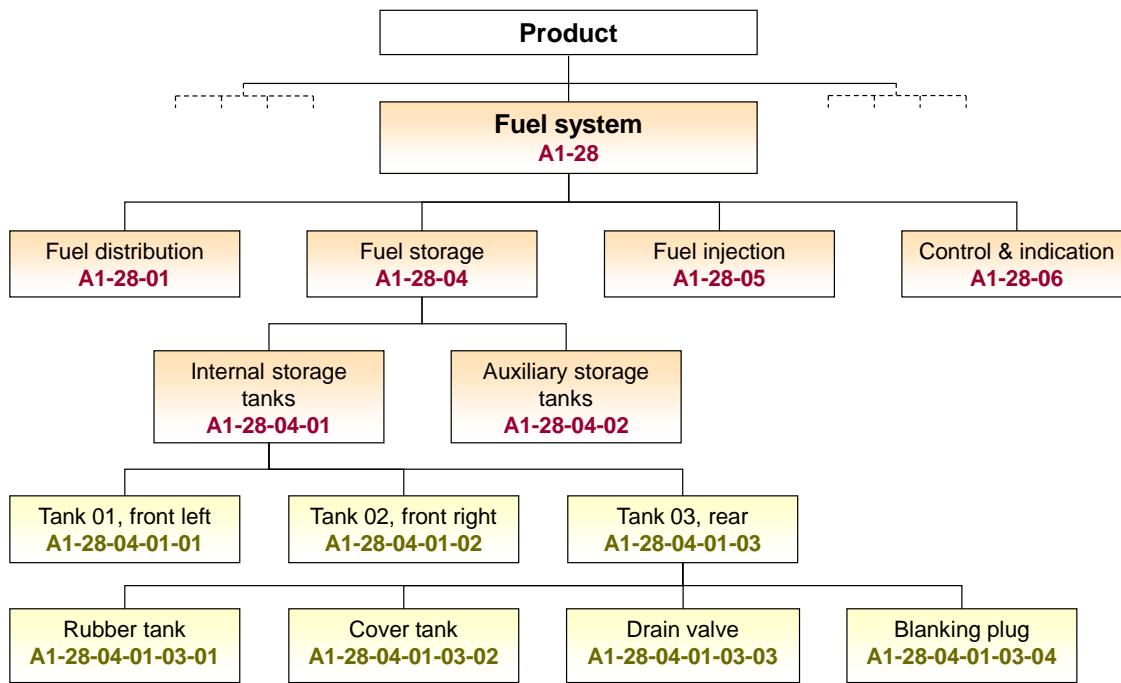
Table 10 Listing of functional breakdown, extended syntax with separators

BEI	Item name	Item type
V1	Vehicle V1	Product
V1-F-28	Fuel system	Main system
V1-F-28-01	Fuel storage	Function
V1-F-28-04	Fuel distribution	Function
V1-F-28-04-01	Tank interconnection	Subfunction
V1-F-28-04-02	Fuel transport to engine	Subfunction
V1-F-28-04-03	Refueling	Subfunction
V1-F-28-04-03-01	Pressure refueling	Sub subfunction
V1-F-28-04-03-02	Gravity refueling	Sub subfunction
V1-F-28-05	Fuel injection	Function
V1-F-28-06	Control & indication	Function

3.2.2

Mixture of functional and physical breakdown

The next example (refer to [Fig 16](#) and [Table 11](#)) shows a typical mixture of functional and physical breakdown. Within this breakdown, starting from the root level, the first breakdown levels are used for functional/system information. From a certain depth of breakdown downwards you will find real physical equipment which are realized by concrete parts. The functional information will be fixed inside the breakdown. This means that a functional system can be identified by the first breakdown level of the BEI, a functional subsystem can be identified by the second breakdown level and so on. Within complex Products, systems can be of different size. For that reason normally an indenture level where functional breakdown ends and physical breakdown starts cannot be fixed. It is recommended to use existing numbering systems (eg S1000D SNS) for an adequate structuring of the Product under analysis.



ICN-B6865-S3000L0029-002-01

Fig 16 Mixed breakdown, extended syntax with separators

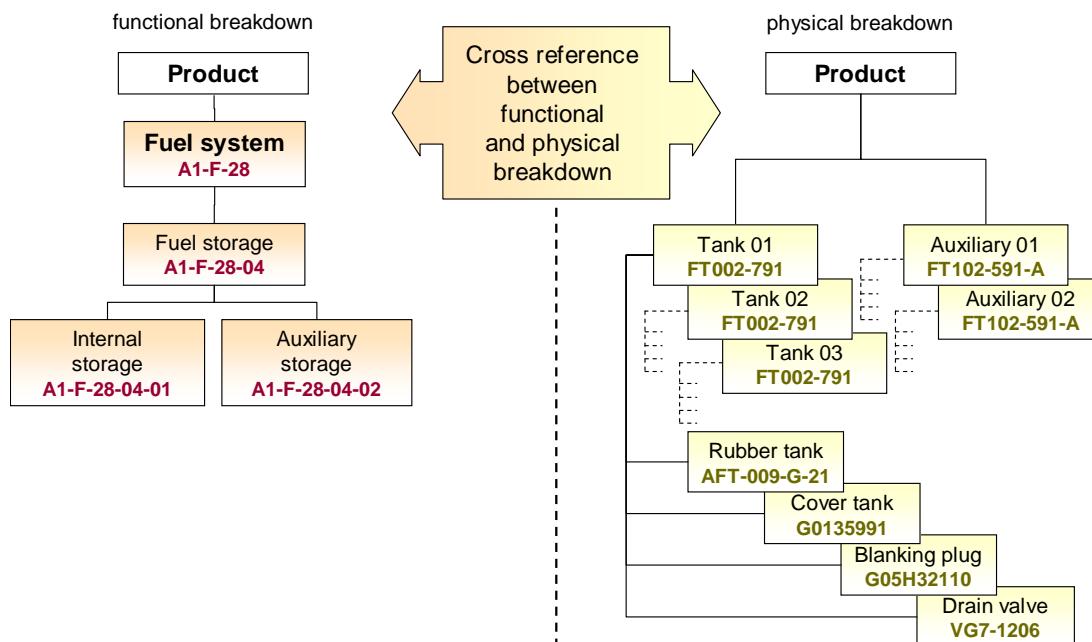
Table 11 Listing of mixed breakdown, extended syntax with separators

BEI	Item name	Item type
A1	Aircraft A1	Product
A1-28	Fuel system	Main system
A1-28-01	Fuel distribution	Function/subsystem
A1-28-04	Fuel storage	Function/subsystem
A1-28-04-01	Internal storage tanks	Subfunction/assembly
.....		
A1-28-04-01-03	Tank 03 rear	Equipment
A1-28-04-01-03-01	Rubber tank	Component/part
A1-28-04-01-03-02	Cover tank	Component /part
A1-28-04-01-03-03	Drain valve	Component /part
A1-28-04-01-03-04	Blanking plug	Component /part
.....		
A1-28-04-02	Auxiliary storage tanks	Subfunction/assembly
A1-28-05	Fuel injection	Function/subsystem
A1-28-06	Control & indication	Function/subsystem

As shown in Fig 16, sometimes the border between a functional BE and a physical BE is fluid. For example the BE A1-28-04-01 representing the internal storage tanks can be a functional BE, covering the function internal fuel storage, or can be an assembly of physical components, in this case, the three internal tanks.

3.2.3

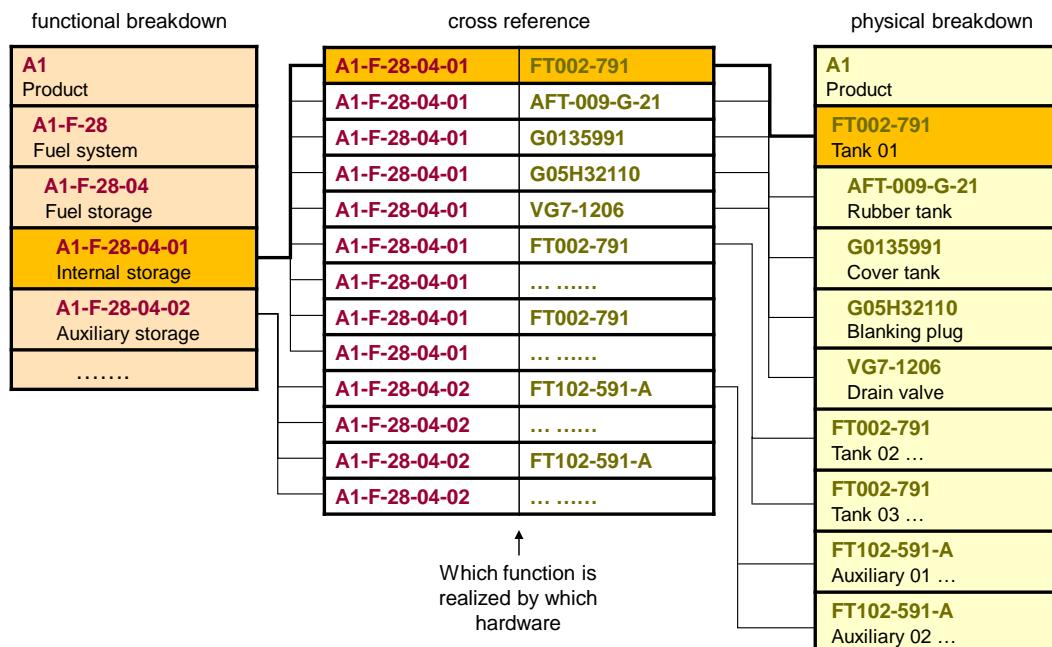
Separation of physical and functional breakdown with cross referencing (hardware only)
 The example in Fig 17 shows the usage of totally separate breakdown trees for both functional and physical breakdowns. The problem of finding an effective connection between these two areas is solved within an LSA database by the means of appropriate cross reference tables.



ICN-B6865-S3000L0030-002-01

Fig 17 Parallel usage of functional and physical breakdown

The cross referencing between separate breakdowns is realized by an $m:n$ relationship structure. On the one hand, one function can be connected to many (m) BEs from physical breakdown, on the other hand one BE can be connected to more than one (n) BEs from functional breakdown. This kind of relation is called an $m:n$ relationship between functional and physical BEs. Refer to [Fig 18](#).



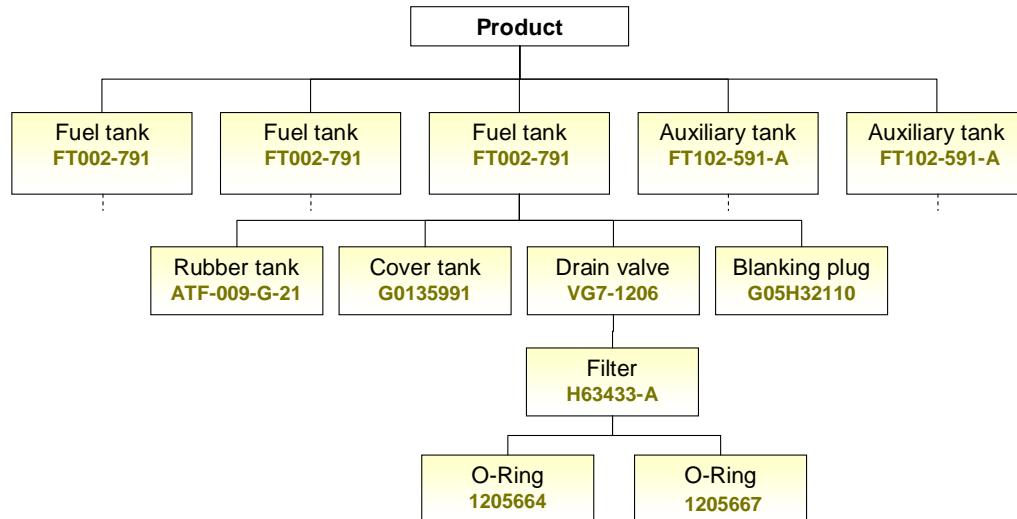
ICN-B6865-S3000L0031-003-01

Fig 18 Cross referencing between functional and physical breakdown

3.2.4

Physical breakdown with explicit parent-child relationship

Another philosophy of breakdown is the usage of a parent-child methodology (refer to [Fig 19](#) and [Table 12](#)). In this case, the hardware is organized in a hierarchical way concerning the installation of the equipment or components in the corresponding parent breakdown element. This type of breakdown is used, for example, in PDM systems. Additional information in form of a BEI can also be given. The functional entity (as given in the examples before by using the "28" from an SNS for fuel system) is not automatically given in this type of breakdown methodology.



ICN-B6865-S3000L0032-001-01

Fig 19 Breakdown with parent-child methodology

In this type of breakdown the BE key information can be given, for example by a part number which identifies the BE. It does not contain any information concerning the functional entity. The interrelation between the BEs is not implicitly defined by the key data element part number. Using this type of breakdown methodology requires the information about the next higher BE and quantity information as shown in

Table 12 Listing of breakdown with parent-child methodology

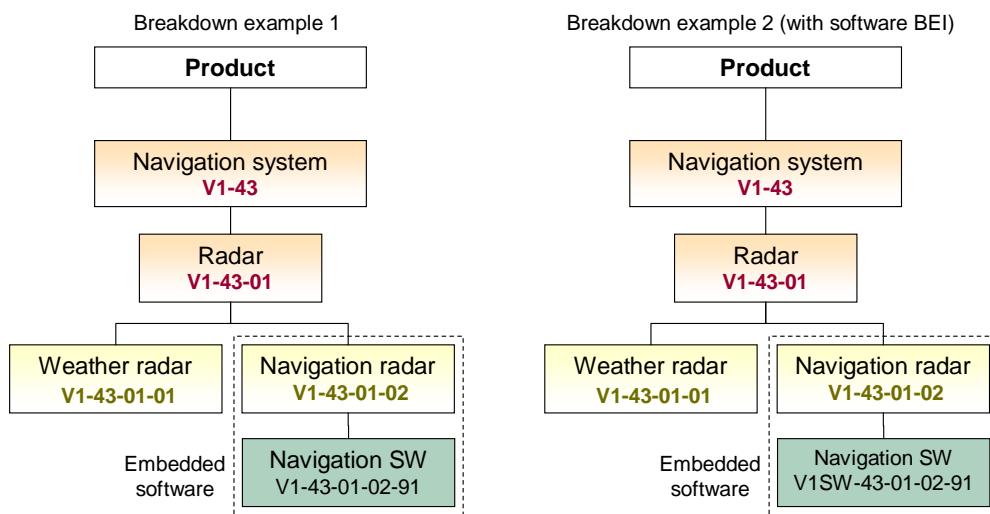
Item number	Item name	Item type	Next higher item	Quantity
V1	Vehicle V1	Product	-	-
FT002-791	Fuel tank	Assembly	V1	3
FT102-591-A	Auxiliary tank	Assembly	V1	2
ATF-009-G-21	Rubber tank	Component	FT002-791	1
G0135991	Cover tank	Component	FT002-791	1
VG7-1206	Drain valve	Component	FT002-791	1
G05H32110	Blanking plug	Component	FT002-791	1
H63433-A	Filter	Component	VG7-1206	1
1205664	O-Ring	Component	H63433-A	1
1205667	O-Ring	Component	H63433-A	1

Within an explicit breakdown, it is also possible to document multiple installations of components. If there is a need to be explicit about each installation location, each installation location must be documented as a separate BE with quantity 1.

3.2.5

Integration of software as part of a single physical hardware item

A software package can be assigned directly to a hardware. The assignment of a BEI for the software package can be done directly below the hardware according to the established breakdown coding rules. The coding rules can include specific digits to identify the BE as software. In Breakdown example 1 of [Fig 20](#), the BEI of the Navigation radar and of the Navigation SW package embedded in the Navigation radar cannot be separated only based on BEI information. By implicit relation with a BEI syntax, the Navigation SW package can be assigned uniquely to the Navigation radar as the carrier of the software.



ICN-B6865-S3000L0033-002-01

Fig 20 Assigning of software BEI in a physical breakdown

In breakdown example 2 of [Fig 20](#), the same breakdown is shown, but with a specific software BEI. It is marked using the two digits "SW" within the BEI. The other breakdown levels are not affected, and the assignment of the SW package as a part of the Navigation radar is unique, too.

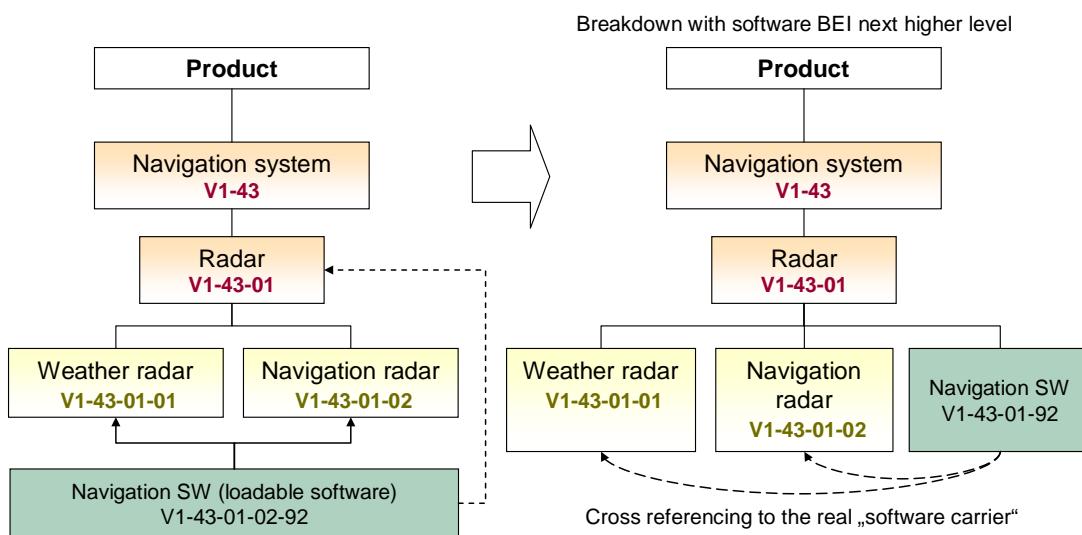
Note

Normally, hardware or software parts are related to the BEs. This also identifies BEs being a software or hardware element of the Product breakdown. In this case, a specific coding of BEI is not required.

3.2.6

Integration of software as part of more than one physical hardware item

If the installation or loading of SW distributes a software package to more than one physical device in the hardware breakdown, a unique assignment of SW to hardware is not possible. An alternative method to assign the SW can be to shift the SW component to a higher breakdown level. Refer to [Fig 21](#).



ICN-B6865-S3000L0034-002-01

Fig 21 Distributed software in a physical breakdown

The loadable radar SW can be loaded in one procedure to two different equipment. The SW cannot be divided up physically into two parts, because it is delivered and installed as one single object. The dividing is done by the loading procedure, a unique assignment is not possible.

In this case, it is possible to address the SW component one level higher to the BEI of the complete radar system. To avoid the loss of information, to which components the SW really belongs, it is necessary to establish a cross reference assigning the BEI of the SW package to both the Navigation radar and to the Weather radar.

3.2.7

Integration of software in a functional breakdown

In a functional structure, SW items are elements of the functional system or breakdown element within which they operate. In allocating a BEI, it is necessary to consider the supportability characteristics of the SW items. Items with particular support requirements are allocated to their own BEI. For example, the BEI scheme must distinguish between SW that is loadable and SW that is embedded. It can also be necessary to distinguish between SW that has been provided by the prime contractor or SW items that are safety or security significant. During the early stages of a project these supportability issues are often not fully considered. The development of a functional BEI allocation scheme can be an iterative process.

In order to develop a functional BEI structure in which SW items are logically grouped at an appropriate level of indenture, and in which the differing support requirements of all software items are exposed, items will often be included at indenture levels against which only limited amounts of information needs to be stored. In the example of [Fig 22](#), little LSA data will be stored in respect to items *Fourier* and *Signature*. The full range of LSA information will be recorded against the items at the lowest indenture level. In such case, SW is represented by the items on which the software maintenance activities are carried out, also refer to [Chap 13](#).

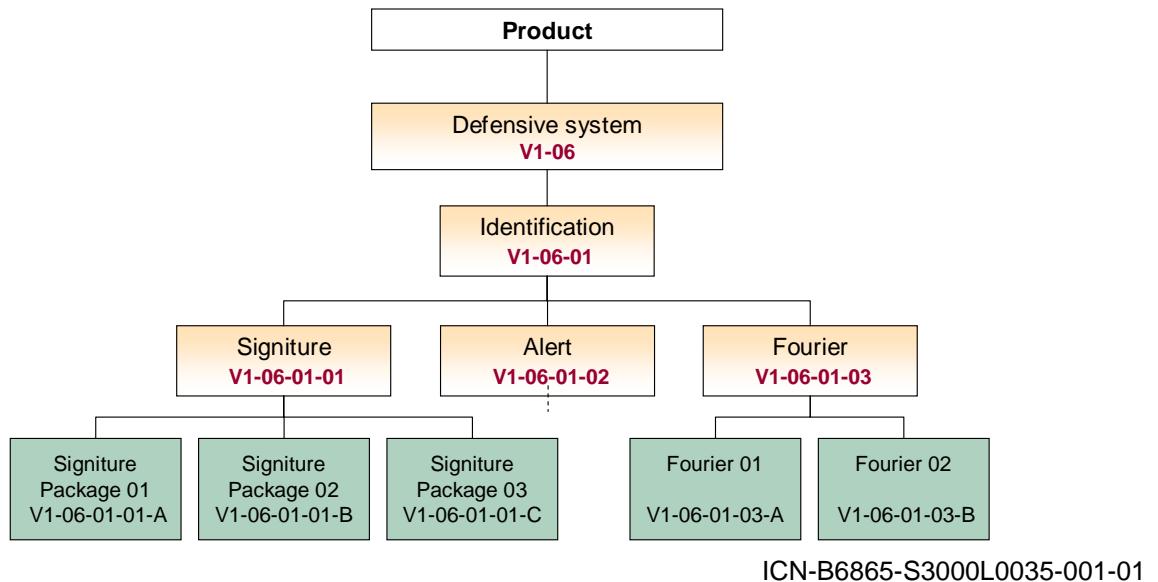


Fig 22 Functional BEI structure concerning SW

It is necessary to maintain both functional and physical views of SW items throughout the life of the parent Product. It can be important to establish links between software items in the functional domain and the associated executable SW package in the physical domain. This can be achieved by the same method of cross mapping the physical hardware items to functional Product breakdown.

The requirements concerning the assignment of SW BEI in a pure functional breakdown are the same as in physical breakdowns. If there are functions that are realized by exactly one definite SW package, the unique assignment can be possible. If more than one function is realized by the loading of an SW package, the need for a higher level assignment is the same as for a physical breakdown.

An additional question concerning SW is when to use a functional breakdown and when to use a physical breakdown. In many cases, SW can be assigned within a physical breakdown on a subsystem or system level (this can be an additional argument to use a mixed breakdown methodology, physical and functional). However, for SW modification purposes, a functional breakdown within a separated SSA database can be preferred. Normally, a mixed breakdown structure including the higher breakdown hierarchies on system/subsystem level as a kind of functional framework is not sufficient for a real functional breakdown.

4

Configuration changes in LSA

4.1

Evaluation of the impact of configuration changes

For proper CM, it is necessary to correctly evaluate all impacts that changes can introduce into LSA. The changes can have an impact on configuration items, on the documentation, or on the support system. In order to control the impact of changes on operating and support issues, the following will be taken into account:

- The effect of the change on the LSA disciplines
- The consequential effects of the change on support elements, (eg spares, test equipment)
- Interchangeability/compatibility of all post- and pre-change items
- If the change has applicability, this will be indicated in the change documentation

LSA and the other ILS disciplines must analyze the impact of the change to identify the necessary activities to be implemented.

Because LSA is the trigger for the ILS disciplines, it also has a coordinating role as focal point of ILS at the program modification committee and to coordinate the jobs between different disciplines. This is necessary to ensure that any approved change is reflected in ILS.

4.2

Sources/drivers for configuration changes

LSA involvement on CM process is required because there are changes to the baseline configuration of a Product. These changes can be internal changes (eg improvement of the support analysis) or external changes (eg design changes, supplier changes).

Potential drivers/sources for configuration changes can be:

- Requirements from the customer
- Changes from the design office
- Manufacturing deviation from design (concessions)
- Supplier changes
- Customer orders for change (in-service changes)

Since specific drivers are linked to specific sources, the management of the relevant sources is required for CM. Specific changes normally will follow defined processes and will require corresponding activities. LSA has to be involved in the change processes to identify LSA activities as result of changes to ensure a correct LSA configuration for the Product. The most important business cases are outlined below.

4.2.1

Customer requirements for change

In some cases, the customer requires changes to a Product. The process to be followed is:

- Reception of Customer Originated Change (COC) proposal
- COC management
 - Analysis of impact with cost implications. The request for change from the customer must be analyzed to determine the impact of the change proposal and the cost to implement the change.
 - Change committee actions. The change can be submitted to the modification committee for approval or rejection.
 - Decision and Resolution notification. The decision on the change proposal must be communicated to the customer.
- COC design validation
 - Requirement for design office to make an analysis. The change proposal is taken to the design office for analysis.
 - Design change proposal associated. When the design office decides to create a design modification, a link between the customer change proposal and the design change must be established and recorded for traceability purposes.
 - Design office documentation collection. The design office prepares the documentation that needs to be collected in order to make the final change proposal derive from the customer requirement.

4.2.2

Design change proposals

The design office can propose changes to the configuration for several reasons. Depending on their origin, changes can be justified in three groups:

- Mandatory changes (safety): This type of change is unavoidable because the safety of the Product is involved. The consequences of not implementing these changes can lead to catastrophic situations in the operation of the Product.
- Improvement changes: Change proposal that improves the functionality or supportability of the Product.

- Version definition changes. A change introduced to create new customer versions, in accordance with the customer contractual requirements.

In order to maintain the traceability of changes, it is necessary to consider certain aspects related to these changes and the subsequent actions that are required, such as:

- Change applicability. Provides the applicability of the change in terms of the Product.
- Design configuration concepts. In general terms the design office uses the concept of configuration item for a functional item. A functional item is a requirement to comply with a specific function of the Product. Since there is not just one way of doing this, it is possible to have different Design Solutions (DS) for the same function. Therefore, there are several DSs associated to one configuration item. A DS can be equipment, a set of equipment, or an assembly of different items that together provide functionality.

The design changes:

- Create configuration items
- Modify configuration items, creating, modifying or deleting new DSs associated to the configuration item
- Drawing and mock-up changes. In some cases, the change does not introduce a new CI/DS, but changes the position of the items (mock-up changes). These changes can also affect the support items (such as maintenance task).
- Changes to design specification and systems descriptions. The functional description of the Product is included in design specifications. Therefore, changes to the configuration of the Product imply change to its functionality, so the design specifications are affected by the design changes. Because design specifications are used in LSA for systems, LSA is impacted by these changes.
- Design change status. In order to authorize/implement a design change, an approval process is to be followed. It is important from the LSA point of view to know this status and to act accordingly. As LSA is an integral part of the entire process, the global status of the change must take into account the LSA status. Therefore, the LSA provides feedback to the program change management.

4.2.3

Manufacturing deviation from design (concessions)

Where current approved configuration documentation is correct and manufacturing errors occur, a form for "deviations and waivers" is completed by manufacturing, either to enable the item to be accepted, or for some kind of repair to be made in order to make the item conform to the drawing or standards, rather than be rejected. It is recommended not to use deviations and waivers to avoid processes required by the life cycle. If non-applicable documents and evaluations are required, these must be tailored and agreed upon prior to contract award.

- Requirements for deviation

Prior to the manufacture of an item, if it is considered necessary to temporarily depart from mandatory requirements of a specification or drawings, for a specific number of units or a specified period of time, the customer can authorize a request from the contractor. Items must not be delivered incorporating a known departure from documentation unless a request for deviation has been approved.

- Requirements for waiver

Supplies or services, which do not conform in all respects to the contractual requirements, shall normally be rejected. An item, which through error during manufacture does not conform to the specified configuration documentation, shall not be delivered to the customer unless a waiver has been processed and granted.

4.2.4 Supplier changes

Any change in the equipment provided by any supplier affects the configuration of the Product. The supplier change affects the equipment or the support items applicable to the supplier equipment.

4.2.5 In-service changes (service bulletins)

When a modification must be incorporated once the Product is operating, normally it is introduced through a Service Bulletin (SB).

A SB is a technical document that describes the modification and how it is to be implemented. It has an associated applicability, and once the SB is approved, it can be treated as a design modification.

The difference between a design modification and an SB is that the contractor does not always have information about the implementation of the SB by the customer. Therefore, the applicability of the SB cannot be considered as a true applicability.

4.3 Traceability between the source of changes and consequential LSA changes

The traceability between sources changes and LSA changes is achieved by:

- Change proposal criterion traceability

The change traceability is guaranteed by using as a criterion (trigger) for change the same code that comes from the design office, or another source, as a change proposal. If this is not the case, it is necessary to maintain a record to relate the criterion used to authorize the changes to LSA/ILS with the original changes.

- Design configuration items versus LSA configuration Items

Since the design office can use different codes for design configuration than the ones used in LSA configuration, it is necessary to establish a link between the two ways of identifying an item of the configuration generating the necessary rules to guarantee traceability between design configuration and LSA configuration, refer to [Para 2.1](#).

4.4 Processing of configuration changes during the LSA process

4.4.1 General aspects

Once it is decided that a change proposal has an effect on LSA and becomes a criterion (trigger), it is necessary to initiate consequential activities in order to ensure that:

- The impact on LSA is properly identified
- The different jobs to implement the necessary changes are identified, assigned planned in time and the necessary resources to implement the change are identified
- The status of the jobs performed by LSA can be known and feedback to the entire configuration management process at the program level about the status of the change implementation is guaranteed
- Any change proposal identified as non-affecting to LSA needs to be recorded with attached justification. This is required to ensure that the entire change proposal has been analyzed, which guarantees the traceability of changes.

4.4.2 LSA compilation by change

Compilation by change means that any data compiled has to be done in accordance with a change code that authorizes it and also that change is considered as the driver for the compilation. There are some basic reasons to compile by change:

- To ensure the traceability of changes produced in design, manufacturing engineering, supplier, or change requirements from the customer. This identifies the real impact of the approved changes.
- To have the authority of any piece of support system changed. This allows the justification of any change.
- To identify the applicability of the items (to be discussed if the part dedicated to an applicability).

The basic way to do it is to identify the impact of any change in any item of the operational and/or support systems. Identification of the item affected by the change and the type of affection is recorded for quality assurance purposes. Potential types of affection are listed in [Table 13](#):

Table 13 Types of affection

Type of affection	Description
New item	The change creates a new item or extends the applicability of one existing item. The item is new for the range of operational system Products indicated by the applicability of the
Deleted item	Most of the time it is not a real deletion, it is only a way to say that the existing item is not any more applicable to the Products covered in the range indicated by the change applicability. This information is very important for getting the configuration item applicability.
Modified item	Used when only traceability function is required for the change, and non-applicability impact is declared. For example a task is changed by a criterion but the task is still applicable to the same range of MSN that was previously to the application of the criterion. What is affected and has different applicability are the components of the task (eg, spares, step description)

The LSA operational system configuration is linked to the design configuration of the operational system through approved design modifications that have impact in ILS/LSA. This modification must be recorded and associated to the breakdown revision identifier, specifying whether to create a new breakdown element, deleting existing ones (eg in the way of restricting applicability of the breakdown element) or just extending the range of applicability.

Maintaining the relation between the breakdown element revision, hardware element realization and Product variant realization applicability and the original design configuration identification provides the means to guarantee the traceability of the LSA configuration. It also provides help to identify BEIs potential affected by new design changes.

4.4.3 Extended applicability management

4.4.3.1 General

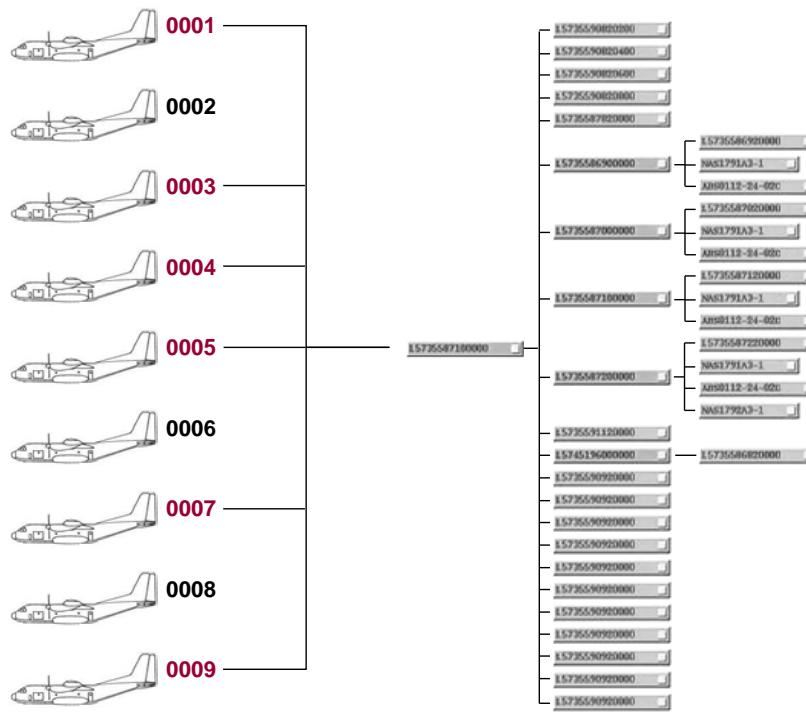
The configuration of a Product is not unique and every Product can have different components. Therefore it is required to know the configuration of every Product or, the other way around, for every item of the configuration the Products where it is assembled. This is provided by means of an applicability management.

4.4.3.2 Manufactured serial number and fleet serial number concepts

From the customer point of view, operation or service are the targets for a system. In the end, the operation is complied by a single Product. A Manufactured Serial Number (MSN) identifies this single Product. It is a unique number to identify the Product. It is mandatory that every



Product has an MSN. Therefore, the range of MSN where it is mounted mainly identifies the applicability of an item. Refer to [Fig 23](#).

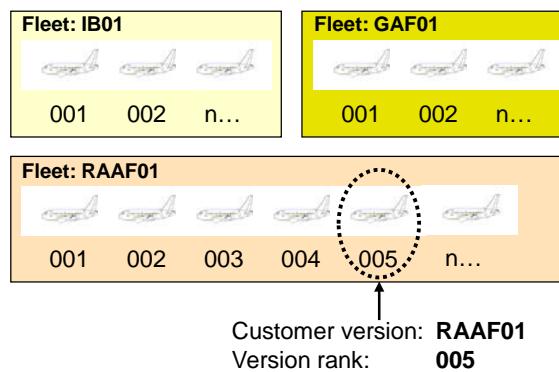


The component is installed on aircrafts: 0001, 0003, 0004, 0005, 0007, 0009
Its applicability is: **0001, 0003-0005, 0007, 0009**

ICN-B6865-S3000L0087-001-01

Fig 23 MSN usage example

The Fleet Serial Number (FSN) is another possibility. The concept of fleet is based in the combination of two concepts: customer and version. Version is a set of Products with basically the same configuration because they are thought to comply with the same type of missions or with the same operational characteristics. Minor differences can occur between the Products belonging to a version due mainly to evolutions or improvements. The conjunction of a Product for a customer is defined as the fleet. A Product is identified by customer + version + serial number inside version. Refer to Fig 24.



ICN-B6865-S3000L0097-002-01

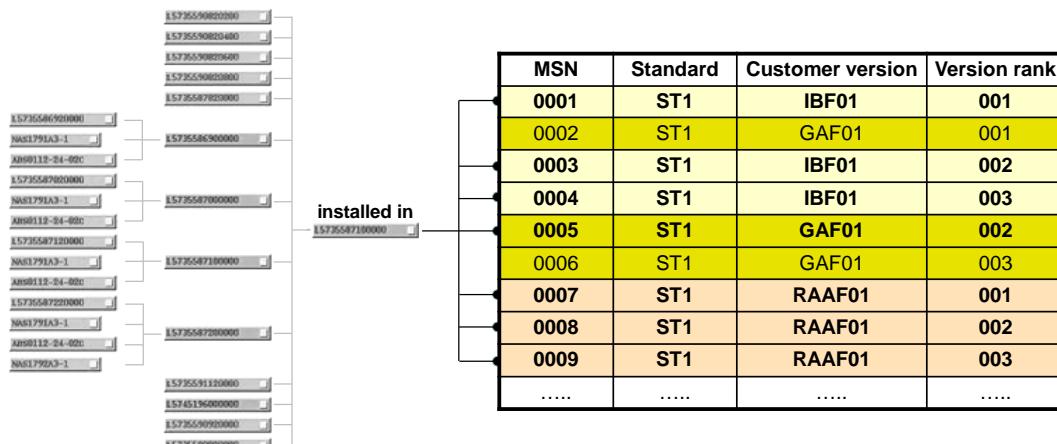
Fig 24 Fleet example

There must be a correspondence between MSN and FSN. Refer to Fig 25.

Applicable to: All

S3000L-A-04-00-0000-00A-040A-A

Chap 4



ICN-B6865-S3000L0098-003-01

Fig 25 FSN (fleet) versus MSN

S2000M defines the applicability by fleet numbering, providing the service (code to identify the customer) version (eg single seat aircraft, twin seat aircraft) and serial number inside the version.

Due to the planning for manufacturing, normally the numbering for MSN and FSN are not coincident. The first MSN is for one customer then the second MSN is for other customer and so on.

4.4.3.3 Effectivity of change/criterion

A change proposal can be applicable to all Products or can be limited to one or more range of Products. This range is defined by two figures:

- MSN from

Identify the first MSN where the change is applicable. This item is included.

- MSN to

Identify the last MSN where the change is applicable. This item is also considered included.

Other possibility of expressing the applicability of change/criterion is providing the version and then the range inside the version. It is possible that a change/criterion apply to more than one version/range.

4.4.3.3.1 Example: Design criteria applicability to LSA applicability

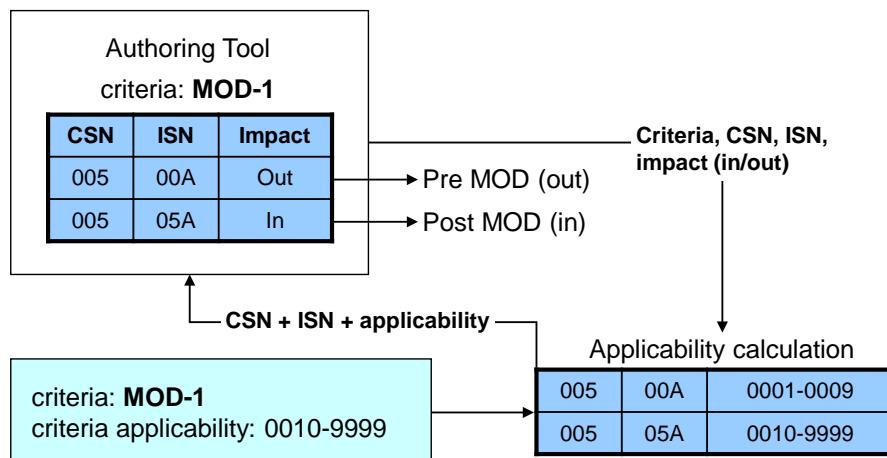
In the traceability management process it has been described how to indicate the affectation a criterion has to a configured item. The following example shows how to convert this into applicability ranges for an item.

- First step: The first criterion called criterion 1 is defined having the applicability by MSN from 0001 to 9999.
- It is used to create a new BER of A212101 and for a new hardware element realization of P/N1. Criterion 1 makes applicable ("in") the range of serial numbers.
- For that affectation BER of A212101 has an applicability from 0001 to 9999.
- A new criterion with an applicability 0020-9999 creates a variant for the existing BER with a new hardware element realization of P/N2 and the same Product variant with a new block of serialized items.
- As a result of it :

- breakdown element revision of A212101 and hardware element realization of P/N1 is only applicable from 0001 to 0019
- breakdown element revision of A212101 and hardware element realization of P/N2 is only applicable from 0020 to 9999

4.4.3.3.2 Example: Design criteria applicability to IP/IPD configuration item (CSN/ISN) applicability

An example of applicability calculation for IP/IPD items based on CSN and ISN is given in [Fig 26](#).



ICN-B6865-S3000L0099-001-01

Fig 26 Applicability calculation

4.4.3.4 MSN versus FSN applicability management

MSN or FSN are used when the Product or element with a change in its configuration does not change its P/N. In this case the MSN or FSN is used to distinguish between the different configured end Items. For example where the P/N of the aircraft is not changed, the MSN or FSN is required to identify the different aircraft manufactured. In the case of an equipment where the configuration of the equipment is changed, the P/N of the equipment is also changed. Then the UOC is used to identify the applicability of the component of every equipment identified by different P/N.

MSN applicability is oriented to the manufacturing. It is a number that does not change during its life. Normally the applicability in design is given by this way so criteria coming from design will usually have this type of applicability.

For FSN applicability, it is recommended that the rules and definition of S2000M are used. It is recommended that the use this type of applicability is utilized because it is easy to customize the information that is to be delivered.

Chapter 5

Influence on design

Table of contents

	Page
Influence on design	1
References.....	2
1 General	2
1.1 Introduction	2
1.2 Objective	2
1.3 Scope.....	3
2 Design considerations	3
2.1 Availability.....	4
2.2 Reliability	5
2.3 Maintainability	5
2.4 Testability.....	5
2.5 Prognostics	6
2.6 Standardization.....	6
2.7 Interchangeability.....	7
2.8 Environmental considerations	7
2.9 Human factors/ergonomics.....	7
2.10 Obsolescence	7
2.11 Supportability	7
2.12 Cost effectiveness	8
2.13 Software design	8
3 Development programs	8
3.1 Strategy for LSA influence on design	8
3.2 Design influence on development programs	9
3.3 Supplier design	9
3.4 Critical/milestone design reviews	9
4 Checklists	10
4.1 Selection of parts/equipment.....	10
4.2 Reliability	10
4.3 Maintainability	11
4.4 Testability.....	11
4.5 Prognostics	11
4.6 Standardization.....	11
4.7 Interchangeability.....	11
4.8 Environment.....	12
4.9 Human factors/ergonomics/accessibility	12
4.10 Obsolescence	12

List of tables

1 References	2
-------------------------	---

List of figures

1 Opportunity to influence design and support cost during Product lifetime	3
2 Breakdown of availability as an approach to structure design influence.....	5
3 Effectiveness - balancing availability and LCC.....	8

References

Table 1 References

Chap No./Document No.	Title
<u>Chap 7</u>	Results of FMEA/FMECA in LSA
<u>Chap 8</u>	Damage and special event analysis
<u>Chap 13</u>	Software support analysis
<u>Chap 17</u>	Disposal

1 General

1.1 Introduction

This chapter covers the subject of LSA parameters influencing the design of the Product.

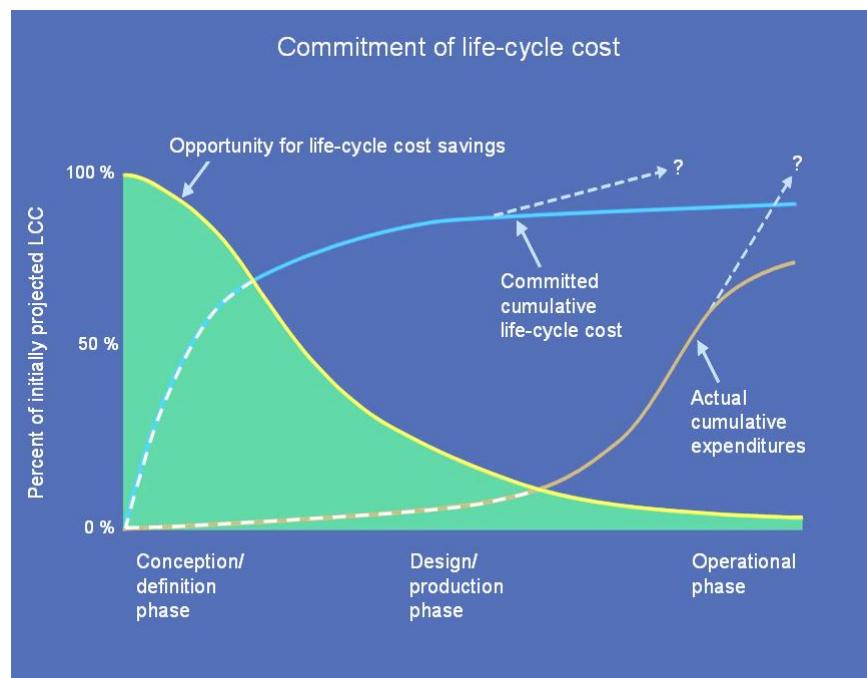
Influence on design is one of the main goals of staff involved in supportability engineering and in the LSA process. For staff involved in design (software or hardware), influence on design is important in order to interact between the Product design program and LSA program.

1.2 Objective

The broad objectives of LSA are to influence design, create the most effective support concept, and define logistic support resource requirements. In this chapter, the focus is put primarily on the influence of the Product design. However, it is not possible to isolate it from the iterative work needed for an effective support solution or from requirements on logistic support resources.

General objectives for the Product must be translated into more specific requirements for an individual project. The key to a productive and cost effective LSA is the concentration of available resources on activities which most benefit the program. Such concentration might be called the analysis strategy. When deciding on an analysis strategy, it is necessary to consider the type and scope of the development project. The possibilities of influence can also vary due to previous design decisions and the life cycle status.

The opportunity for influencing design in order to fulfill LSA requirements and reduce LCC is at its highest in the beginning of a project, during the conceptual phases (refer to [Fig 1](#)). Early design influence can reduce or eliminate the need for later design changes (need for redesign) in order to make the Product fit for operation and support. LSA requirements are considered useful for designing a new Product with regards to total system availability and cost effectiveness. The purpose is to influence the design with LSA requirements in a manner similar to how the design of the support system is influenced by the primary Product design and requirements. This is done in a structured way within a project through reviews and baselines.



ICN-B6865-S3000L0077-001-01

Fig 1 Opportunity to influence design and support cost during Product lifetime

Integrated design teams enable requirement and Product design to benefit both the Product and the customer. LSA is considered to be an integral part of systems engineering.

The logistics footprint, the physical size and distribution of the support resources of the supporting logistics system is influenced by the Product design and a structured iterative interactive closed loop is necessary between design disciplines and LSA disciplines.

1.3

Scope

The scope of this chapter is to describe:

- design parameters to be influenced by LSA and vice versa
- how to make a strategy for LSA influencing design
- the perspectives of supplier and vendors
- reviews of milestones, critical or not
- the benefits of LSA influencing the design

The approach to influence the design with the parameters of availability is applicable on systems and subsystems within the Product as well as the support systems design and development.

2

Design considerations

Many considerations are possible upon design, which can be used to influence the design of a system/subsystem or equipment with regards to make the complete Product more efficient and cost effective. Such considerations are:

- Availability
- Reliability
- Maintainability
- Testability
- Prognostics
- Standardization

- Interchangeability
- Environmental considerations
- Human factors/ergonomics
- Obsolescence
- Supportability
- Cost effectiveness

2.1

Availability

Availability is the measure of the degree to which an item is in an operable and ready-for-use state at the start of a mission or operation, when the mission or operation is called for at an unknown time. This is sometimes called operational readiness.

Reliability, maintainability and supportability all contribute to the availability of the Product (refer to [Fig 2](#)).

The **inherent availability** of a system is driven by the reliability and maintainability of the Product. It is described as the probability that a system, when used under stated conditions in an ideal support environment (eg no lack of support resources) will operate sufficiently at any point in time. It excludes preventive maintenance, delay times and is expressed as:

$$A_i = \frac{MTBF}{MTBF + MTTR}$$

Where

- A_i is the inherent availability
- $MTBF$ is the mean time between failure
- $MTTR$ is the mean time to repair

The **achieved availability** is similar to inherent availability with the exception that preventive maintenance is included. It is expresses as

$$A_a = \frac{MTBM}{MTBM + \bar{M}}$$

Where

- A_a is the achieved availability
- $MTBM$ is the mean time between maintenance
- \bar{M} is the mean active maintenance action time

Operational availability is achieved through the combination of the Product and its support system. It is described as the probability that a system, when used under stated conditions in an actual support environment, will operate sufficiently when called for. It is expresses as:

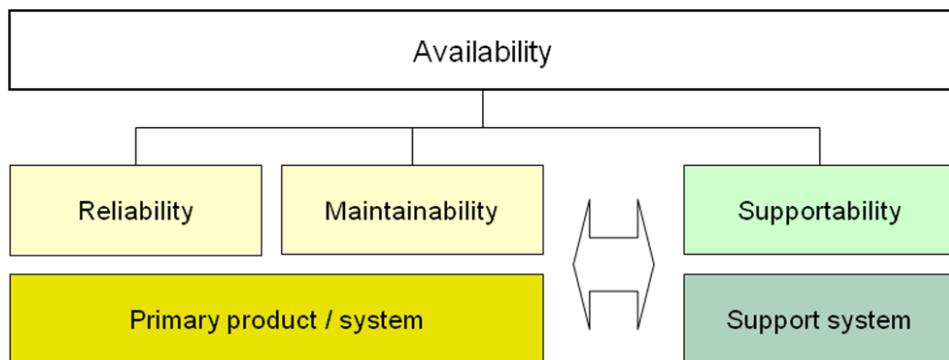
$$A_o = \frac{MTBM}{MTBM + MDT}$$

Where

- A_o is the operational availability
- $MTBM$ is the mean time between maintenance

- MTD is the mean maintenance down time ¹

Here is a most important point - the availability of the Product, is effectively gained by putting requirements on both the Product and the support system, and is achieved by the combination of the two through maintenance actions.



ICN-B6865-S3000L0078-001-01

Fig 2 Breakdown of availability as an approach to structure design influence

2.2

Reliability

Reliability is a prime driver of support resources. It relates to the duration or probability of failure-free performance of a Product under stated conditions, or the probability that an item can perform its intended function for a specified interval under stated conditions.

Products with high reliability are usually cost effective. In case of unavoidable low reliability, the design of the Product must support a compensation by realizing easy failure location and easy performance of maintenance activities. To maintain availability, it is to some extent possible to compensate low reliability with increased maintainability and supportability.

2.3

Maintainability

Maintainability is the measure of the ability of an item to be retained in or restored to a specified condition, when maintenance is performed by personnel having specified skill levels, using prescribed procedures and resources, at each prescribed level of maintenance and repair.

Characteristics of maintainability are, for example, the duration to replace or repair an equipment in order to restore or maintain its functionality, the amount and complexity of support resources that are required and the required skill levels of the personnel performing the activities.

Product design can be sensible or be able to withstand the environment in different degrees. For example the design can be specific to withstand an operational and maintenance environment in airports/airbases, loading/unloading with support equipment or to withstand sandstorms, hail and salt laden environments. For possible inputs to consider, refer to [Chap 8](#).

For maintainability aspects for software, such as software loading, refer to [Chap 13](#).

2.4

Testability

Testability is a design characteristic which allows the status (operable, inoperable, or degraded) of an item and the location of any faults/failures within the item to be confidently determined in a timely fashion.

¹ Availability definitions after Benjamin S Blanchard, *Logistics Engineering and Management*, 6th edition, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632

A design solution to fulfill testability requirements can be achieved by a test system integrated into the Product, or by a test system as a part of the support resources requiring an interface to the Product in order to locate failures. Carefully designed test systems reduce the effort needed to locate and correct failures/faults and can also be used to verify full system functionality.

Redesign driven by testability requirements is a highly complex activity. Milestones in the design program must be added to allow for necessary iterations. For more information about FMEA/FMECA results which drive testability requirements, refer to [Chap 7](#).

2.5

Prognostics

Product parameters and data can be used in models to predict maintenance need or repair. The functionality of maintenance or repair prognosis can either be part of the Product or the support system.

The benefits of prognostics are increased means of knowing a Product' "health" status and, based on that knowledge, avoid critical failures and plan for activities to maintain system functionality.

In a system that does not have a means of prognostics, preventive maintenance is used to prolong the system lifetime and system safety and is balanced against the corrective maintenance actions and resources.

2.6

Standardization

It is usually cost effective to use standard equipment instead of equipment which is specially designed. Development/design cost and time is reduced when using an existing part or system that fulfils the requirements.

Utilization of existing logistic support resources, which fulfill the requirements, can also be possible when using standardized equipment and reduce the need for investments in design and acquisition of support systems resources. Design decisions that lead to the need of specially developed SE must be considered carefully since such equipment drives cost.

Other positive effects are an increase of Product/system maturity and knowledge and an increase of mobility of the operational unit.

Factors that support the potential benefits are the following:

- Use of existing items avoids the development costs that would be incurred to develop new support resources
- Cost to develop new training programs can be avoided
- Commonality of support resources can increase the availability on support resources and reduce the logistics footprint
- Use of standardized items reduces the development time required to determine support resource requirements
- Personnel proficiency in using support and test equipment can be increased through an increase in frequency of use of the same item, rather than having to learn how to use different items

Standardization includes the requirement of interchangeability.

Software design can also be influenced by standardization requirements concerning for example programming languages, information structures or software and information carrying media.

To achieve the benefits, requirements on standardization must be established prior to initiation of the design effort so that the cost of designing or redesigning to meet requirements can be minimized.

2.7 Interchangeability

A Product design is recommended, which enables interchangeability of equipment, components and parts within or between Products, where applicable. The purpose is to increase flexibility of usage of the equipment and a reduction of spare stock levels.

To achieve the benefits, requirements on interchangeability must be established prior to initiation of the design effort so that the cost of designing or redesigning to meet requirements can be minimized.

2.8 Environmental considerations

Known operational environment and maintenance environment information can influence the choices of equipment and design solutions. Temperatures, humidity, sandy and salty environments, are examples of parameters that influence operation and maintenance.

It is also of interest to ensure that the Product fulfils requirements in-between operation during transportation, assembly and disassembly and storage. Taking these considerations into account increases the robustness of the Product.

Chemicals/materials needed for maintenance, operation and disposal from an environmental perspective are also important. The implications of including hazardous materials, hazardous waste, and environmental pollutants have an impact on the Life Cycle Cost (LCC) as well as the safety of personnel in contact with the Product. For more information and depth on this subject, refer to [Chap 17](#).

2.9 Human factors/ergonomics

Requirements on the Product for human factors and ergonomics must be identified. Product design must consider accessibility for operators and technicians. Activities during operation and maintenance with humans in the loop must be considered, too. Handling qualities, for example, parameters such as weight and size are relevant drivers for support resources in order to physically be able to perform activities.

Design tools can be used to analyze virtually or by other means the accessibility for human and the disassembly and assembly activities.

Chemicals/materials as part of the Product or needed for maintenance, operation and disposal are important from a human factors perspective, too. The implications of including hazardous materials and environmental pollutants have an impact on the safety of the personnel and the need for support resources.

Product interface requirements such as access panels for technicians or maintenance personnel and operator stations requirements are a joint interest for LSA and design.

2.10 Obsolescence

Depending on the Product life cycle length, it is recommended to consider the possible situation of obsolescence already during the design phase. Where obsolescence occurs, actions will be necessary to keep the functionality. Examples of typical actions are re-design/modification, lifetime purchasing activities or scrapping of the system and replacement with a new system. When designing or choosing systems or subsystems to a design of a Product, it is useful to keep in mind the future replacement of an item due to obsolescence.

2.11 Supportability

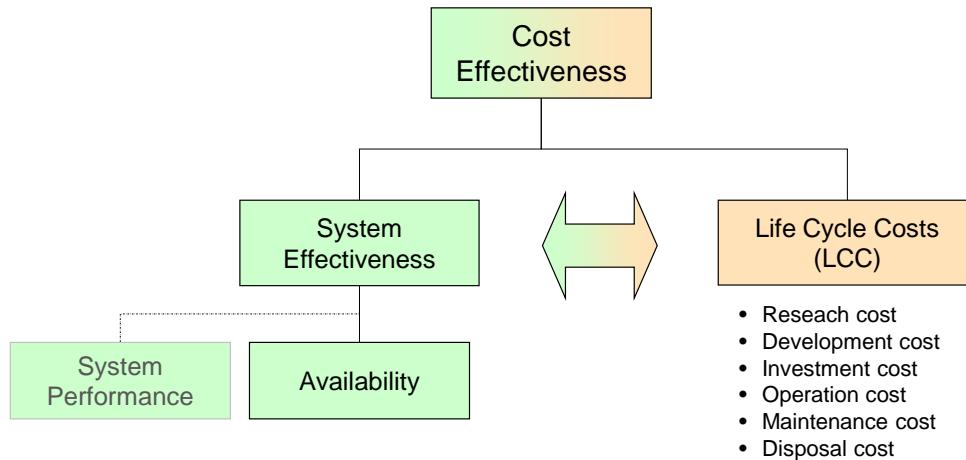
Supportability is the measure of the degree to which all resources required to operate and maintain the Product are provided in sufficient quantity. Supportability encompasses all elements of ILS support resources such as technical information, support equipment, spares or personnel.

Through careful system design, the amount of resources required for supporting a Product can be reduced by considering maintainability and reliability. Product downtime, in reality consisting

of active maintenance time, logistics delay times and administrative delay times can be reduced. A Product that requires a lot of support resources is more exposed to the risk of shortage of support resources, waiting or queuing for resources.

2.12 Cost effectiveness

The design and the choices involved in design have a cost implication for the LCC, which can be broken down into design/development, production/procurement, operational and disposal costs. Comparative analyses between different alternatives of design solutions, including the support resources, are required. These characteristics and costs are necessary to balance availability and LCC (refer to [Fig 3](#)).



ICN-B6865-S3000L0079-001-01

Fig 3 Effectiveness - balancing availability and LCC

2.13 Software design

Software is usually a part of modern Product design. Careful considerations are required to influence the software design in order to give the software some of the features mentioned previously in this paragraph.

Considering the consequences of software support activities described in [Chap 13](#), decisions are required regarding how the software will be designed and implemented in order to achieve the availability requirement of the system.

3 Development programs

It is recommended to include LSA in development projects and to perform the corresponding LSA activities in a documented, structured and interactive closed loop process.

A representative for LSA is an important member of the entire project team.

3.1 Strategy for LSA influence on design

The key to a productive and cost effective analysis effort is the concentration of available resources on activities which most benefit the program.

The broad objectives of LSA are to influence design, structure the most effective support concept, and to define logistic support resource requirements. These general objectives must be translated into more specific objectives for individual projects. The amount of design freedom (flexibility) and opportunity of design influence varies with the type of program/project and is the main input when deciding on a strategy.

Objectives and the analysis strategy must be iterated, refined and balanced against available resources until they become firm program goals or requirements.

3.2

Design influence on development programs

The very nature of development programs can vary from:

- New development programs where the Product and support products are developed from the very beginning
- System or subsystem design changes/improvements of an existing Product
- Fast track programs using existing technology developed in-house or by a supplier

The amount of design freedom and possible design influence therefore can shift. Often, a development program of a complex Product is a combination of the different types. LSA effort and objectives must be focused accordingly.

Furthermore, the possibility of design freedom can exist for the support system but not the Product, and vice versa. The LSA objective of making reliability, maintainability and supportability requirements an integrated part of Product requirements and design can best be achieved if designers are oriented toward reliability, maintainability and supportability objectives commencing with the design effort.

3.3

Supplier design

Supplier design, eg design performed by a vendor or subcontractor, must be approached in the same manner as in-house design.

It is important to provide the supplier with LSA goals and requirements specific to the supplier in order to influence supplier design before the design efforts commence.

In conjunction with design influence by requirements and goals, the requiring party must initially decide and specify the LSA tasks that are to be done by the supplier, which tasks are to be shared between the requiring party and the supplier, and those that are to be performed solely by requiring party. Once done, the LSA portion of the contracting plan can be developed and work requirements written into the procurement documentation.

It is very useful to allow the prospective performing activities, under the bidding terms of the procurement, to recommend adding or deleting LSA activities and to provide a more detailed subtask definition and schedule.

Additionally, prospective performing activities can be encouraged to make use of cost effective data generation procedures. Acquisition program objectives must be considered in preparing procurement documents. For example, in technology demonstration procurement, certain LSA task requirement can be specifically excluded. Supportability objectives for this type of procurement would best be served through design influence and generation of LSA data for subsequent detailed analysis efforts when the technology is utilized. If the acquisition program is oriented to develop and procure a Product, then other LSA tasks become equally important.

3.4

Critical/milestone design reviews

It is essential to establish and document design review procedures (where procedures do not already exist) which provide for official review and control of released design information with LSA program participation in a timely and controlled manner.

These procedures define accept/reject criteria, the method of documenting reviews, the types of design documentation subject to review, and the degree of authority of each reviewing activity.

Program planning must coordinate the design and development reviews and the LSA reviews. Formal review and assessment of LSA requirements are an integral part of each Product design review. Results of each Product design review must be documented. Design reviews must identify and discuss all pertinent aspects of the LSA program.

Technical information generated and documented during the design process must be disseminated among designers and supportability specialists in order to surface interface problems between design concepts and operators, maintainers, and support equipment. Technical design information such as diagnostic features, interfaces, reliability estimates, obsolescence evaluations, and item functions, which determines supportability, must be an integral part of design documentation.

Scheduled “bottom up” reviews are recommended to include subcontractors and suppliers, as appropriate. Agendas must be developed and coordinated to address the relevant topics as they apply to the program phase activity and the review being conducted.

Examples of topics are:

- LSA conducted by task and WBS element
- LSA assessment of proposed design features including supportability, cost, and readiness drivers and new or critical support resource requirements
- Corrective actions considered, proposed, or taken, such as:
 - Support alternatives under consideration
 - System/equipment alternatives under consideration
 - Evaluation and trade-off analysis results
 - Comparative analysis with existing Products
 - Design or redesign actions proposed or taken
- Review of LSA requirements - supportability (with review of specifications as developed)
- Progress toward establishing or achieved goals
- LSA documentation required, completed and scheduled
- Design, schedule, or analysis problems affecting LSA

4

Checklists

To facilitate design work and reviews, the following checklists can be used as a starting point and input. However, creativity in defining checklist questions and in adjustments to the program in question is absolutely necessary. Often, the checklists can be defined through dialogue between the LSA and design disciplines.²

4.1

Selection of parts/equipment

A checklist for the selection of parts/equipment can include but is not limited to:

- Have appropriate standards been consulted for the selection of parts?
- Have the selected parts/equipment been evaluated with regards to reliability, maintainability and supportability?
- Have supplier sources for component part procurement been established?
- Is the supplier reliable in terms of quality level, ability to deliver on time and on budget?

4.2

Reliability

A checklist for the consideration of reliability can include but is not limited to:

- Has the system/equipment wear-out period been defined?
- Have failure modes and effects been identified?
- Are item failure rates known?
- Have parts with excessive failure rates been identified?
- Has mean life been determined?

² Checklist developed from Benjamin S Blanchard, *Logistics Engineering and Management*, 6th edition, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632

- Has equipment design complexity been minimized?
- Is protection against secondary failures (resulting from primary failures) incorporated where possible?
- Have reliability requirements been met?

4.3 Maintainability

A checklist for the consideration of maintainability can include but is not limited to:

- Are the total numbers of different kinds of fasteners minimized?
- Are the fasteners used standard items?
- Have the fasteners been selected based on the requirement for standard tools in comparison with special tools?
- Are equipment identified as replaceable items?
- Is the time it takes to replace an item minimized?
- Is consideration taken upon the operational environment, such as hail storms, sandy and salty environment?
- Is consideration taken on how to load software, loading times and media?

4.4 Testability

A checklist for the consideration of testability can include but is not limited to:

- Have self-test provisions been incorporated where appropriate?
- Is the extent of self-test compatible with LORA?
- Is the self-test automatic?
- Have direct fault indicators been provided? (eg light, message)
- Are test points/interface provided to enable check-out and fault isolation beyond the level of self-test?
- Are test points/interface accessible?
- Are test points/interface functionally or conveniently grouped to facilitate sequential testing?
- Are test points/interface provided for direct test of replaceable items?
- Are test points labeled?
- Can every equipment malfunction be detected by a no-go indication at system level?
- Will software provide adequate test information?

4.5 Prognostics

A checklist for the consideration of prognostics can include but is not limited to:

- Is the functionality to predict maintenance need identified for the equipment?
- Are the parameters used to predict maintenance need identified and obtainable?
- Is the sampling frequency of the parameters used for prognostics adequate?

4.6 Standardization

A checklist for the consideration of standardization can include but is not limited to:

- Are standard equipment/parts incorporated in the design to the highest extent possible?
- Are the same parts used in similar applications?
- Is the number of different part types used throughout the design minimized?
- Are identifying labels and markings and nomenclature standardized to the highest extent?

4.7 Interchangeability

A checklist for the consideration of interchangeability can include but is not limited to:

- Are modules and components that have similar functions electrically, functionally and physically interchangeable?

- Are components with the same part number but provided by different suppliers completely interchangeable?

4.8 Environment

A checklist for the consideration of environmental aspects can include but is not limited to:

- Are the hazardous material and pollutants identified and kept to a minimum?
- Are the material and equipment chosen for the design costly to handle, store and to disassemble or waste?
- Are special containers or facilities necessary to handle hazardous materials or pollutants?

4.9 Human factors/ergonomics/accessibility

A checklist for the consideration of human factors, ergonomics and accessibility can include but is not limited to:

- Are doors provided where appropriate? Are they hinged?
- Are the size and location of openings adequate for access?
- Are the doors and openings labeled? What information is contained in the label?
- Are access doors fasteners minimized?
- Are access doors fasteners of the type quick-release?
- Can access be gained without tools?
- If tools are needed to gain access are the number minimized and of standard variety?
- Are access between modules and components adequate?
- Are the hazardous material and pollutants identified and kept to a minimum?
- Is it necessary to use protective equipment when performing the maintenance task?
- Are maintenance tasks possible to perform with protective equipment if necessary (eg gloves, helmet)?
- Are number of lifting devices for heavy or bulky items minimized?
- Is the time it takes to carry out a maintenance task reasonable compared with the working position?
- Are access requirements compatible with the frequency of maintenance?

4.10 Obsolescence

A checklist for the consideration of obsolescence can include but is not limited to:

- Is the item at risk for obsolescence during the lifetime of the Product?
- Is the design made to facilitate redesign if necessary due to obsolescence?
- Is there a possibility of emulation or re-creation of the design?
- Is there an aftermarket source available?
- Is the supplier required to inform and initiate a purchase process of parts to cover the rest of the lifetime?

Chapter 6

Human factors analysis

Table of contents

	Page
Human factors analysis.....	1
References.....	1
1 General	1
1.1 Introduction	1
1.2 Objective	1
1.3 Scope.....	2
2 Logistic support analysis and human factors	2
2.1 Physical abilities and limitations	2
2.2 Limitations due to hazardous conditions	2
3 Human factors analysis aspects.....	3
3.1 Influence on design	3
3.2 Guidance for LSA	3
4 Human factors to be considered.....	4
4.1 Anthropometric aspects.....	4
4.2 Ergonomic aspects	4
4.3 Environmental aspects	4
5 Additional information	5

List of tables

1 References	1
2 Sample limits for lifting.....	4
3 Sources for additional information	5

References

Table 1 References

Chap No./Document No.	Title
MIL-STD 1472F	Human engineering, design criteria for military systems, equipment and facilities

1 General

1.1 Introduction

Human factors provide source data that must be used within LSA activities to determine maintenance crew and support equipment requirements. This relationship begins in the design process and continues through the development of the maintenance task analysis.

1.2 Objective

The functions of LSA, maintainability and supportability respectively must be closely coordinated to ensure that potential support solutions are within established support thresholds including the requirements of human factors. This is accomplished by considering the crew size

and the required support resources for operational and maintenance tasks. Human factors limitations influence the establishment of support environment as well as the design of the Product itself. The limited capabilities of a human being determines the limitations on Product usage and support. In the area of support particularly, human factors have significant influence on the practicability of operational support or maintenance.

1.3

Scope

This chapter is directed at logistics personnel who perform technical and logistic analysis activities. It describes the relationship and integration between human factors and the logistic support analysis process. Some of the analysis activities can be influenced by the abilities and the natural limitations of a human being. It is recommended that the results of the human factors analysis be documented and be available at a very early project phase. Within the LSA database the influence of human factors can appear at different places. Warnings and cautions can be a typical criterion as well as the need for specific equipment to perform a task and/or to protect a person while in an unpleasant environment.

2

2.1

Logistic support analysis and human factors

Physical abilities and limitations

The LSA and human factors integration occurs throughout the whole Product life cycle. Changes to maintenance that includes human factors constraints and/or limitations can be driven by Product modifications or proposed design changes.

LSA and human factors analysis have a common objective, but specific purposes and goals. The human factors aspects focus on various standards, which give rules and guidance for:

- Anthropometric aspects
- Ergonomic aspects
- Other physiological aspects

LSA must take into account these standards to evaluate potential support solutions. One of many examples of this is the diameter of the average human forearm. Any access panel that requires a reach beyond the wrist must conform to the minimum size that is given in appropriate standards. In this case, the human factors influence the design to ensure that this size requirement is met for all these access panels. In addition, LSA must consider the need for special support equipment required to complete a task being performed under limited moving space conditions. If alternate design solutions are identified during analysis activities, they can be presented back to design for reconsideration.

2.2

Limitations due to hazardous conditions

Another aspect concerning human factors is the limitation of human activity because of health threat.

The handling of dangerous material or material which is hazardous to health must follow strict regulations to ensure physical integrity. At the end of the Product life cycle during the disposal phase in particular, human factors can be of crucial interest due to an increased need to handle material, which is a risk for human health.

Other limitations must be considered for work which is carried out under extreme environmental conditions such as:

- cold or hot environments
- humid environment
- working underground or underwater
- critical environment due other reasons (eg dust, exposure to fumes, noise)

Regulations must ensure the protection of human beings against the effects of these environmental impacts. These regulations must be addressed during the LSA as well as the limitations because of physical restrictions.

3

Human factors analysis aspects

3.1

Influence on design

There are many industry standards that address human factors design constraints and requirements. These must be appropriately referenced as part of the developer/customer agreements and included in the contract.

The human factors qualitative data must be taken into account to determine the maintenance crew skills, size and support equipment needs. For example, human factors will provide weight limits for a single person lift. These weight limits are then used to identify the maintenance crew size for a specific task. This same analysis will also identify the need for mechanical lifts and/or maintenance stands. This information can be used in a trade-off study for design alternatives as an iterative process. Human factors provide specifications and standards that have to be applied to the individual maintenance tasks.

If applicable, the personnel requirement "crew size" should take into account the demographic aspects as well. For example, an all-male crew can have a greater lift limit than a mixed male/female crew. Therefore, the weight of the item defines the proper number of people required for lifting but that quantity can change with the specific crew demographics. The weight restriction is to be taken into account for mechanical lift requirements and therefore impact support equipment requirements. The location can influence the need for maintenance stands but the maintenance crew size must be considered in the definition of the type of stand required.

3.2

Guidance for LSA

The LSA GC must identify the rules and guidelines that will be applied to the LSA and human factors analysis which can include eg maintenance crew demographics, lift, reach and other standards and limitations. The customer and contractor must agree on specific standards that will be applied. In addition to this agreement it is recommended to review the applicable standards and document any exceptions.

LSA receives input from design and develops trade-off analysis of each drawing alternative. From the first drawing effort, LSA will analyze the design and compare support requirements between alternatives and make recommendations based on the supportability and LCC. The process involves identifying all the possible maintenance actions. Each maintenance task will result in a list of resources required to accomplish the task. These resources include spare parts and consumables, maintenance man-hours, training, support equipment and test equipment. The specific considerations relative to human factors will be the number of maintainers and the amount of human factor related support equipment.

The number of maintainers is influenced by the variety of skills required and the weight and size of the parts being changed during the maintenance task. For lifting, the weight of the item and the height must be taken into account in order to identify the maintenance crew size. Each of the various human factors standards has weight limits for a single person lift. The amount can vary based on crew demographics and the height the item must be lifted. Also, if the item is to be carried over a distance, standards define the weight and distance limits. A sample of limits reflecting US DOD regulations from MIL-STD 1472 is given at [Table 2](#).

These limits can vary between standards but in all cases they must be applied to the MTA. If an item exceeds these limits, additional maintainers are required providing that the appropriate number of lifting handles are attached. If multiple persons are required for lifting, then the charts in the human factors standards that define maintainer size must be checked against the physical dimensions of the item to ensure that multiple persons can actually assist in the lifting. It is recommended that lifting by mechanical means be considered when the weight or size of an item exceeds personal lift limits.

Table 2 Sample limits for lifting

Handling function	Population (male and female)	Population (male only)
Lift an object from the floor and place it on a surface not greater than 152 cm (5 ft) above the floor.	16,8 kg (37 lb)	25,4 kg (56 lb)
Lift an object from the floor and place it on a surface not greater than 91 cm (3 ft) above the floor.	20,0 kg (44 lb)	39,5 kg (87 lb)
Carry an object 10 m (33 ft) or less.	19,0 kg (42 lb)	37,2 kg (82 lb)

The entire analysis process is an iterative process and must be used for each design change. Within the LSA process, the more detailed MTA starts once the Product design is approved. Each maintenance task is compared against the human factors standards to ensure that all maintenance and operational support activities are within the human factors limitations.

4 Human factors to be considered

The lists of human factors are not limited to the aspects given here, but can serve to give an impression of which aspects of human factors can influence Product design and, especially in the area of LSA, the design of the support environment (operational and maintenance related).

4.1 Anthropometric aspects

Anthropometric aspects that influence design include but are not limited to:

- Lines of sight (visual field, vertical and horizontal)
- Audio signals requirements
- Muscle strength of arms, hands and thumb
- Required muscle strength for vertical pull extensions
- Required muscle strength for horizontal push and pull movements
- Maximum weight of units to be lifted
- Maximum weight of support equipment
- Arm and hand access dimensions

4.2 Ergonomic aspects

Ergonomic aspects that influence design include but are not limited to:

- Design of controls (eg switches, cranks, joysticks, ball controls, hand wheels, levers, pedals, knobs)
- Minimum handle dimensions
- Workspace design (eg seated, standing, mobile)
- Difficult accessibility - ramps and ladders
- Doors and access panels dimensions
- Illumination requirements

4.3 Environmental aspects

Environmental aspects that influence design include but are not limited to:

- Effective temperature
- Limits of extreme cold and warm temperature conditions
- Influence of wind-chill on human beings
- Decreased performance of human beings under extreme climatic conditions
- Ventilation requirements
- Exposure limits ultraviolet radiant energy

- Exposure limits to pollution like dust, fumes
- Noise limitations
- Shock current intensities

5 Additional information

Each project must determine human factors standards for use in the analysis process. In [Table 3](#), websites are listed for more detailed information how to evaluate the human factor related support requirements of any design.

Table 3 Sources for additional information

Source	URL
Human Factors and Ergonomics Society (HFES)	http://www.hfes-europe.org
Designing for humans	http://www.designingforhumans.com

Chapter 7

Results of FMEA/FMECA in LSA

Table of contents

Page	
1 General	3
1.1 Introduction	3
1.2 Objective.....	3
1.3 Scope.....	3
1.4 Interface with related analyses	3
1.4.1 FMECA in design and development activities	4
1.4.2 LSA FMEA and design-oriented FMECA	4
1.4.3 LSA FMEA and maintainability, maintenance and safety analysis	4
2 Analysis procedure and sub process description	5
2.1 General flow chart.....	5
2.2 Establish FMEA for replaceable units	5
2.2.1 Limit the breakdown level	5
2.2.2 Easy identification	6
2.2.3 Take software into account in the breakdown	6
2.2.4 Physical breakdown and LSA FMEA.....	6
2.2.5 Prediction of failure rates	7
2.3 Identify available means of failure detection	8
2.3.1 Detection means.....	8
2.3.2 Detection rates and detectability ratings	8
2.3.3 Identification of requirement for new test equipment.....	9
2.4 Identify possible means of localization of failed items	9
2.4.1 Localization means	9
2.4.2 Localization ability ratings.....	10
2.5 Analyze requirements for troubleshooting procedure.....	10
2.5.1 Cases in which a procedure is required	10
2.5.2 Detection confirmed upon possible false alarm.....	11
2.5.3 Interface with criticality analysis	11
2.5.4 Troubleshooting upon detected but not localized failure.....	12
2.6 Elements required for a troubleshooting procedure	12
2.7 Record results.....	13
3 Inputs	13
3.1 Physical breakdown.....	13
3.2 FMEA.....	13
4 Outputs	13
4.1 Sub process outputs to be recorded.....	13
4.2 Failure Mode Analysis for LSA tabular report.....	14
5 Complementary analysis	16
5.1 Criticality analysis	16
5.2 Removal criteria.....	16
5.3 Software failure analysis.....	17
5.4 Troubleshooting task analysis	17
5.5 New test equipment management documents	17
6 Lessons learned	17
6.1 Undetectable failures	17
6.2 Random failures.....	18
6.3 Influence on design	18
6.4 Failure rates and FMR	18

Page	
1 General	3
1.1 Introduction	3
1.2 Objective.....	3
1.3 Scope.....	3
1.4 Interface with related analyses	3
1.4.1 FMECA in design and development activities	4
1.4.2 LSA FMEA and design-oriented FMECA	4
1.4.3 LSA FMEA and maintainability, maintenance and safety analysis	4
2 Analysis procedure and sub process description	5
2.1 General flow chart.....	5
2.2 Establish FMEA for replaceable units	5
2.2.1 Limit the breakdown level	5
2.2.2 Easy identification	6
2.2.3 Take software into account in the breakdown	6
2.2.4 Physical breakdown and LSA FMEA.....	6
2.2.5 Prediction of failure rates	7
2.3 Identify available means of failure detection	8
2.3.1 Detection means.....	8
2.3.2 Detection rates and detectability ratings	8
2.3.3 Identification of requirement for new test equipment.....	9
2.4 Identify possible means of localization of failed items	9
2.4.1 Localization means	9
2.4.2 Localization ability ratings.....	10
2.5 Analyze requirements for troubleshooting procedure.....	10
2.5.1 Cases in which a procedure is required	10
2.5.2 Detection confirmed upon possible false alarm.....	11
2.5.3 Interface with criticality analysis	11
2.5.4 Troubleshooting upon detected but not localized failure.....	12
2.6 Elements required for a troubleshooting procedure	12
2.7 Record results.....	13
3 Inputs	13
3.1 Physical breakdown.....	13
3.2 FMEA.....	13
4 Outputs	13
4.1 Sub process outputs to be recorded.....	13
4.2 Failure Mode Analysis for LSA tabular report.....	14
5 Complementary analysis	16
5.1 Criticality analysis	16
5.2 Removal criteria.....	16
5.3 Software failure analysis.....	17
5.4 Troubleshooting task analysis	17
5.5 New test equipment management documents	17
6 Lessons learned	17
6.1 Undetectable failures	17
6.2 Random failures.....	18
6.3 Influence on design	18
6.4 Failure rates and FMR	18

6.5	Limits of software failure analysis.....	18
-----	--	----

List of tables

1	References	2
2	Failure mode ratio distribution	7
3	Failure mode ratings	8
4	Detection ability ratings	9
5	Localization ability ratings.....	10
6	Troubleshooting upon false alarm	11
7	Troubleshooting assessment from localization rate of built-in tests.....	12
8	Outputs of each sub process.....	13
9	Recommendations for Failure Mode Analysis for LSA tabular report	15

List of figures

1	Logical flowchart of analysis procedure	5
2	From technical FMEA to LSA FMEA - grouping of failure modes	6
3	Failure Mode Analysis for LSA tabular report.....	14

References

Table 1 References

Chap No./Document No.	Title
<u>Chap 3</u>	LSA Business process
<u>Chap 8</u>	Damage and special event analysis
<u>Chap 10</u>	Scheduled maintenance analysis
<u>Chap 13</u>	Software support analysis
<u>Chap 21</u>	Terms, abbreviations and acronyms
MIL-HDBK 217	Military Handbook for Reliability Prediction of Electronic Equipment
MIL-STD-882	Standard Practice System Safety
MIL-STD-1629	Procedures for performing a Failure Mode, Effects, and Criticality Analysis
NSWC-11	Handbook of Reliability Prediction Procedures for Mechanical Equipment
RDF-93	Handbook of Reliability Data for Electronic Components
RDF-2000 UTE-C 80-180 (IEC-62380)	Reliability Prediction
S4000P	International specification for developing and continuously improving preventive maintenance

1 General

1.1 Introduction

Failure Modes Analysis (FMA) is a part of event-driven maintenance analysis.

Scheduled Maintenance Analysis (SMA) (refer to [Chap 10](#)) and Special Event Analysis (refer to [Chap 8](#)) focuses on scheduled and/or preventive maintenance. FMA and Damage Analysis focuses on corrective maintenance (refer to [Chap 8](#)). Damage can also be subject to preventive inspections.

FMA establishes the methodology and decision logic that is a prerequisite to the identification of corrective maintenance tasks to be applied to the Product in case of an inherent failure occurrence.

1.2 Objective

All items and equipment liable to fail are candidates for corrective maintenance whether they are subject to servicing or preventive maintenance tasks or not. Some failures can be detected or observed during operation (by person or by integrated test system) or during a maintenance task or inspection. Other failures can be detected but not localized, and others are not detectable at all.

Therefore, in several cases, additional search and troubleshooting work will be necessary to precisely identify and localize failed items so that they can be removed and replaced. This work:

- is characterized by data that can be derived from a Failure Modes and Effect Analysis (FMEA) and must be documented in the LSA record (Refer to [Chap 19](#))
- must be documented in the technical publications
- can require special tools and test equipment

The objective of this procedure is to provide a methodology that helps define the maintenance tasks rendered necessary by the inherent limits of reliability and testability, whether integrated or not, and the related means of support.

1.3 Scope

The scope of this procedure is to:

- exploit existing documents as far as possible (FMECA from the design office, for instance), or build on existing FMEA data or build a new FMEA, tailored to LSA needs, if no existing documents are available
- analyze means available for detection of failures and for localization of failed items
- identify possible requirements for specific troubleshooting procedures
- identify possible requirements for special tools and test equipment
- record all results

The process is applicable to any item within the Product, specifically developed for the needs of a new Product, re-used from previous projects, or acquired from the Commercial-Off-The-Shelf (COTS) range of a supplier.

1.4 Interface with related analyses

Several types of FMEA and Failure Mode, Effects and Criticality Analysis (FMECA) exist as parts of design, development and production management activities. According to the nature and requirements of the individual project, these FMECA can be:

- mandatory to satisfy regulatory requirements (eg, nuclear plants, chemical plants, transports systems subject to airworthiness or seaworthiness regulations, military systems)
- purposely implemented to manage the achievement of an objective (eg, improve safety, improve reliability, improve mission availability) throughout the project

- purposely implemented to identify and justify corrective maintenance activities triggered by inherent failure causes: this is the purpose of LSA FMEA

1.4.1

FMECA in design and development activities

The FMECA is an essential function in design from concept through development. It is iterative and corresponds with the nature of the design process itself. The level of effort and sophistication of the approach used in the FMECA depends on the nature and requirements of the individual project. This makes it necessary to tailor the requirements for a FMECA to each individual project. Tailoring requires that, regardless to the degree of sophistication, the FMECA contributes meaningfully to project decision. A properly performed FMECA is invaluable to those who are responsible for making project decisions regarding the feasibility and adequacy of a design approach.

The usefulness of the FMECA as a design tool and in the decision making process depends on the effectiveness with which problem information is communicated for early design attention. Timeliness is one of the most important factors in differentiating between effective and ineffective implementation of FMECA. While the objective of a FMECA is to identify all modes of failure within a Product design, its first purpose is the early identification of all catastrophic and critical failure possibilities so they can be eliminated or minimized through design correction at the earliest possible time. Therefore, the FMECA is generally initiated as a functional FMECA as soon as preliminary design information is available at the higher system levels, and extended to the lower levels as more information becomes available on the items in question.

Although the FMECA is an essential reliability task for design activities, it also provides information for other purposes. The use of the FMECA, or of some analysis derived from it, is called for in maintainability, safety analysis, availability analysis, LSA, elaboration of maintenance concept, and in failure detection and fault isolation. This coincidental use must be taken into account during the FMECA planning effort to prevent the harmful proliferation of requirements and the duplication of efforts within the same contractual project.

1.4.2

LSA FMEA and design-oriented FMECA

Whereas design-oriented FMECA needs to analyze and quantify the reliability of each individual component of a Product, such a level of detail is not always essential for maintenance purposes because the maintenance activities do not focus on the individual components but on replaceable or repairable units, which are located at a higher level of breakdown than that of individual components.

Therefore the LSA FMEA is generally coincident with, but not identical to, the design-oriented FMECA. Tight interface, coordination and milestones must be implemented between these two activities to ensure adequate consistency, timeliness and traceability between them. The interfaces and specific data element needs must be defined in the LSA Program Plan, and agreed between LSA, design FMEA functions and testability analysis.

The main difference between design FMECA and LSA FMEA is that different failure modes from a design perspective can be grouped in the same failure mode for LSA FMEA because they lead to the same maintenance action.

1.4.3

LSA FMEA and maintainability, maintenance and safety analysis

Other maintainability, maintenance or safety analyses use data from LSA FMEA or derived from FMECA, eg, Level of Repair Analysis (LORA), Reliability-Centered Maintenance (RCM) and Scheduled Maintenance Analysis (SMA). In order to take full benefit of existing works and to avoid useless duplication of works, these other activities must be tightly interfaced and coordinated with FMECA and LSA FMEA in the LSA PP. Refer to [Chap 3](#).

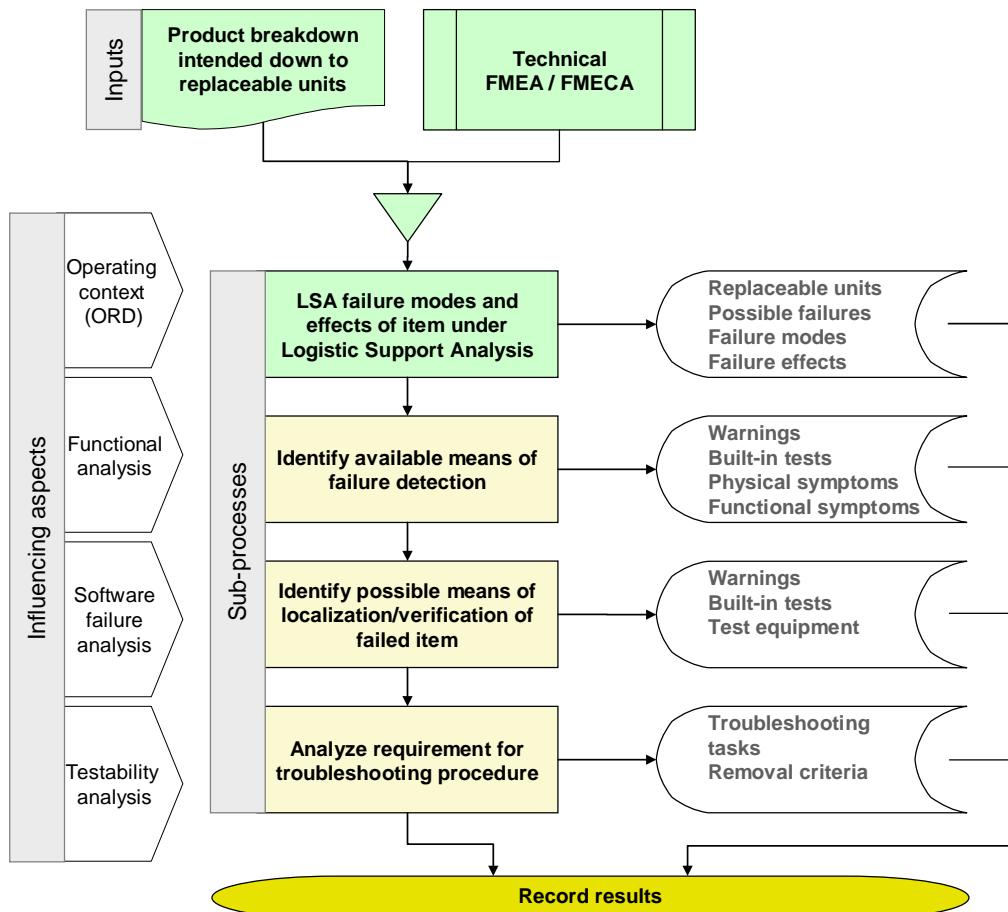
2

2.1

Analysis procedure and sub process description

General flow chart

The workflow of the analysis procedure is presented on the general flowchart, refer to [Fig 1](#).



ICN-B6865-S3000L0080-001-01

Fig 1 Logical flowchart of analysis procedure

2.2

Establish FMEA for replaceable/repairable units

In general, the process is based on the re-use of an existing FMEA/FMECA and possible additional data elements or information to be added. These results are sensibly grouped as much as they can be, in order to keep the effort to be spent at a sensible, low level. This lower level process essentially consists of merging the FMEA/FMECA and the physical Product breakdown. Preparatory works can be necessary before this merging is rendered possible.

The FMEA is started at the end and then worked upwards to top elements. By using this approach the analyst has the failure modes of the smaller items in mind so the effect can be determined more accurately.

2.2.1

Limit the breakdown level

The physical Product breakdown generally goes down to individual components required for detailed definition purposes. For the needs of malfunctioning analysis, it does not need to be developed beyond the level of replaceable units, except if further development of breakdown helps in understanding or describing failure modes detection/localization and/or possible inspection or repair of the replaceable unit. A first step is to limit the physical breakdown at the

appropriate level. This is usually done by extraction and simplification of the complete physical breakdown.

2.2.2 Easy identification

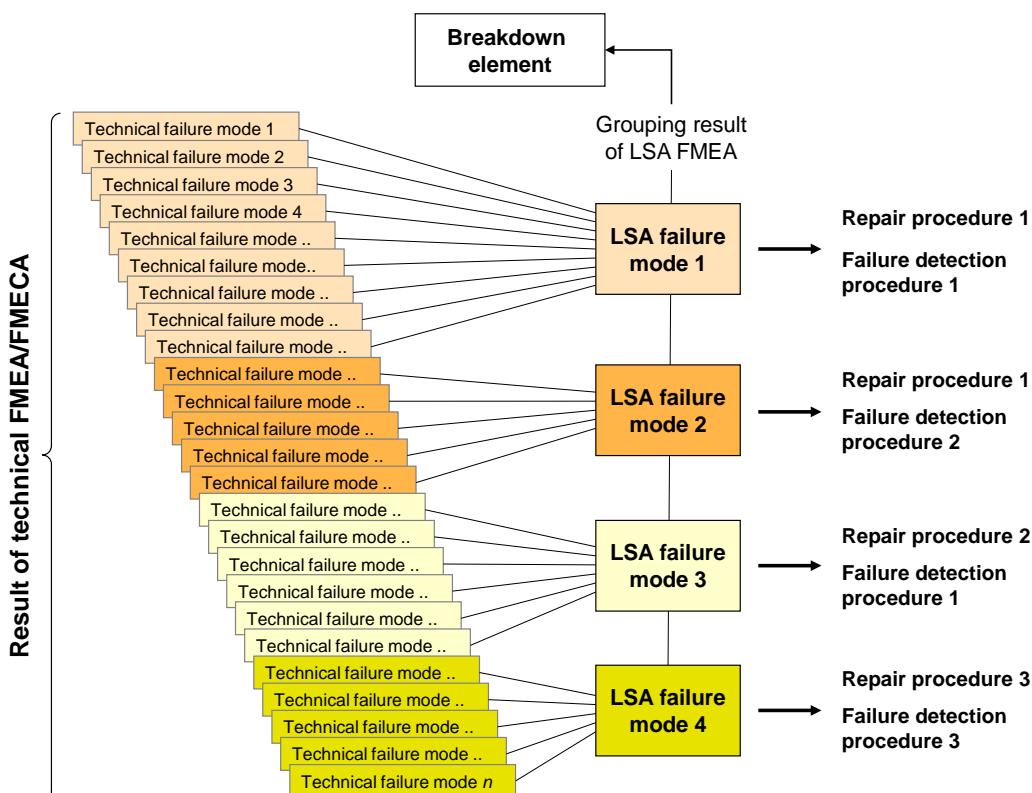
Failures, failure modes and items of the physical breakdown will be frequently quoted in further analysis and analysis reports. It is recommended to adopt a simple identification method (see Breakdown Element Identifier) for these failures, failure modes and items.

2.2.3 Take software into account in the breakdown

Replaceable units that have software should include this software as subcomponent for configuration management purposes. This is independent of the reload ability of software, provided that reload ability is equivalent for software of replace ability for hardware. Note that some software can have defects, too, especially on non-mature systems.

2.2.4 Physical breakdown and LSA FMEA

LSA FMEA must be compliant with the physical breakdown. Items liable to fail must be included in the physical breakdown. Failures must also be listed or grouped for each replaceable unit.



ICN-B6865-S3000L0081-002-01

Fig 2 From technical FMEA to LSA FMEA - grouping of failure modes

It is recommended to perform the grouping by taking into consideration:

- All failure modes generated within a technical FMEA/FMECA, which cause one and the same chain of maintenance activities, can be grouped to one LSA failure mode within the LSA FMEA. The Failure detection procedure evoked in [Fig 2](#) describes how to detect the failure from its warnings, built-in tests, physical symptoms and functional symptoms listed as outputs of the sub-process "Identify available means of detection, shown in [Fig 1](#). If,

despite all these detection means, the failure remains hidden or dormant during normal operation, a specific procedure can be necessary as a failure-finding task. This requirement is addressed by SMA (refer to [Chap 10](#)).

- All aspects which have to be taken into consideration concerning grouping of failure modes must be agreed between customer and contractor within the LSA Guidance Conference Document (GCD). [Fig 2](#) gives an overview how these crucial grouping criteria can be realized.

While grouping, the failure rate of the LSA failure modes will be defined. This is required for calculation of the corresponding task frequencies of the rectifying tasks linked to the LSA failure modes.

2.2.5

Prediction of failure rates

FMA must establish failure rates for each LSA failure mode. There are a number of methods which include:

- a) Failure rates can be calculated in a bottom-up approach by an analytical method (eg, MIL-HDBK 217, RDF-93, RDF-2000 UTE-C 80-180, NSWC-11) that aggregate failure rates of elements, components, etc., up to the LSA failure mode.

Note

In practice the breakdown is often incomplete down to the last small part. In this case it can be possible that the sum of failure rates of indenture level $n+1$ is smaller than the failure rate of the breakdown element on indenture level n . Alternatively, a failure rate on indenture level n , which is smaller than the sum of failure rates on the deeper indenture level $n+1$ is not logical and must be rechecked.

In the case of the ideal situation of a complete breakdown, the sum of the failure rates of indenture level $n+1$ must be the failure rate of the breakdown element of indenture level n .

- b) If the failure rate of the LSA candidate is known, the failure rates of each of its LSA failure modes can be assessed in a top-down approach by multiplying its value by the Failure Mode Ratio (FMR) of the LSA failure mode. Refer to [Table 2](#). The sum of all failure mode ratios to a failure mode should always be 1 (100%).

Table 2 Failure mode ratio distribution

Failure mode	FMR	Rectifying tasks
LSA failure mode 1	0,10 (10%)	Repair procedure 1 + Failure detection procedure 1
LSA failure mode 2	0,15 (15%)	Repair procedure 1 + Failure detection procedure 2
LSA failure mode 3	0,50 (50%)	Repair procedure 2 + Failure detection procedure 1
LSA failure mode 4	0,25 (25%)	Repair procedure 3 + Failure detection procedure 3
Total	1,00 (100%)	

The FMR identifies the fraction of the individual failure mode on the entire failure rate of the item under analysis.

- c) If no analytical method is applicable, the failure rates can be assessed by best engineering judgment from experience on similar Products of comparable complexity and operated in a similar context.
- d) If no dependable failure rate is available, a rough rating table can be established. [Table 3](#) provides an example of a qualitative rating table. It can be tailored according to needs and to the operating context, which must be defined first through the Guidance Document.

Table 3 Failure mode ratings

Rating	Occurrence	Description	Probability of single failure
1	Very low likelihood	Very few failures likely	Less than 0.001
2	Low likelihood	Few failures likely	Between 0.001 and 0.01
3	Moderately low likelihood	Occasional failures likely	Between 0.01 and 0.10
4	Medium likelihood	Medium number of failures likely	Between 0.10 and 0.20
5	Moderately high likelihood	Moderately high number of failures likely	Higher than 0.20

Note

The ratings 1 to 5, in Table 3, correspond to the levels A thru E, respectively, of the qualitative approach of MIL-STD-1629.

2.3

2.3.1

Identify available means of failure detection

Detection means

For each possible failure of a replaceable unit:

- Check if the failure is detected by any form of automatic detection mean which triggers a warning device
- Check if the failure is liable to be detected by a Built in Test (BIT). If yes, assess the detection rate and false alarm rate of this BIT. These rates are relevant here when related to replaceable units, not to functions.
- Irrespective of the answer to the previous question, check functional symptoms and/or physical symptoms likely to help detect the failure. For this check, failure effects listed in FMEA can be considered as a guideline.
- List all failures detectable by any of the above means, record the associated detection means
- List failures undetectable by any of the above means

Note

These detection means are a generalization, irrespective of the level of maintenance, of the failure detection methods and failure detection means of MIL-STD-1629.

2.3.2

Detection rates and detectability ratings

The detection rate is the probability of the detection means to effectively detect the failure.

Detection rates for replaceable units that have embedded BIT can be assessed by an analytical method as part of testability analysis. Therefore the FMA for LSA must be adequately interfaced with testability analysis in order to benefit from the detection rates issued from it. This data can be unavailable from the Technical FMEA/FMECA.

If no analytical method is applicable, a qualitative approach can be preferred. This qualitative assessment can be based on best engineering judgment. An example of this is shown in [Table 4](#), which must be tailored according to needs.

Table 4 Detection ability ratings

Rating	Detection probability	Description	Detection rate
1	High likelihood	High effectiveness for detection	Higher than 80%
2	Moderately high likelihood	Moderately high effectiveness for detection	Between 80% and 60%
3	Medium likelihood	Has medium effectiveness for detection	Between 60% and 40%
4	Moderately low likelihood	Moderately low effectiveness for detection	Between 40% and 20%
5	Low likelihood	Low effectiveness for detection	Less than 20%

Note

The probability of the detection means to effectively detect the failure must be rated in accordance with what is considered effective in terms of SMA. The above ratings must be consistent with the effectiveness of visual inspections, detailed inspections, special detailed inspection, visual checks, operational checks and functional checks deemed effective for the purpose of scheduled maintenance.

2.3.3**Identification of requirement for new test equipment**

[Table 4](#) can be used to identify a requirement for additional test equipment or BIT that is likely to raise detection probability from an unacceptable rating up to an acceptable one.

This identification is generally performed as part of testability analysis for BIT, and as part of Maintenance Task Analysis (MTA) for additional test equipment. Therefore the use of [Table 4](#), for the purpose of identifying a requirement for an additional detection means, can be interfaced with testability analysis and with MTA.

2.4**Identify possible means of localization of failed items****2.4.1****Localization means**

For each detected failure of a replaceable unit:

- Check whether an automatic detection (if any) triggering a warning device is likely to localize without ambiguity the one replaceable unit that has failed, or if it is likely to localize one out of n replaceable units that has failed (one-among- n ambiguousness).
- Check whether a BIT (if any) is likely to localize without ambiguousness the one replaceable unit that has failed, or if it is likely to localize one out of n replaceable units that has failed (one-among- n ambiguousness).
- Check whether a monitoring system is likely to localize without ambiguousness the one replaceable unit that has failed, or if it is likely to localize one out of n replaceable units that has failed (one-among- n ambiguousness). For example, there are such monitoring systems for health and usage monitoring.
- Check whether applicable functional checks, leading to detection by functional symptom, are likely to localize without ambiguousness the one replaceable unit that has failed, or if they are likely to localize one out of n replaceable units that has failed (one-among- n ambiguousness).
- Check whether cross-checking of BIT results can help reduce the ambiguousness of the localization (eg reduce the value of n in the "one-among- n ambiguousness"). For this cross-check, a special attention can be paid to BIT results which are impacted by a common cause.

- Check whether applicable visual checks or measurements without any dismounting or removal, leading to a detection by physical symptom, is likely to localize without ambiguousness the one replaceable unit that has failed, or if it is likely to localize one out of n replaceable units that has failed (one-among- n ambiguousness).

2.4.2 Localization ability ratings

Localization is limited to detected failures.

Localization ability ratings can be elaborated as part of testability analysis on Products entirely covered by built-in tests. Some additional engineering can be required if the results of these tests have to be cross-checked with other source of information (eg, functional checks, visual checks, visual inspections without removal) in order to reduce the ambiguousness. In some cases, this additional engineering can be performed by an activity external to LSA (eg, fault tree analysis). In this case, an adequate interface must be implemented with the FMA for LSA. This data is can be unavailable from the technical FMEA/FMECA. If that is the case the data can be retrieved from the testability analysis.

An example of this is shown in [Table 5](#), which must be tailored according to needs.

Table 5 Localization ability ratings

Rating	Localization probability	Description	Localization ability rating
1	High likelihood	High effectiveness of the abovementioned localization means	Higher than 90%
2	Moderately high likelihood	Moderately high effectiveness for localization	Between 80 and 90%
3	Medium likelihood	Medium effectiveness for localization	Between 60% and 80%
4	Moderately low likelihood	Moderately low effectiveness for localization	Between 40% and 60%
5	Low likelihood	Low effectiveness for localization	Below 40%

2.5

2.5.1

Analyze requirements for troubleshooting procedure

Cases in which a procedure is required

A troubleshooting procedure is required when:

- a failed item is not likely to be localized without ambiguity by any of the localization means described in [Para 2.4](#)
- a failure has been detected by a built-in test known to have a high false alarm rate

However, cases in which such a procedure is required are hardly ever defined by a simple cross-grid of the previous tables. The main reasons for this are:

- Detection rate and localization rate are statistical variables, and are generally bound to failure rate and failure mode rating
- Defining a threshold on the detection and the localization rates would create an artificial threshold effect
- If the failure has been detected by a BIT with a false alarm rate known to be high, its results must not be ignored if the failure has some possible impact on safety. This remains true even if the BIT has been repeated without detecting a failure.

- The requirement for troubleshooting is not entirely dictated by detection and localization ability and can be analyzed in the light of complementary elements, such as accessibility, removal and replacement costs, or safety

Therefore, analyzing the requirement for a troubleshooting procedure needs an expert's best engineering judgment. [Table 6](#) provides guidelines that can be used after being tailored according to needs. The guidelines assume that detection rate and localization rate take the failure rate/failure mode rating into account.

2.5.2 Detection confirmed upon possible false alarm

Prior to any troubleshooting, the failure detection can be confirmed if there is doubt about the reality of the detected failure. However, note that a non-repeated detection of failure can be due to a random failure of the tested equipment itself, regardless of false alarm rate of the testing device.

Therefore, it is recommended to assess the need for this preliminary step from the importance of the consequences of the failure, as given in [Table 6](#).

Table 6 Troubleshooting upon false alarm

Consequences of failure	Repetition of test for removal of ambiguousness	Follow-on actions
Negligible	The test can be repeated a certain number of times (to be determined) with the usual test device or built-in test until it is confirmed or invalidated.	If the failure detection is invalidated, no further analysis is to be carried out and the equipment can be declared back to operational status. If the failure is confirmed, need for troubleshooting is assessed as in Para 2.5.4 .
Not negligible but acceptable	The test can be repeated a certain number of times (to be determined) with an independent test device until it is confirmed or invalidated.	If the failure detection is invalidated, no further analysis is to be carried out and the equipment can be declared back to operational status. If the failure is confirmed, need for troubleshooting is assessed as in next paragraph.
Unacceptable	The test needs not be repeated.	Proceed as in next paragraph.

In any case, the need for repetition of test and the follow-on actions must be documented in a procedure which is applicable prior to the troubleshooting procedure itself.

2.5.3 Interface with criticality analysis

For Products that have a criticality analysis performed, thresholds of acceptability used in the left column of [Table 6](#) must be consistent with the consequences of the failure mode and/or the criticality of the replaceable item.

In this approach, the worst potential consequence of the failure must be taken in consideration and determined by the degree of injury, property damage, system damage or environmental

impacts that can ultimately occur. This analysis is addressed by the severity analysis in FMECA methods (eg, MIL-STD-1629, MIL-STD-882).

This potential consequence of a failure mode can be combined with the probability of occurrence of the failure, depending on the nature of the Product. This combination is addressed by the criticality analysis in FMECA methods (eg, MIL-STD-1629).

The failure on critical or very severe failure modes must always be considered as unacceptable, even if there is a possibility for the detection to be a false alarm. Therefore the troubleshooting procedure requirement analysis must be interfaced with the FMECA.

2.5.4

Troubleshooting on detected but not localized failure

[Table 7](#) is provided as a guideline for troubleshooting detected non-localized failures. It can be tailored according to needs.

Table 7 Troubleshooting assessment from localization rate of built-in tests

Localization rating	Troubleshooting required
1	No troubleshooting required. Localization will be performed by the BIT.
2	Troubleshooting is required but can be performed by cross-checking of BIT results.
3	Troubleshooting is required and can require an intervention of maintenance team to perform additional research of symptoms
4 or 5	Troubleshooting is required, using test equipment likely to compensate the inability of the BIT to localize the failed replaceable unit.

2.6

Elements required for a troubleshooting procedure

Elaborating on the troubleshooting procedure itself is out of the scope of FMA for LSA. However, in order to simplify subsequent works, several documents or analyses used to determine if such a procedure is required should be noted for further use:

- The technical documents and definition files that must be used to analyze the troubleshooting requirement (eg, technical plan, wiring diagram, interface description).
- Logical order of the tests to be carried out until the failed replaceable unit is identified and localized. This logical order can be supported by a flow-chart diagram on which all decisions are applicable (eg, associated with the clear interpretation of a symptom or a BIT result). All decision not associated with such an interpretation should be considered as not applicable. It is recommended to document test equipment or inspection procedures which can be necessary.
- Possible cross-checks that can help identify and localize the failed unit. Such cross-checks are useful for identifying sources of common cause failures. They should concern items exposed to the effects of the common cause failure.
- Symptoms to be monitored or recorded during these cross-checks
- Measurements to be carried out to check the physical and/or functional characteristics of the replaceable units
- Test equipment required by the above measurements
- Expected values for each of these measurements, and signification of unexpected measures

Refer to [Para 5.4](#) for the elaboration of the troubleshooting procedure itself.

2.7 Record results

Results must be recorded as troubleshooting tasks, which are preliminary tasks for corrective maintenance. These tasks are to be attached to the next higher assembly including all the replaceable units concerned. They must list all the resources which have been identified during the above sub process in addition to usual data characterizing a maintenance task. The corrective maintenance task for removal and replacement of the failed replaceable unit should be mentioned. Troubleshooting task and removal/replacement tasks can be subtasks of a repair task.

3 Inputs

3.1 Physical breakdown

Physical breakdown (or Product breakdown) provides an indented decomposition of the Product. For FMEA purposes, it should be developed down to the level of replaceable units at least. The corresponding depth of indenture can vary depending on the Product's design. Further development to sublevels of these items can be necessary only when it helps characterize the failure modes of the item or the related failure effects. Otherwise it can be dispensed with. Replaceable units should be easily identifiable in this breakdown.

3.2 FMEA

The content of the FMEA includes but is not limited to:

- a list of items likely to fail
- the associated failure rates
- the related failure modes foreseen
- the probability of occurrence of these failure modes
- the effects of these failure modes regarding the availability of the replaceable unit and its impacts on safety and overall availability
- the signature of the failure (eg, physical symptoms, functional symptoms, BIT results)

This FMEA can be carried out as part of a Product safety analysis, of a Product mission reliability analysis, or provided by the supplier of the equipment.

4 Outputs

4.1 Sub process outputs to be recorded

The sub process outputs that are recommended are listed in [Table 8](#) and summarized in [Fig 3](#).

Table 8 Outputs of each sub process

Sub-process	Outputs
Establish FMEA for replaceable/repairable unit	Replaceable units/repair procedures Possible failures Failures modes Failure effects
Identify available means of failure detection	Warnings Built-in tests reports Physical symptoms Functional symptoms Requirement for new test equipment

Sub-process	Outputs
Identify possible means of localization of failed item	Warnings Built-in tests reports Ground test equipment Requirement for new test equipment
Analyze requirements for troubleshooting procedure	Answer yes/no to need for a troubleshooting If yes, inputs for troubleshooting task analysis

Replaceable unit identification (eg BEI)		Function	Failure Modes	Failure causes	Failure effects	Failure mode ratio (FMR)	Detectability rating	Detection means	Localize ability rating	Localization means	Troubleshooting required	Procedure requirement	Test equipment required
1	2	3	4	5	6	7	8	9	10	11	12	13	

ICN-B6865-S3000L0083-002-01

Fig 3 Failure Mode Analysis for LSA tabular report

4.2

Failure Mode Analysis for LSA tabular report

It is recommended that a tabular report, which provides all important information and outputs in a convenient form be issued.

The columns of this table can be populated as sub-processes progress.

An example of Failure Mode Analysis for LSA tabular report is given in [Fig 3](#). Recommendations for filling in the columns are provided in [Table 9](#).

Note

Header possibly required for purposes of traceability and quality management of the FMA for LSA has not been included. This header can provide release date, issue number, author's name, approval signatures, information required for identification of the item/replaceable unit, etc.

This tabular report must be tailored according to needs.

The numbers refer to recommendations provided in [Table 9](#).

Table 9 Recommendations for Failure Mode Analysis for LSA tabular report

Column	Description
1	Replaceable/Repairable unit identification: Item name. This item can be followed by its physical breakdown, provided that each line contains only one item or sub-part of item. The item name can be accompanied with or replaced by appropriate reference number or BEI.
2	Function: Function that the unit fulfills or contributes to
3	LSA failure modes
4	Failure causes: Cause-to-effect phenomena leading to the failure
5	Failure effects: Impacts and consequences of the failure, in terms of safety, function availability, collateral damage, non-compliance with regulations or specifications. This column is to be used to document effects of an LSA failure mode. Failure effects are sorted and analyzed as: Local effects: impacts of the failure on the operation and functions of the replaceable unit in the breakdown level under consideration. The impacts include second-order effects resulting from failure. The purpose of this characterization of local effects in LSA FMEA is to help define detection/localization criteria of failures. Next higher effects: impacts of the failure on the operation and functions of the replaceable unit in the next higher breakdown level. The purpose of this characterization of next higher effects in LSA FMEA is to help define detection/localization criteria of failures. End effects: total effect that the failure has on the operation, function, or status of the Product. The purpose of this characterization of failure effects in LSA FMEA is to help define detection/localization criteria of failures. If necessary, column 5 can be split in sub-columns to clarify the presentation of the above failure effects.
6	Failure mode ratio (refer to Para 2.2.5)
7	Detectability rating: Detection rate as assessed in testability analysis (if any and if appropriate) or rated as in Para 2.3.2 .
8	Detection means: Short description of detection means available for this failure. These means can be used during normal operating of the system (eg, warning on control panel, abnormal behavior of the system, perceptible loss of function, functional symptom) or during a maintenance activity (eg, visual inspection, post-incident checks, maintenance task, exploitation of BIT reports, physical symptoms).

Column	Description
9	Localize ability rating: Localization rate as assessed in testability analysis (if any and if appropriate) or rated as in Para 2.4.2 .
10	Localization means: Short description of means available for localizing the replaceable unit or group of replaceable units likely to have failed: warnings, BIT reports (especially initiated BIT which cannot be initiated during normal operation of the system, use of ground test equipment). This column can be used to document the signature of the failure.
11	Troubleshooting required: Yes or no, in accordance with Para 2.5 .
12	Procedure requirement: Short summary or title of the maintenance task. This procedure can be either a simple procedure if, for example, no troubleshooting is required because failed unit has been localized without ambiguity, or a troubleshooting procedure as described in Para 5.4 .
13	Test equipment required: Short functional description of any additional test equipment found necessary or helpful to simplify, shorten or facilitate the troubleshooting. This test equipment can be Built-in Test Equipment (BITE) or separate equipment.

5 5.1

Complementary analysis

Criticality analysis

If applicable and required, a criticality analysis can be carried out to identify critical items, for which failure would result in unacceptable impact. This analysis would support identification of scheduled maintenance action that would preclude the critical failures from occurring. Refer to [Chap 10](#).

After SMA has been completed, no critical item must be left with undetectable possible failures.

Adequate identification means must be implemented to provide adequate traceability between critical items and drawings, manufacturing, provisioning, technical publication and training. This is normally part of a safety analysis, which must be interfaced with FMA for LSA in order to review their consistency.

5.2

Removal criteria

For items that are exposed to progressive deterioration or to wear-out, removal criterion must be defined in terms of:

- parameter to inspect or to monitor
- threshold for removal or repair action

5.3 Software failure analysis

This analysis is complementary to physical FMEA and identifies software components liable to be buggy or defective and also identifies the possible losses of functions and/or apparent failures of the item (hardware and software) that can be induced by the defects of the software. Refer to [Chap 13](#).

5.4 Troubleshooting task analysis

Troubleshooting is carried out in the same way as MTA and identifies detailed list of tools, spares, test equipment, preliminary tasks to perform for accessibility or safety, ingredients and consumables, repair validation requirements.

This procedure will identify:

- The technical documents that has to be used to perform the troubleshooting (eg, technical plan, wiring diagram, interface description). These documents will be included, quoted or summarized in the troubleshooting work card.
- Logical order of the tests to be carried out until the failed replaceable unit is localized
- Possible cross-checks that can help identify and localize the failed unit
- Symptoms to be monitored or recorded during these cross-checks
- Measurements to be carried out to check the physical and/or functional characteristics of the replaceable units
- Test equipment required by the above measurements
- Expected values for each of these measurements, and signification of unexpected measures
- Skills and manpower required for the above cross-checks and measurements
- Possible requirement for a specific facility (eg, dark room, dust-free atmosphere) in which the troubleshooting must be carried out
- Recommended maintenance level, where applicable, at which the troubleshooting must be carried out
- Removal criteria or repair criteria of each replaceable unit, if they are not already quoted in the corresponding maintenance procedures
- Reference of maintenance procedures that are to be applied to fix, repair or replace the failed item

5.5 New test equipment management documents

It is recommended that any test equipment or BIT deemed necessary to improve detection and/or localization be:

- described in a specification,
- managed through a development and qualification plan,
- adequately milestones regarding planning for development of the system itself
- adequately milestones so that the test equipment can be taken into account in the troubleshooting task analysis and in the maintenance procedure

6 Lessons learned

6.1 Undetectable failures

Special attention must be given to troubleshooting analysis of items:

- exposed to failures undetected, or hardly detectable, during operation or routine maintenance
- whose failure can have hazardous effects

For these items, appropriate periodical inspection or check must be implemented in the preventive maintenance and/or inspection plan. Also refer to S4000P.

For failures with catastrophic effects (eg, more severe than hazardous) and, in some cases, for failures with hazardous effects, this can be performed as part of the SMA, refer to [Chap 10](#). If it is the case, no special attention is to be paid to undetectable failures during FMA for LSA process. However, FMA for LSA and SMA must be adequately interfaced to ensure consistency of definitions, completeness of the failure modes taken into account, and timeliness of analyses.

6.2 Random failures

Random failures can, in some cases, not be detected during the troubleshooting.

This can be incorrectly interpreted as a false alarm generating erroneous failure detection on a failure-free item. This interpretation is flawed, given that a failure actually occurred, even if it is not detected again on replay.

The particular form of troubleshooting induced by random failures can generally not be addressed by FMEA and is therefore out of the scope of this FMA for LSA procedure. It can be addressed and managed through a specific procedure established on a case-by-case basis upon detection of such a failure.

6.3 Influence on design

Unsolvable troubleshooting can lead to propose modifications of design, such as:

- adding test sockets
- enhancing accessibility so that a given test equipment or tooling can be used for troubleshooting
- improving testability and/or detectability

Such possible influence on design must be appropriately managed and allocated appropriate milestones in the design process, especially if a requirement for new BIT or test socket is deemed necessary to improve detection and/or localization. Refer to [Para 2.3.3](#).

6.4 Failure rates and FMR

The failure rates and FMR possibly provided by the FMEA can be used to assess which failure cause or failure mode is the most liable to occur leading to the failure. This can be useful to propose an order of priority among the checks and tests to be carried out as part of the troubleshooting process.

However, in several cases this order of priority is dictated by accessibility reasons or by costs.

6.5 Limits of software failure analysis

This analysis, if carried out, usually does not help rating the failure occurrence probability.

In some cases, this rating can be assessed from quality insurance levels applied during the development of the software, or from knowledge and confidence level placed in the provider.

Importantly, a software failure is generally not associated with a failure rate, given that software failures are not time-related. Software failures, if any, generally occurs during initial deployments of the system and are progressively corrected through a maturity plan or a Product quality improvement plan. Software failure is therefore often considered as a defect that is discovered after it is corrected, and so, it does not reoccur.

For these reasons, software failure can be considered as highly unlikely, or better, on mature systems. Support and maturation of software are addressed in [Chap 13](#).

Chapter 8

Damage and special event analysis

Table of contents

	Page
Damage and special event analysis	1
References.....	2
1 General	2
1.1 Introduction	2
1.2 Objective.....	2
1.3 Scope.....	2
1.4 Terms, abbreviations and acronyms	2
2 Analysis logic	3
3 DSE analysis process.....	3
3.1 Identification of special events.....	3
3.2 Identification of potential damages	5
3.3 Define element or part of the Product concerned.....	6
3.4 Criticality analysis	7
3.5 Identification of maintenance task requirements.....	7
4 Inputs	7
4.1 Product breakdown.....	7
4.2 Description of Product usage	8
4.3 Statistical elements on special events.....	8
4.4 Product definition.....	8
5 Outputs	8
5.1 Damage and special event analysis tabular report	8
5.2 Influence on design	10
5.3 Lessons learned	10

List of tables

Table 1 References.....	2
Table 2 Terms, abbreviations and acronyms.....	2
Table 3 Causes of events	4
Table 4 Probability of occurrence of special events	5
Table 5 Technology behavior knowledge rating	5
Table 6 Technology sensitivity rating.....	6
Table 7 Recommendations for damage and event analysis tabular report	9

List of figures

1 General flow chart of damage and special event analysis application logic	3
2 Technology/damage evaluation rating	6
3 Damage and event analysis tabular report.....	9

References

Table 1 References

Chap No./Document No.	Title
Chap 3	LSA Business process
Chap 4	Configuration management in LSA
Chap 12	Maintenance task analysis
Chap 21	Terms, abbreviations and acronyms

1 General

1.1 Introduction

Maintenance activities which are caused by damages or special events which occur during normal operation of the Product must be taken into consideration as an important part of the overall maintenance concept. After the identification of these activities they are analyzed in detail by an MTA as described in [Chap 12](#).

1.2 Objective

A methodology for identifying and justifying the maintenance tasks appropriate when a damage or a special event occurs to a Product is given here.

In-service feedback on similar Products in operation can be used to directly identify maintenance requirements for damages and special events that also can occur on the new Product. The use of this information can shorten the Damage and Special Event (DSE) analysis process.

1.3 Scope

Throughout the service life of a Product, DSE analysis can occur to the Product itself and/or the technology the Product uses. The analysis required to understand, assess and minimize these DSEs, is given here.

1.4 Terms, abbreviations and acronyms

Definitions of terms that are specific to DSE analysis are given in [Table 2](#). The complete set of LSA terms, abbreviations and acronyms is given in [Chap 21](#).

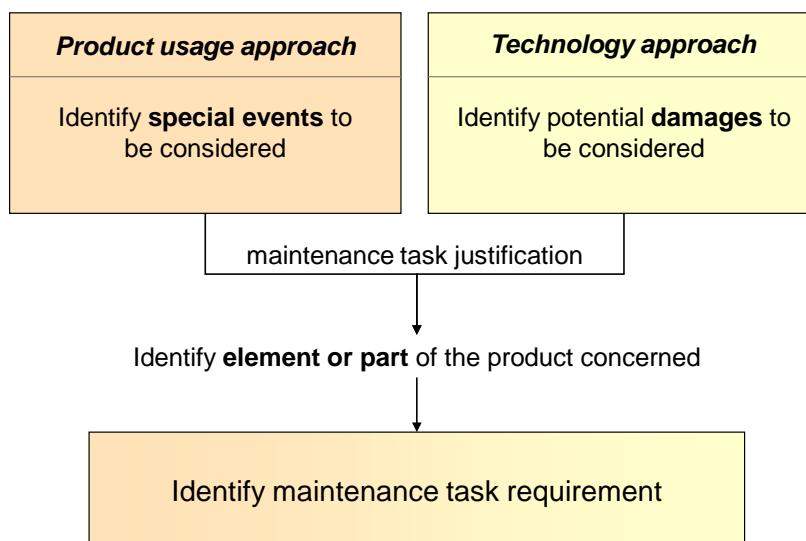
Table 2 Terms, abbreviations and acronyms

Term	Definition
Damage	A loss or reduction of functionality, excluding inherent failure (intrinsic reliabilities). Normally a maintenance task will be required. Damages can be grouped into damage families. For example, scratches, dents and cracks can be grouped as a typical damage family for structures. These damage families are typical candidates for a standard repair procedure.
External cause	A cause is described as external when an event independent of Product usage occurs.
Internal cause	A cause is described as internal as a result from Product usage alone.

Term	Definition
Special event	An event that occurs during a Product's life that cannot be considered as normal operation. It can be due either to external causes (eg meteorological phenomenon) or due to out of bound use (eg over-G maneuver of an aircraft).

2 Analysis logic

DSE analysis identifies maintenance task requirements that are justified by the identification of special events or potential damages. Refer to [Fig 1](#).



ICN-B6865-S3000L0068-002-01

Fig 1 General flow chart of damage and special event analysis application logic

3 DSE analysis process

The DSE analysis involves five processes. These are:

- Identification of special events. Refer to [Para 3.1](#)
- Identification of potential damage. Refer to [Para 3.2](#)
- Definition of elements or parts of the Product. Refer to [Para 3.3](#)
- Criticality analysis. Refer to [Para 3.4](#)
- Identification of maintenance task requirements. Refer to [Para 3.5](#)

3.1 Identification of special events

This analysis process is based on a simple Product usage analysis. It identifies the different possible causes and analyzes them through each phase of the Product life to determine whether they can lead to a special event or not. This process follows five steps:

Step 1: List the different causes that can produce an event.

Events can be due to external or internal causes and affected by natural phenomenon or humankind. [Table 3](#) provides examples of different types of causes. It is recommended that a more complete table be developed for each project.

Table 3 Causes of events

Types of causes		Examples
External causes	Natural phenomenon	Meteorological Animal Stones, trees, etc.
	Caused by humankind	Combat threat Material maneuver
	Caused by operation environment	Electromagnetic field Salt/ sand/pollution laden atmosphere
	Caused by transport and storage conditions (sea, air, truck, rail, etc)	Shocks, movements and vibrations Depressurization
Internal causes	Caused by unusual use	Over operational limits like "hard landing", "over-g maneuver"
	Caused by internal dysfunction	Excessive heat Excessive pressure Excessive vibration

Step 2: Identify each usage phase of the Product.

Product usage phases are, for example, operation, maintenance, storage or transport. If necessary, each usage phase can be subdivided into subphases (eg for an aircraft: take off, flight and landing).

Step 3: Apply pertinent type of causes to the different phases of Product life to identify all possible events.

Examples of special events due to an external cause:

- Impact on external skin due to ground handling equipment during servicing
- Impact on external skin due to cargo handling equipment during transportation preparation
- Multiple impacts due to hail
- Impacts on inner external skin due to runway debris during take off
- Lightning strike

Examples of special events due to an internal cause:

- Water entrapment due to a leak
- Excessive stress on a tank structure due to dysfunction of pressure relief valve during a refueling operation on ground
- Excessive temperature in a bay in case of warm air leak
- An unexpected wear-out of mechanical parts

Step 4: Analyze each possible event in order to characterize its probability of occurrence.

For some special events a quantitative approach can be applied based on feedback from previous similar Products. For example: number of bird strikes happening every year.

When it is not possible to get sufficient statistics, a qualitative approach can be applied, refer to [Table 4](#).

Table 4 Probability of occurrence of special events

Rating	Occurrence	Description
1	Extremely unlikely	A special event whose probability of occurrence is essentially zero
2	Remote likelihood	A special event whose probability of occurrence is unlikely. Rare numbers of special events are likely to happen.
3	Occasional	A special event whose probability of occurrence is occasional. A few special events are likely to happen.
4	Reasonably probable	A special event whose probability of occurrence is moderate. A certain number of special events are likely to happen.
5	Frequent	A special event whose probability of occurrence is very high. This special event is almost certain to occur.

Step 5: Depending on the occurrence rating from Step 4, categorize the special event as maintenance significant or no further analysis is required. Refer to [Para 3.3](#). The rating level can be used to determine a threshold for tailoring analysis activities.

3.2

Identification of potential damages

This analysis process involves the evaluation of the behavior of technology used in the Product design. Experience shows that technologies used can be more or less sensitive to damage. The approach consists of identifying the technologies used on the different subsystems/items of the Product and analyzing them from two points of view. These are:

- Is the technology well known or new?
- Is the technology damage proof or sensitive to damage?

Identify the potential damage to be considered using the following methodology:

Step1: For all subsystems/items of the Product, identify the technologies used and characterize them regarding knowledge of their behavior.

A qualitative approach to characterize this knowledge is given in [Table 5](#). The grade of accuracy of the rating can be tailored according to the project requirements.

Table 5 Technology behavior knowledge rating

Rating	Technology behavior knowledge	Description
1	Well known	This technology has been used on many similar projects
2	Known	This technology has been used on similar projects but has been marginally modified for the intended projects
3	Quite new	This technology has been already used on similar recent projects but very little feedback is available

Step 2: Evaluate the sensitivity of the technology.

For each technology identified in Step 1, identify the possible damage types that can occur. For example:

- Corrosion on metallic parts (general, galvanic or other types of corrosion)

- Stress corrosion on a metallic assembly with constraints installed
- Delaminating or humidity absorption on composite material

Evaluate its sensitivity to damage types in relation with possible damage sources.

A qualitative approach to characterize this sensitivity is given in [Table 6](#). The grade of accuracy of the rating can be tailored according to the project requirements.

Table 6 Technology sensitivity rating

Rating	Sensitivity degree	Description
1	Extremely low	This technology has an extremely low chance of damage during Product life
2	Low	This technology has a very low chance of damage during Product life
3	Medium	This technology has a low chance of damage during Product life
4	High	This technology is likely to be damaged during Product life
5	Extremely high	This technology is very likely to be damaged during Product life

Step 3: Select the association technology/potential damage suitable for further analysis.

A selection threshold can be identified by combining the technology behavior knowledge rating and the technology sensitivity rating. Refer to [Fig 2](#). The selection threshold can be adjusted depending on the needs of the individual project.

		Sensitivity				
		1	2	3	4	5
Technology behavior	1	1	1	2	3	4
	2	1	2	3	3	4
	3	2	3	4	4	4

ICN-B6865-S3000L0088-001-01

Fig 2 Technology/damage evaluation rating

3.3

Define element or part of the Product concerned

There is a link between the two approaches (special event and technology behavior), which can be described in terms of the item's potential for damage as:

- A relevant special event that can generate damage to an item that is made with sensitive technology
- An element made with sensitive technology that can be damaged because of a special event

The two approaches are complementary and identify potentially damageable items that can be encountered during Product life. A four step process can be used to identify these items:

Step 1: For each special event selected, identify the affected items of the Product breakdown.

Step 2: For each association technology/damage selected, identify the affected items of the Product breakdown.

Step 3: For each item of the Product breakdown identified at the previous steps, analyze the installation area/design to evaluate exposure to the different threats. For example, level of exposure to corrosion depends on where the structural part is located. An item in a landing gear bay is very exposed to corrosion while a watertight bay that is very rarely opened is significantly less exposed to corrosion.

Step 4: Using the results of the previous steps, decide for each potentially damageable items whether or not it is suitable for further analysis.

3.4

Criticality analysis

In order to focus on the most significant special events /damages, it is recommended that a criticality analysis be considered.

For each potentially damageable item evaluate the criticality in terms of impact on the Product service life by asking:

- Does the damage result in lowering safety level?
- Does the damage result in lowering availability?
- Does the damage result in significant ownership deterioration?
- Does the damage result in significant cosmetic deterioration?

Note

This list of questions is not extensive. Projects can have other questions that are peculiar to the project.

3.5

Identification of maintenance task requirements

The previous analysis process have identified and selected the significant items of the Product breakdown affected by a special event or a possibility of being damaged during service life.

Each significant item and event must now be analyzed to determine the corresponding maintenance requirements.

This analysis involves identifying the different maintenance tasks required to be able to:

- Detect, localize, measure or follow each damage that can occur (eg visual inspection, non destructive test)
- Restore the function of the element (eg remove and replace task, repair task, refurbish task)

Some items can be grouped into families and associated damages are candidates for standard repair procedures (eg standard repair procedures for structural parts, standard repair procedures for wiring)

4

Inputs

The DSE analysis requires four inputs. These are:

- Physical breakdown. Refer to [Para 4.1](#).
- Product usage description sheet. Refer to [Para 4.2](#).
- Statistical elements on special events. Refer to [Para 4.3](#).
- Product definition. Refer to [Para 4.4](#).

4.1

Product breakdown

The Product breakdown provides a structured representation of the Product given by eg systems, subsystems, breakdown elements and parts. Refer to [Chap 4](#).

4.2 Description of Product usage

Within the ORD and CRD, Product usage is described in detail. These documents provide information like operation environment and maintenance situation. Refer to [Chap 3](#).

Note

If an ORD and CRD are not available, the description of Product usage must be documented in an alternate way, eg a simple Product usage description sheet.

4.3 Statistical elements on special events

This is feedback from previous similar projects that quantify the occurrences of special events.

4.4 Product definition

Elements of the Product definition needed are those associated with the technologies used in the different Products, systems, subsystems and items.

5 Outputs

The DSE analysis provides three outputs. These are:

- A special event analysis tabular report. Refer to [Para 5.1](#).
- Influence on design. Refer to [Para 5.2](#).
- Lessons learned. Refer to [Para 5.3](#).

5.1 Damage and special event analysis tabular report

It is recommended that a tabular report be developed that provides all information and outputs in a readable format. It is recommended that these results be recorded in an LSA database in order to keep the traceability of maintenance task requirements.

An example of a DSE analysis tabular report is given in [Fig 3](#). A description of each column is contained in [Table 7](#).

Note

Although not shown in [Fig 3](#), it is recommended that, header information including release date, issue number, author's name, approval signatures, information required for identification of the maintenance task and other project identifiers, be included in the report.

This tabular report can be tailored to meet the project requirements.

Significant element identification	Special event description	Special event occurrence rating	Technology description	Damage description	Technology/damage evaluation rating	Exposure evaluation rating	Criticality analysis result	Maintenance task requirement
1	2	3	4	5	6	7	8	9

ICN-B6865-S3000L0069-002-01

*Fig 3 Damage and event analysis tabular report**Table 7 Recommendations for damage and event analysis tabular report*

Column	Description
1	Significant element identification Element name. This element name can be followed by its breakdown element identifier. Some elements (typically structural elements) can be grouped into families in order to minimize the number of entries in the table.
2	Special event description The special event is described by a complete name. The special event can affect more than one element of the breakdown structure. In this case identify each one on a different line.
3	Special event occurrence rating Give the rating and description of occurrence.
4	Technology description Give a short technical description of the technology. The technology can be associated with more than one element of the Product breakdown. In this case try to group them by families in order to minimize the number of entries.
5	Damage description Give a short description of damage. If there are multiple damages that can affect the same element then identify each one on a different line.
6	Technology/damage evaluation rating Give the rating and description explaining the rating in terms of sensitivity and newness.
7	Exposure evaluation description Give a short technical description of the location of the element identified and how it is exposed to threats.

Column	Description
8	Criticality analysis result Describe the justification of importance of damage regarding its impacts.
9	Maintenance task requirement A short summary or title including the type of maintenance task and the element to which it applied. If more than one maintenance task is needed to cover the same element then identify each one on a different line. Conversely, the same maintenance task can address more than one element or damage. This is typically the case of a standard repair procedure.

5.2

Influence on design

Early in the project development/design process there is the opportunity to influence the design from the standpoint of special events and damage occurrences. Design modifications can reduce or eliminate special events and damages, for example:

- Adding protection
- Enhancing accessibility/removals
- Changing the technology

Such possible influences on design must be documented and managed as mile stones in the design process.

5.3

Lessons learned

This section can give examples of best practices or bad experiences to be taken into account for future studies.

Chapter 9

Logistics related operations analysis

Table of contents

	Page
Logistics related operations analysis	1
References.....	2
1 General	2
1.1 Introduction	2
1.2 Objective	3
1.3 Scope.....	3
2 Logistics relevant operations	3
2.1 Product usage supporting aspects	3
2.1.1 Preparation for usage	3
2.1.2 Servicing.....	3
2.1.3 Adjusting.....	4
2.1.4 Weighing.....	4
2.1.5 Loading and unloading	4
2.2 Packing, handling, storage and transport aspects	5
2.2.1 Packing and unpacking	5
2.2.2 Handling.....	5
2.2.3 Storage	5
2.2.4 Stacking	6
2.2.5 Lifting	6
2.2.6 Transportation.....	6
2.2.7 Mooring.....	7
2.2.8 Shoring	7
2.3 Role change.....	7
2.4 Deployment.....	7
2.5 Software and data aspects	7
2.6 Product or system recovery.....	7
2.7 Special scheduled maintenance.....	8
2.8 Disposal and recycling.....	8
2.9 Extraordinary logistics relevant operations.....	8
3 Documenting logistics relevant operations tasks	8
3.1 LSA database aspects.....	8
3.2 Operational and customer requirements as a source	8
4 Checklist of logistics relevant operations tasks	8

List of tables

1 References	2
2 Checklist for logistics related operations analysis	9

List of figures

1 Logistics related operations analysis within technical documentation	2
2 Typical servicing tasks.....	4
3 Typical PHST tasks	5

References

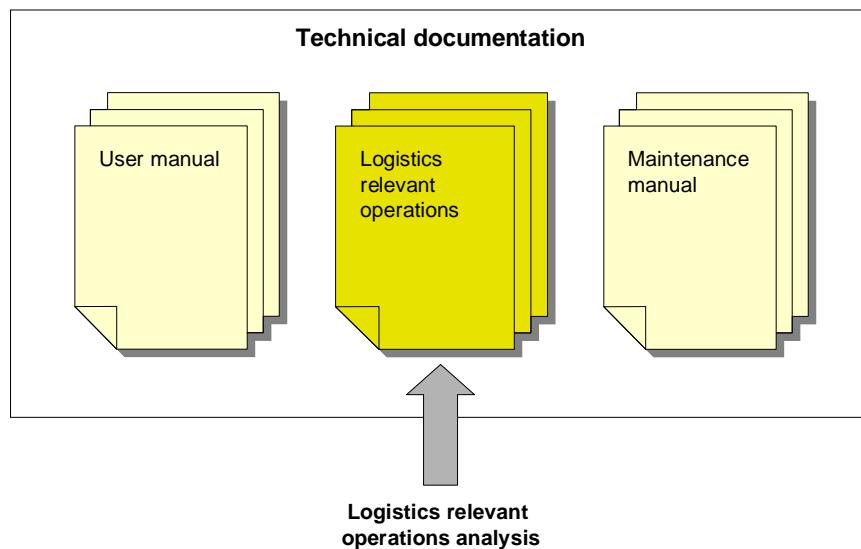
Table 1 References

Chap No./Document No.	Title
<u>Chap 3</u>	LSA Business process
<u>Chap 10</u>	Scheduled maintenance analysis
<u>Chap 13</u>	Software support analysis
<u>Chap 17</u>	Disposal
S1000D	International specification for technical publications using a common source database

1 General Introduction

Beside the activities concerning maintenance and repair of a Product, there are additional aspects concerning the operation and the handling to be considered. Logistic relevant operations are tasks, which cannot be assigned to an area of direct usage of a Product (documented in operating instructions) or an area of maintenance (documented in maintenance manual). However, these tasks can be of importance for the proper usage of any Product. Many aspects such as ease of operation, usability, flexibility of usage or mobility are restricted by logistics relevant operations tasks. The border between pure usage of a Product and the logistics relevant operations is subjective and it must be decided individually for each project, how and where these aspects will be documented. Sometimes it can be difficult to assign a maintenance task exactly to one of these two areas.

It is recommended to clarify at an early stage of any project, the responsibility of both, performance of logistic analysis and documentation of logistics relevant operations.



ICN-B6865-S3000L0039-001-01

Fig 1 Logistics related operations analysis within technical documentation

All tasks that are identified by the logistics relevant operations analysis will close the possible gap between user manual and maintenance manual. Refer to [Fig 1](#).

1.2

Objective

This chapter is a guideline for identifying relevant operational activities. It is directed to the logistics personnel responsible for analyzing the logistics relevant operational tasks. To support the analyst, a detailed list of potential operational or handling activities is provided in [Para 4](#).

1.3

Scope

In the following paragraphs the typical logistics relevant operations are described in more detail to explain the concept. The collection of examples is limited to the most common tasks. There can be more activities, especially in relation to environmental conditions (eg preparation for transport under extreme climatic conditions such as an arctic environment). For each example a short description is given along with special aspects to be considered.

2

Logistics relevant operations

The identification of logistics relevant operations, including the requirements concerning personnel, support equipment, consumables, spare parts, facilities and required training, is an important area of logistics analysis activities. Some of the tasks require very early consideration in the development life cycle and some can be considered later, eg when a prototype of the Item under Analysis (IuA) is available.

2.1

Product usage supporting aspects

The activities that support the usage of the IuA are described in the following paragraphs.

2.1.1

Preparation for usage

The tasks required to prepare a Product for usage/operation can include, for example, the exchange of equipment to provide a special capability. In addition to the modification of the Product by changing some pieces of equipment, it can be possible that peripheral Products/components must be prepared for proper usage.

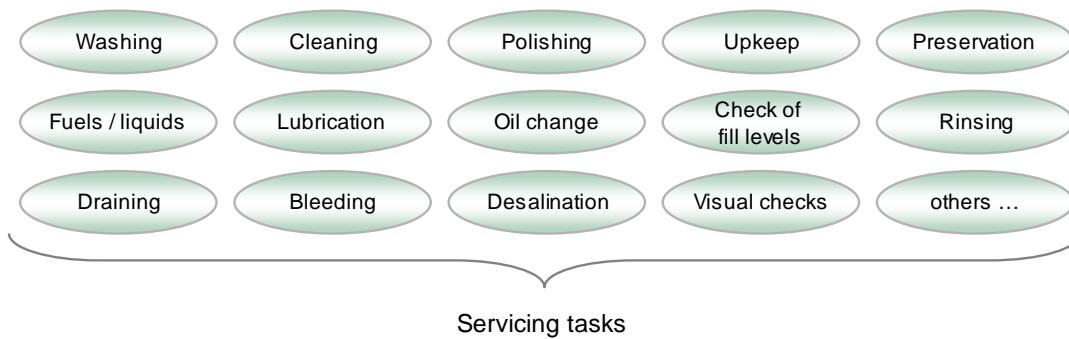
Typical preparation tasks can be classified as servicing tasks, (eg the cleaning of windows before driving a vehicle). Others are typical preparation tasks, which describe the complete conversion of a Product for another use case, such as the change of the tooling within a machine to produce a new product line. Typical examples for tasks concerning preparation for usage, which can be taken into account by Logistics Relevant Operations Analysis, are:

- Preparation of machines for new production segment (eg by exchange of tooling or dies)
- Role change of a vehicle (role change of a normal transport vehicle to an ambulance transport vehicle)
- Preparation of an aircraft for a reconnaissance mission by installing special equipment
- Preparation of a ship including the required equipment for sailing

2.1.2

Servicing

The term servicing can describe a wide range of tasks performed on a Product, also in connection with usage preparation, refer to [Fig 2](#). Other aspects are handling after usage or the upkeep of the Product. This includes, for example, a simple visual inspection for foreign objects during washing or cleaning procedures as well as simple visual inspections concerning damage or the general condition of equipment (eg condition of tires).



ICR-B6865-S3000L0040-001-01

Fig 2 Typical servicing tasks

Servicing tasks must be included in the Maintenance Task Analysis (MTA) to identify the requirements. Servicing tasks can have an important impact on resources, such as personnel, support equipment (eg for lubrication), consumables and facilities (eg washing facilities with oil separator and water recycling equipment). Typical examples for servicing tasks are:

- Washing an engine
- Gear lubrication
- Changing engine oil
- Product polishing and following preservation
- Bleeding brakes
- Changing hydraulic liquid
- Replenishing fluids
- Desalination and rinsing equipment after diving

2.1.3 Adjusting

The typical adjust tasks can also be performed in connection with preparation for usage. Product functionality is not changed, but precision and quality considerations are addressed as preconditions for the usage of the Product. Typical examples for tasks in connection with adjusting are:

- Leveling of the entire Product as a basic task
- Calibration of measuring instruments
- Adjustment of the sight of a gun

2.1.4 Weighing

The analysis of weighing tasks involves the preparation of the IUA for weighing and the procedure itself. This includes information on the weighing equipment to be used. The purpose of the weighing procedure must be defined and agreed to between the contractor and the customer (eg weighing as a mission preparation task or a task to establish proof of specification requirements of the Product).

2.1.5 Loading and unloading

Loading and unloading procedures must be analyzed for each Product that can be used for the transport of cargo. This information must be collected in an early stage in the development life cycle. Examples of loading and offloading techniques, interior layout, floor loadings, location and strength of lashing points, methods of stowing and securing, capacities and dimensions of compartment and doors must be documented carefully. These include:

- What kind of cargo is planned to be transported?
- Which size and weight parameters must be considered?
- Is the expected cargo sensitive to special impacts (acceleration, magnetic or electric fields, pushes, humidity, etc.)?

- Is the cargo critical or even dangerous in case of improper handling?
- What type of cargo securing (lashing and lashing points, stowing) is required?
- Will a container concept be employed?
- Are special loading devices required because of the type of the load (eg rolling devices)

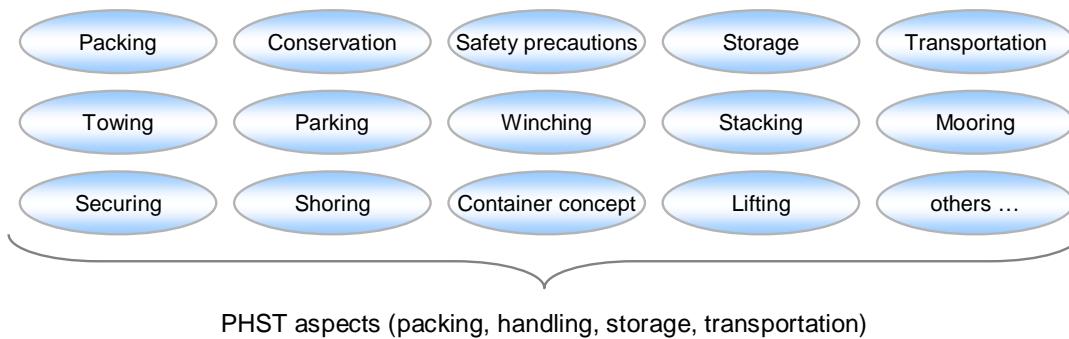
The requirements for special support equipment for loading and unloading must be addressed, too. These include:

- Is it necessary to use special support equipment or vehicles for the loading/unloading?
- Is special support equipment required to secure the cargo?

2.2

Packing, handling, storage and transport aspects

[Fig 3](#) shows an overview of activities as packing, handling, storage and transport (PHST) during other than normal use of the Product.



ICN-B6865-S3000L0041-002-01

Fig 3 Typical PHST tasks

2.2.1

Packing and unpacking

A packing concept for the IUA and/or for components includes the following:

- Is packing required for short term storage and/or for long term storage?
- Is packing required for transportation and what kind of transportation will be carried out?
- Is a special container required for storage or transport?
- Is special preservation required because of extreme climatic conditions during the storage or transportation period (eg sea transport)?
- Is it required to unpack and repackage the IUA during transportation or storage period, for example, to perform maintenance activities (storage tasks)?
- What is required for the unpacking and removal of preservation concerning support equipment and facilities?

2.2.2

Handling

Handling tasks for the IUA includes the following:

- Safety precautions and limitations due to handling
- Instructions necessary to park, tow, winch or move the IUA in any way, other than normal usage
- Instructions necessary to jack the IUA including eg jacking points, required adapters, supports for special components, balance weights, jacking procedures
- Additional equipment and materials required when handling the IUA (eg tow bars or cables)

2.2.3

Storage

To guarantee product functionality during a storage period and at the conclusion of storage, storage procedures must be analyzed and documented. These include:

- Is the required storage considered long term storage or short term storage solution?
- Is the Product/component to be stored of special sensitivity?
- Is it necessary to remove components from the Product to be stored and to store these components separately under special conditions?
- What type of inspection and preventive maintenance is required to safeguard structural and system integrity during storage (eg wheel rotation, provision with electrical power, engine running, and pressure checks)? Provide a timescale for such in-storage maintenance.
- Are there any extreme climatic conditions which are expected during storage period (heat, frost, humidity)?
- Are there any special techniques required for entry into storage (eg cleaning and preservation, fluid system draining/replenishing, static grounding, protective blanking, removal of special components)?
- Are there any special techniques required for removal from storage and to return to usage (eg cleaning and removal of preservation, fluid system replenishing, re-installation of special components, functional checks, preparation for usage)?
- What kind of securing is required during the storage period (eg mooring, blocking of movable components, securing against light in a dark storage place)?
- Is it necessary to consider the storage time in connection with the life of the stored Product/component?

2.2.4 Stacking

A special aspect of storage and transportation of any Product/component is stacking. It is essential that the safety requirements for stacking for storage or transport are addressed. For example, the impact of external events such as pushing must especially be analyzed for storage.

2.2.5 Lifting

A lifting concept must include all necessary procedures to lift the IUA with corresponding hoist devices, cranes, jacks or slings because of:

- Lifting for transportation or loading purposes
- Lifting for repair or maintenance purposes
- Lifting for recovery

In addition to the lifting of the complete Product, the lifting of components must also be considered (eg the lifting of the engine of an aircraft).

2.2.6 Transportation

Based on customer requirements, the analysis of the transportability of the Product must include:

- What is the effort and duration of the preparation for transport and for recovery after transport?
- What are the requirements for preparation for transport and for recovery after transport concerning personnel, tools, consumables, and facilities?
- What are the additional environmental requirements for transportation under special conditions (eg extreme climatic conditions such as sea transport, sandy desert environment, humidity, heat, frost)?
- What are the additional safety requirements for transportation under special conditions (eg mooring within a transport aircraft in case of air transport)?
- What are the required servicing actions during the transportation period (especially during long journeys)?
- Is it necessary to remove components for transportation or to disassemble the entire IUA down to a specific level?
- Should a container concept be considered?
- Is the usage of sledges and/or pallets acceptable?

2.2.7 Mooring

Analysis of mooring tasks for the IUA must be addressed under all weather conditions or for transportation within any transportation vehicle. Mooring is considered as a long or short term activity, the purpose of which is to tie down or otherwise secure the IUA to the ground or within a transportation vehicle to avoid any damage. Also, information such as special techniques applicable to ballasting, definition of lashing points and installation/use of special support equipment applicable to mooring must be included.

2.2.8 Shoring

Similar to mooring, shoring must be analyzed concerning shoring points, shoring procedures and the shoring equipment used during, for example, maintenance, repair and recovery.

2.3 Role change

A Product can be called to cover different duties (eg, a helicopter that can transport personnel and/or cargo). In this case the logistic relevant operations are focused on the tasks (eg, removal of seats, removal of intercommunications apparel and installation of specific equipment and corresponding fixing parts) to be accomplished to change the configuration of the Product to enable the usage in a special role.

2.4 Deployment

Product deployment may include several task considerations depending on the complexity of the Product itself or of a system of systems which contains several Products. It can be related to different activities like just an installation procedure or transportation to the operational area including preparation, operation set up and test of functionalities before operation. Therefore, a task analysis must be evaluated for each system considering the following aspects:

- Operational scenario (eg, facility preparation task, installation or preparation to operation task, personnel skill to deploy the system)
- Transportation capability (eg, fixing procedures tasks, safety procedures)
- Preparation to operation (eg, remote or local set up tasks, calibrating procedures, tuning procedures)

2.5 Software and data aspects

Logistics relevant operations related to software are addressed in [Chap 13](#).

2.6 Product or system recovery

Recovery analysis must include any information on planned recovery procedures and the required support equipment needed to recover any IUA from any condition to which it can be subjected.

Product recovery during the testing phase must be addressed separately. It is important to guarantee a fast reaction to any unexpected event, where it is necessary to recover the Product from an undesired condition. It is essential that any action does not endanger personnel or the environment. A recovery plan for all emergency situations must be detailed enough to avoid endangering personnel or the environment.

Recovery also includes rescue procedures or safeguarding of an area after a catastrophic event. All procedures that are necessary to rescue any Product after an accident must be analyzed before the first usage of the Product. For example, an aircraft performing its first flight marks a specific risk of a crash. All required activities after a potential crash must be analyzed and documented carefully before the first usage of the system. Recovery training must be included as part of the analysis process in order to reduce the risk of extended damage to personnel or the environment in case of a catastrophic event. All aspects must be covered, eg an accident on land, in water (eg the foundering of a ship with the requirement to salvage the wreckage) or within areas that are difficult to access.

2.7 Special scheduled maintenance

In general, the identification of scheduled maintenance is described in [Chap 10](#). However, servicing tasks or preparation for usage can also be of a scheduled nature, but are not identified by the extended Scheduled Maintenance Analysis (SMA). These tasks can be of high impact, eg for personnel requirements, because they are performed frequently (eg even daily).

Examples include:

- Visual check before the start of usage (eg engine start)
- Visual check after special missions
- Scheduled replenishment of fuel, fluids or other consumables to guarantee proper function
- Cleaning before or after each usage

2.8 Disposal and recycling

Analysis concerning requirements after the life cycle of a Product is of increasing importance. This includes both the disposal of the Product, and the handling of components and consumables to be disposed of during the life cycle of the Product. For this reason, these aspects are described in detail in [Chap 17](#).

2.9 Extraordinary logistics relevant operations

Extraordinary logistics relevant operations must be addressed in the analysis process. These include but are not limited to:

- Decontaminating a vehicle
- Disinfecting personnel before starting a job
- De-icing an aircraft or ship
- Checking electrical charge
- Checking the strength of magnetic fields during storage
- Completing paperwork for statistical purposes

3 Documenting logistics relevant operations tasks

3.1 LSA database aspects

The identification of any required logistics relevant operation must be documented within the LSA database eg by using the appropriate information codes from S1000D. It is recommended that logistics relevant operations tasks be documented at the Product level or the appropriate breakdown level (use of a mixed physical/functional breakdown provides the most flexibility). This supports the exchange of data with S1000D for technical publications.

3.2 Operational and customer requirements as a source

In general, the requirements for logistics relevant operations tasks can be derived from the operational and customer's requirements, which are collected and documented within the establishment of Product usage data, refer to [Chap 3](#). It is recommended to use the content of the relevant documents ORD (Operational Requirements Document) and CRD (Customer Requirements Document) as a first input for the Logistics Relevant Operations Analysis. Each aspect concerning usage of the IUA, which can be found in the ORD or CRD, can trigger the requirement for a logistics relevant operations task.

4 Checklist of logistics relevant operations tasks

[Table 2](#) provides an alphabetical checklist for potential relevant activities.

Table 2 Checklist for logistics related operations analysis

Activity	Analysis recommended at which phase	Probable large impact on requirement for facilities	Probable large impact on requirement for special support equipment	Probable large impact on design
Adjusting	When necessary	No	Possible	No
Bleeding	When necessary	No	No	No
Calibration	When necessary	No	Yes	No
Checking	When necessary	No	No	Possible
Cleaning	Early phase	Yes	Possible	No
Conservation	Early phase	Yes	Possible	No
Container concept	When necessary	No	No	Possible
Conservation removal	Early phase	Yes	No	No
Desalination	When necessary	No	No	No
Disposal	Early phase	Possible	Yes	Yes
Draining	When necessary	No	Possible	Possible
Jacking	Early phase	Possible	Yes	Yes
Leveling	Early phase	Possible	Possible	No
Lifting	Early phase	Possible	Yes	Possible
Loading cargo	Early phase	Yes	Yes	Yes
Loading data	Early phase	Yes	Possible	No
Loading Software	When necessary	No	Possible	No
Lubrication	When necessary	No	Possible	Possible
Mooring	When necessary	No	Yes	Yes
Oil Change	When necessary	No	No	No
Packing	When necessary	Possible	Possible	No
Parking	When necessary	Possible	Possible	No
Polishing	When necessary	No	Possible	No
Preservation	Early phase	Possible	Possible	No
Recovery	Early phase	No	Possible	Possible
Recycling	When necessary	Possible	Possible	Yes
Replenishment	When necessary	No	Possible	Possible

Activity	Analysis recommended at which phase	Probable large impact on requirement for facilities	Probable large impact on requirement for special support equipment	Probable large impact on design
Rinsing	When necessary	Possible	No	No
Safety precautions	Early phase	Possible	Possible	No
Securing	When necessary	No	Possible	Possible
Shoring	When necessary	No	Possible	Possible
Stacking	When necessary	No	Possible	No
Storage	Early phase	Yes	Possible	No
Towing	When necessary	No	Yes	Possible
Transportation	Early phase	Possible	Yes	Possible
Unloading cargo	Early phase	Possible	Yes	Yes
Unloading data	When necessary	No	Possible	No
Unloading Software	When necessary	No	Possible	No
Unpacking	Early phase	Possible	Yes	No
Upkeep	Early phase	Possible	Possible	No
Washing	Early phase	Possible	Possible	No
Weighing	Early phase	Possible	Yes	No
Winching	When necessary	No	Possible	No

Chapter 10

Development of a scheduled maintenance program

Table of contents

	Page
Development of a scheduled maintenance program	1
References.....	2
1 General	2
1.1 Introduction	2
1.2 Objective	3
1.3 Scope.....	3
2 Development of a PMTR	3
2.1 System analysis.....	3
2.2 Structure analysis	4
2.3 Zonal analysis.....	4
3 Thresholds, intervals and trigger events for a PMTR	4
3.1 General	4
3.1.1 PO threshold.....	4
3.1.2 PE interval	5
3.1.3 Combination of PO threshold and PE intervals	5
3.1.4 Trigger events.....	5
3.1.5 More than one PE interval in parallel.....	6
3.2 PMTR with threshold, intervals or trigger events from other sources	7
4 PMTR transfer to LSA tasks	7
4.1 Transfer of PMTR out of system analysis	7
4.2 Transfer of PMTR out of structure analysis	7
4.3 Transfer of PMTR out of zonal analysis	7
4.4 LSA activities for a transferred PMTR	8
5 PMTR packaging	8
5.1 Preparation of the PMTR packaging	8
5.2 Development of an operator maintenance program.....	9
5.3 Packaging process for PMTRs	11
5.3.1 General aspects.....	11
5.3.2 Impact of maintenance level on master task packages.....	12
5.3.3 Predefined master task packages	14
5.4 Adaptation of a PMTR interval type and/or numerical interval value	14
5.4.1 General aspects.....	14
5.4.2 Summary of PMTR interval adaptation.....	16
5.5 Maintenance task analysis for master task packages	17
6 In-service maintenance optimization	18

List of tables

1 References	2
2 General interval adaptation rules.....	15

List of figures

1 PO threshold.....	5
2 PE interval	5

3	PO threshold and PE intervals dependent from each other	5
4	Triggers.....	6
5	More than one PE interval for a preventive maintenance	7
6	Process overview from PMTR via LSA tasks to master task packages	10
7	Traceability of scheduled maintenance tasks with intervals.....	11
8	Initial distribution of PMTR depending on maintenance level	13
9	Clustering of PMTRs to support the identification of master intervals.....	14
10	Principles for interval modification	15
11	Integration of single PMTR LSA tasks into master task packages.....	17

References

Table 1 References

Chap No./Document No.	Title
Chap 11	Level of repair analysis
Chap 12	Maintenance task analysis
Chap 14	Life cycle cost considerations
Chap 21	Terms, abbreviations and acronyms
S1000D	International specification for technical publications using a common source database
S4000P	International specification for developing and continuously improving preventive maintenance
ATA/A4A MSG-3	Operator/Manufacturer Scheduled Maintenance Development
DEF-STAN 00-45 Part 3	Using Reliability Centered Maintenance to Manage Engineering Failures, Part 3, Guidance on the Application of Reliability Centered Maintenance
MIL-STD 1843	Reliability-centered maintenance for aircraft, engines and equipment
MIL-STD 2173	Reliability-centered maintenance for naval aircraft, weapon systems and support equipment

1 1.1

General Introduction

Preventive maintenance of a Product comprises scheduled maintenance tasks with intervals to achieve continued Product safety, law conformity, environmental integrity, operational/mission availability and best economic feasibility during a Product's in-service phase. Product integrated Built-In-Test (BIT) capacities and the collection and evaluation of condition and health monitoring data can minimize but not exclude scheduled maintenance of a Product during its in-service phase.

Preventive Maintenance Task Requirements (PMTR) with intervals result from the application of analysis methodologies like S4000P or alternative methodologies. These PMTRs must be consolidated prior to the data entry into the LSA database.

Within the LSA database, all PMTRs must be allocated to effective task packages with certain numerical interval values and interval types in order to reduce the total maintenance effort for a Product. Packaging solutions and rules must be in accordance with this chapter. When the packages are complete, a Maintenance Task Analysis (MTA) must be carried out for each of the final packages. Refer to [Chap 12](#).

1.2

Objective

The objective of this chapter is to define the process and rules for developing a Product's common Operator Maintenance Plan (OMP) for carrying out the resulting scheduled maintenance tasks originally required by the PMTRs.

1.3

Scope

This chapter describes the process for developing the common OMP for a Product based on the PMTRs with their original numerical interval values and original interval types. The traceability from each PMTR and its additional task-specific parameters to the resulting scheduled maintenance task within a scheduled task package of the common OMP is essential. If there are any updates, changes and/or supplements to the Product design or in case of application of an In-Service Maintenance Optimization (ISMO, refer to S4000P), the process described in this chapter must be applied again prior to any data transfer to and/or update to the LSA database.

Based on the proposal of task packages provided by the Product manufacturer and on user inputs and decisions (including usage data), a user-specific OMP must be elaborated in a subsequent work step. This user-specific OMP is the prerequisite for preparing the Product's technical publications. Refer to S1000D.

In addition to the above mentioned process an MTA of all scheduled maintenance task packages must be conducted to determine the more realistic scheduled Product maintenance effort. For example, identical subtasks performed in multiple scheduled tasks of the same maintenance task package can be eliminated.

2

Development of a PMTR

Analysis specifications such as S4000P, DEF-STAN 00-45 Part 3, ATA/A4A MSG-3, MIL-STD 1843 or MIL-STD 2173 contain three analysis methodologies covering the complete Product under analysis in parallel to Product development activities:

- System analysis
- Structure analysis
- Zonal analysis

Note

The following nomenclature and wording used in this chapter is based on S4000P.

Each of these analysis methods methodologies develops applicable and effective PMTRs with numerical interval values and interval types.

S4000P also describes interfaces between the analysis methodologies, such as:

- Reuse of Failure Cause (FC) assessment logic from the system analysis also in Zonal Analysis Modules (ZAM)
- Harmonization of General Visual Inspections (GVI) with intervals on equipment/items without "stand-alone" requirements with the GVI and the interval of a related zonal inspection

2.1

System analysis

The system analysis is an analysis methodology for developing applicable and effective PMTRs for Product systems, including equipment/items. Based on a Product breakdown structure, these systems are analyzed in a structured and transparent approach, which is comprised of a

system FMECA and the application of logic diagrams. As far as the development of PMTRs fails, a feedback to design responsibles must be provided in time if the system or a part under analysis is deemed to be non-maintainable.

2.2 Structure analysis

The structure analysis is an analysis methodology for developing applicable and effective PMTRs for the mechanical structure of a Product (eg the body of land vehicles, aircraft airframes or ship hulls).

The structure analysis in S4000P defines and harmonizes PMTRs for:

- Structure Significant Items (SSI) with or without Significant Details (SD)
- Maintenance relevant structure
- Other structure (including non-critical structures)

Structure analysis takes into account:

- Results from structure fatigue analysis
- Environmental Deterioration (ED) parameters
- Accidental Damage (AD) impacts

Next to the definition of regular scheduled intervals, which are valid from the start of the in-service phase, threshold intervals can be defined for a selected PMTR. In addition the performance of a maintenance task satisfying a PMTR can be limited on sampling concepts and/or fleet leader selection. This limitation must be accepted by responsible regulatory authorities and/or the Product manufacturer.

2.3 Zonal analysis

In accordance with S4000P, the zonal analysis is composed of different instances of ZAMs:

- Standards Zonal Analysis (ZAM 1)
- Enhanced Zonal Analysis (ZAM 2)
- Lightning/High Intensity Radiation Fields (L/HIRF) Analysis (ZAM 3)
- Additional analysis modules depending on the individual Product under analysis (ZAM X)

Depending on the Product type and its usage scenario, both a complete set of ZAMs and the corresponding content must be defined and documented, for example, in a Policy and Procedure Handbook (PPH).

Following the analysis logics in S4000P, all resulting PMTRs are harmonized at the end of each zonal analysis.

3 Thresholds, intervals and trigger events for a PMTR

3.1 General

For a single PMTR, different types of scheduled thresholds, intervals or trigger events can be selected:

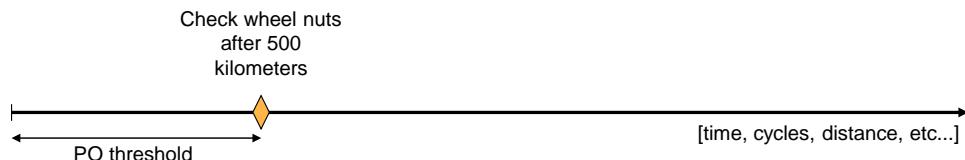
- Perform Once (PO) thresholds
- Periodical (PE) intervals
- Combination of PO thresholds and PE intervals
- Trigger events in combination with PE

The selected PMTR type must be documented at the respective LSA candidate in the LSA database.

3.1.1 PO threshold

A task satisfying a PMTR that is defined to be performed only once is characterized by a PO threshold. For example, a maintenance task to check the wheel nuts after a certain driving

distance or to check the torque of special screws after a certain time of Product usage. Refer to [Fig 1](#).

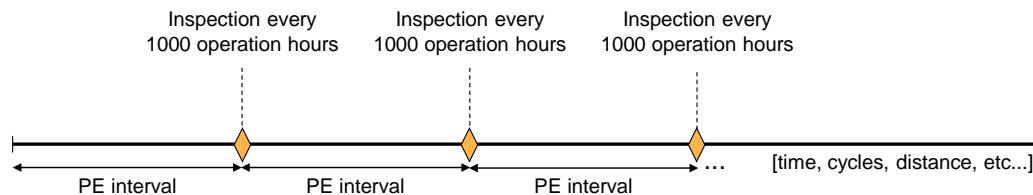


ICN-B6865-S3000L0043-002-01

Fig 1 PO threshold

3.1.2 PE interval

A task satisfying a PMTR that has to be performed repeatedly during the life time of a Product is characterized by a PE interval. For example, a periodic inspection or a periodic servicing task, such as an engine oil change. Refer to [Fig 2](#).

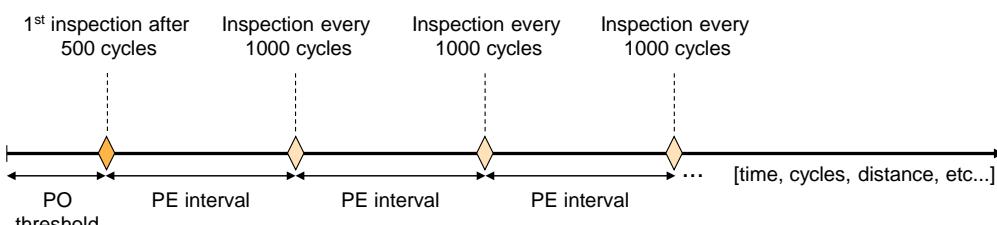


ICN-B6865-S3000L0044-002-01

Fig 2 PE interval

3.1.3 Combination of PO threshold and PE intervals

A combination of PO threshold with PE intervals is a common Product maintenance practice, for example, for Product structures. Both interval data must be documented in the LSA database. The start of the initial PE interval depends on the maturity of the PO threshold and the performance of the corresponding maintenance task. Refer to [Fig 3](#).

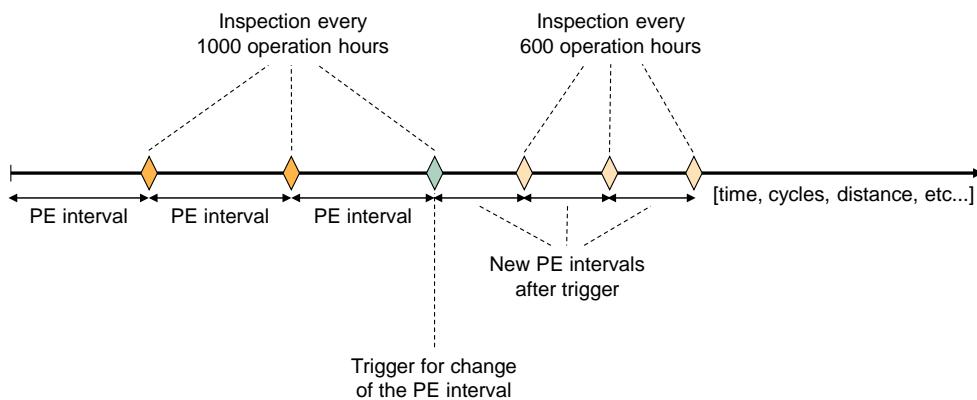


ICN-B6865-S3000L0045-002-01

Fig 3 PO threshold and PE intervals dependent from each other

3.1.4 Trigger events

A further approach to the fact that intervals can vary over the complete Product life cycle, is the inclusion of a trigger event. A trigger event determines a period of scheduled maintenance at a certain interval and starts with another time period with updated intervals and/or with updated maintenance activities. Refer to [Fig 4](#).



ICN-B6865-S3000L0046-002-01

Fig 4 Triggers

A typical use of a trigger event is the consideration of familiarization, learning phases and phases after the achievement of a certain life of a Product, when deterioration effects can force the implementation of a modified scheduled maintenance concept. For example, the In-Service Maintenance Optimization (ISMO) process described in S4000P can justify a trigger event. Refer to S4000P.

Note

A trigger event is also able to terminate a PMTR. That means no preventive follow-on task will be performed after the trigger event.

3.1.5

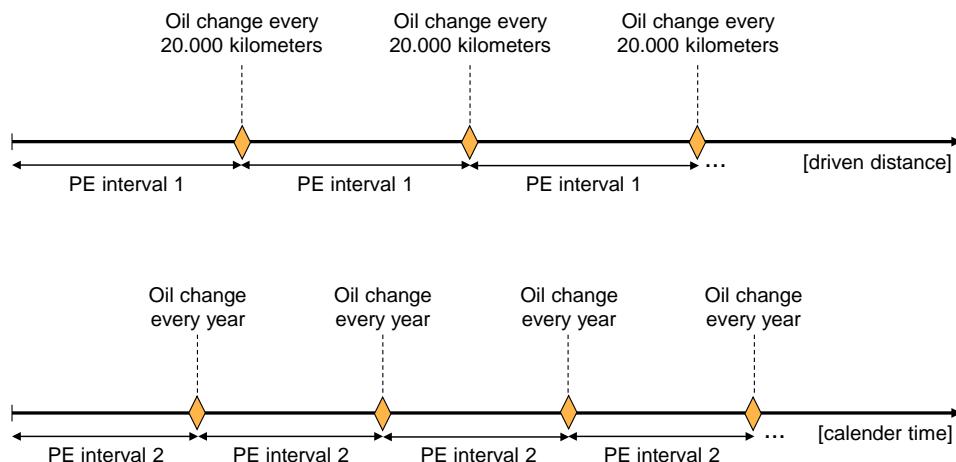
More than one PE interval in parallel

Depending on the deterioration process impacting equipment/items and leading to FC, applicable and effective PMTR can be limited to more than one interval type with different interval values.

In many cases, analysts define, for example, a usage based interval type in parallel to a calendar based interval type. Depending on the individual usage scenario, one of these interval types with the corresponding interval value triggers the performance of the preventive maintenance task prior to the other one. In this case, the PMTR that comes first is the relevant one.

Example:

A Product manufacturer determines that the engine oil of a vehicle engine either after 20.000 kilometers driven distance (PE interval 1) or after 1 calendar year (PE interval 2) needs to be replaced. The engine oil is deteriorated by both interval types by different impact parameters. If the Product's driven distance is, for example, only 8000 kilometers per year, the calendar based interval (1 year) launches the scheduled oil replacement. If the Product's driven distance is already 20.000 kilometers after only 6 months, the usage based interval (20.000 kilometers) launches the scheduled oil replacement. Refer to [Fig 5](#)



ICN-B6865-S3000L0111-001-01

Fig 5 More than one PE interval for a preventive maintenance

3.2

PMTR with threshold, intervals or trigger events from other sources

In addition to the identification of PMTRs based on S4000P analysis methodologies, there are further sources to justify a PMTR such as:

- Certification Maintenance Requirements (CMR) including (national/international) law-based requirements and requirements from regulatory authorities
- Requirements from Product safety analysis/Fault Tree Analysis (FTA)
- Fatigue related structural inspections
- Original Equipment Manufacturer (OEM) recommendation/requirements (often a prerequisite for warranty)
- Best engineering judgment
- Experience from other projects
- Individual user selected PMTRs
- Recommended scheduled activities during storage periods

4

PMTR transfer to LSA tasks

4.1

Transfer of PMTR out of system analysis

PMTRs identified in a system analysis are allocated to the Breakdown Element Identifier (BEI) of the system under analysis itself or to an equipment/item on a lower hierarchy level of the Product breakdown structure. The relevant system or the equipment/item for the PMTR automatically becomes an LSA candidate.

4.2

Transfer of PMTR out of structure analysis

To document PMTRs for structural items in an LSA database, it can be necessary to extend the Product breakdown for these identified structural items including limited areas or SD on those structural items. The documentation of an SD within Product structure in an LSA database requires the generation of artificial breakdown element. These additional LSA candidates must be introduced to enable the correct assignment of the identified PMTR for these structural areas. Refer to S4000P.

4.3

Transfer of PMTR out of zonal analysis

PMTRs identified during the zonal analysis can either be allocated to:

- the 3-dimensional zonal areas of the Product under analysis (Product zones)
- selected equipment/items of Product systems
- selected structural items of a Product

Product zones can comprise equipment/items from different systems and structural items of a Product. It is possible that the selected Product zones only contain structural items.

The LSA database must provide a BEI for each Product zone identified in an applicable Product zonal plan. In general, each zonal BE becomes an LSA candidate because of the standard zonal analysis (ZAM 1) that defines a GVI with a scheduled interval for each Product zone. Exceptions must be justified and agreed to by the manufacturer or the responsible authorities.

The PMTR that are allocated to selected equipment/items of Product systems or structural items of a Product are dealt with in the same way as the PMTR resulting from a system analysis or from the structure analysis.

4.4 LSA activities for a transferred PMTR

All PMTRs allocated to LSA candidates are subsequently subjected to an MTA. Refer to [Chap 12](#).

As described in [Para 2](#), a PMTR is developed on analytical basis. The analysis methodologies in S4000P use selection logics to identify one or more applicable and effective preventive maintenance task types. After a harmonization and consolidation phase by analysts and responsibles for design, the remaining maintenance task types for systems, structure and zones must be documented at LSA candidates in the LSA database.

For each identified PMTR the "worst-case" Functional Failure Effect Code (FFEC) possible on Product system level must be documented in conjunction with the PMTR. This FFEC categorizes the criticality of the Functional Failure Effect (FFE) if the FC occurs without the performance of the maintenance task satisfying the PMTR.

In those cases where a subsequent MTA for a single PMTR detects potential maintenance problems at a later analysis stage, feedback to the analyst responsible for the PMTR must be given to clarify the origin of the assumptions related to task applicability and effectiveness for:

- PMTR and FFEC allocated to safety or leading to a conflict with law/environmental integrity, immediate feedback with the urgent need for clarification
- PMTR and FFEC allocated to mission/operation and economy, feedback and request for clarification depending on contractual requirements between user and manufacturer

5 PMTR packaging

5.1 Preparation of the PMTR packaging

To start the correct packaging process for all identified PMTRs, the Product manufacturer must evaluate in conjunction with the user if the Product is used within the usage parameters taken as basis for Product design and development. Additionally, processes and rules for packaging the PMTRs must be determined and documented in, for example, a specific PPH.

The packaging requires a master interval, which must be determined for each complete task package. The master interval can be based on:

- Product usage parameters
- Calendar based parameters
- Combination of Product usage and calendar based parameters

Examples for Product usage parameters are:

- Operating hours
- Start cycles
- Activations
- Distance covered (eg, for vehicles)

If calendar based master intervals are selected, the real usage intensity of the Product can vary without an impact on the scheduled Product maintenance effort.

Note

The real usage intensity is better represented by usage oriented master intervals. For example, the real Product usage can significantly differ between individual users.

Depending on the individual PMTRs and the FCs and FFEs they focus on, more than one interval type with numerical interval value can be applicable and effective. In those cases the interval that is exceeded first becomes the relevant one for the predicted scheduled maintenance.

5.2

Development of an operator maintenance program

As described in [Para 2](#) and [Para 3](#), the development of PMTRs, the harmonization of these requirements and a consolidation with other PMTR sources (mainly CMR) is subject of an analytical process on basis of S4000P or of alternative analysis methodologies.

According to [Para 4](#), all resulting PMTRs with original numerical interval and/or threshold values and original interval types must be entered into the LSA database at the BEI of the affected systems/equipment, structural items and/or zones. A PMTR can have only one single interval (type and value) or two or more combinations of intervals (type and value). If more than one interval is defined, the PMTR is initiated by the one that comes first due to the corresponding usage scenario. Refer to [Para 3.1.5](#).

Each PMTR with original interval types and numerical interval values must be analyzed by the MTA to determine the Maintenance Level (ML) and the required resources such as personnel, support equipment, spares, consumables, technical publication and facilities. Analysis results must be documented within the LSA database.

After complete documentation of the PMTRs and the confirmation of validity and completeness in terms of Product built-standard/configuration, the intervals of master task packages (types and numerical values) for the later Product maintenance program/Operator Maintenance Program (OMP) must be defined. It is recommended that the Product user is involved in this activity. Refer to [Para 5](#).

Interval types of single PMTRs which are different from the selected interval type of the corresponding master task package must be adapted. For this purpose a conversion based on the Product usage scenario must be performed to adapt the PMTR interval types to the master task package interval type.

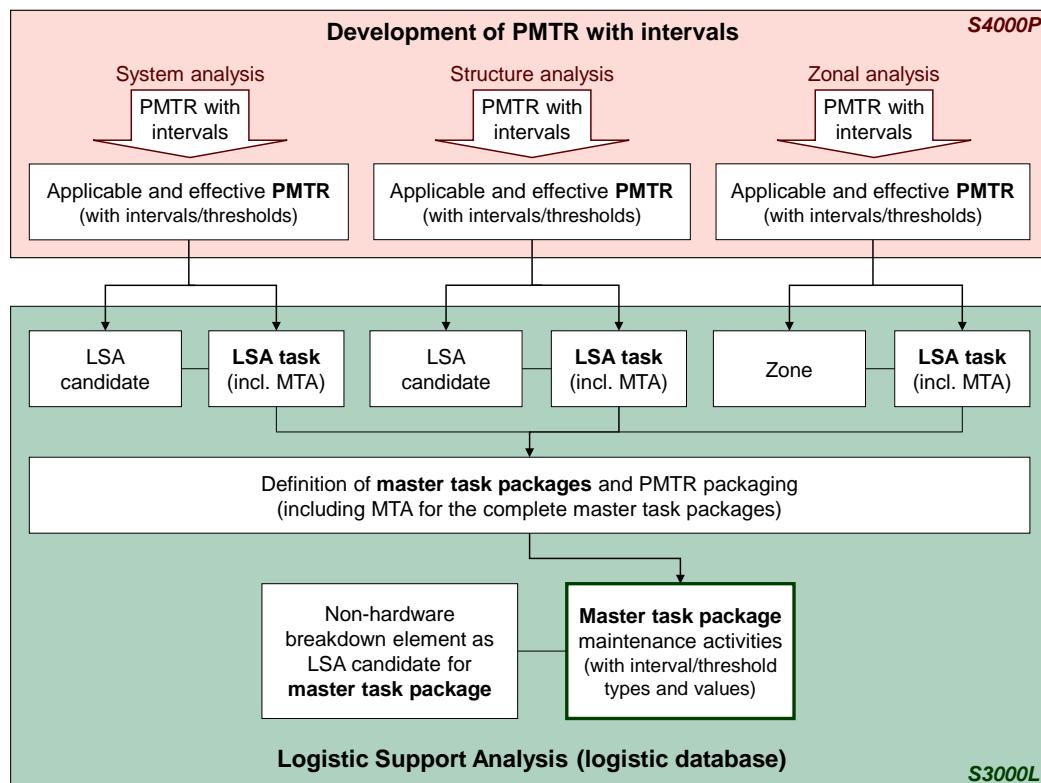
For individual Products it can be necessary to define multiple packaging concepts with different master interval types (eg calendar based or usage oriented) to allow different users an individual selection of in-service Product maintenance.

If a user intends to deviate from the baseline Product usage scenario, all calculation parameters used to convert intervals from one interval type to another interval type must be checked and, where necessary, adapted.

It is recommended that at a minimum the following aspects are documented in a Product manufacturer guideline or PPH:

- Usage scenario and deviations
- Decisions/predictions about interval types of the master task packages
- Calculation rules for interval adaption
- Rules for the PMTR allocation to a master task package
- Responsibilities

An overview of the entire process starting from identifying PMTRs within an analytical process (eg, based on S4000P) to a final harmonization in form of Maintenance, Repair and Overhaul (MRO) master task packages is shown in [Fig 6](#).



ICN-B6865-S3000L0066-003-01

Fig 6 Process overview from PMTR via LSA tasks to master task packages

The individual PMTR documented for LSA candidates and/or Product zones and the allocation to one or more master task packages must be linked together in the LSA database. This is the prerequisite for traceability as well as for quick updates, reporting and evaluation. The maintenance task types (eg, functional test) of the LSA tasks will not change when being transferred into one or more master task packages with their master intervals. However, interval types and/or numerical values of the task intervals are subject to adaption and can change.

Example:

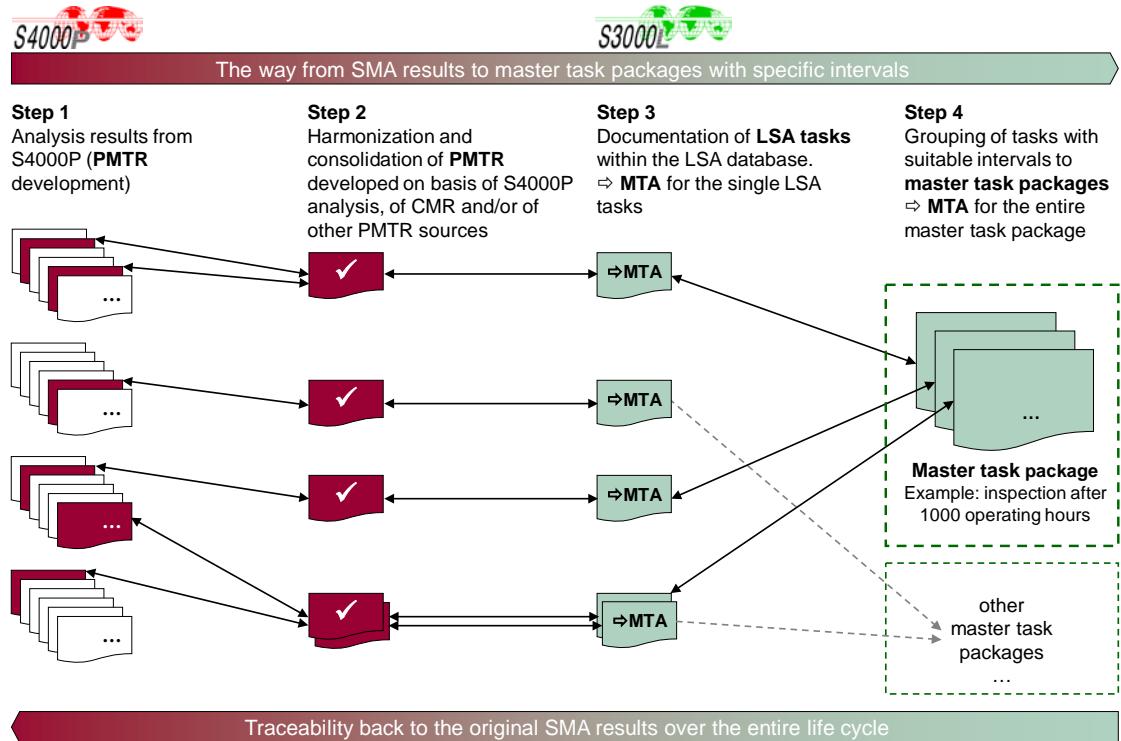
A safety relevant PMTR justifies a visual inspection on a hydraulic actuator every 600 Operating Hours (OH). Therefore the PE interval is 600 OH without any PO threshold. The design usage scenario is defined as:

The Product is planned to be operated maximum 1000 OH in one calendar year. The interval type for the master task package is determined calendar based by the user with a small inspection package every 6 months and a full inspection package every calendar year. Due to the PMTR category "safety-relevant" in this example, the numerical size of the scheduled interval is only allowed to be lowered, resulting in a higher task frequency.

Based on this limitation, the first scheduled task must be performed during master task packages that don't exceed the PE interval of 600 OH or the corresponding calendar intervals. The scheduled task must therefore be performed every 6 months, which correlates with the 500 OH. The selection of a task which is performed every year, which correlates with the maximum of 1000 OH, is not permitted because of a clear interval excess for a safety relevant PMTR.

The original PMTR at the LSA candidate in this example is a detailed visual inspection on the hydraulic actuator every 600 operating hours.

The result after calculation and packaging process in this example is a detailed visual inspection on a hydraulic actuator every 6 months.



ICN-B6865-S3000L0067-003-01

Fig 7 Traceability of scheduled maintenance tasks with intervals

[Fig 7](#) shows the complete traceability path from original PMTR developed on basis of S4000P to the point of LSA tasks and the final master task packages for Product MRO. Finally, the maintenance task packages for Product MRO will be described in the Product's technical publication, refer to S1000D. They are the binding activities which must be performed during the Product in-service phase.

For later Product maintenance optimization or evaluation purposes, the ISMO process described in S4000P can be applied.

5.3

5.3.1

Packaging process for PMTRs

General aspects

Prior to starting the interval packaging process, the LSA database must be checked to ensure that all PMTRs are approved and released for further interval packaging activities. The PMTRs must represent all valid Product configurations including variants of the Product under analysis.

Effective and applicable PMTRs, which are documented as LSA tasks, must be assessed to develop a scheduled maintenance program/Operator Maintenance Program (OMP) in a correct and effective way.

This process aims to ensure an effective packaging of the PMTRs taking into account at least the following aspects:

- A PMTR can become relevant in more than one master task package (eg, a maintenance task satisfying a PMTR is defined to be performed prior and after each Product operation day)
- Each identified effective and applicable PMTR has its own allocation to a "worst-case" criticality of the FFE, to which the PMTR related FC is allocated. This criticality category

- enables decisions in terms of interval extensions or reductions during the interval packaging process.
- One identified effective and applicable PMTR can have one or more different interval types (eg, operating hours, cycles, calendar time, distances) and different numerical interval values. This requires the use of conversion factors for interval calculation and allocation of PMTR to master task packages.
- PMTR can be performed only on one or more predicted MLs
- PMTR can require specific training and experience of maintenance personnel
- PMTR can require specific support and test equipment
- PMTR can require spares and/or consumables
- PMTR can require specific facilities

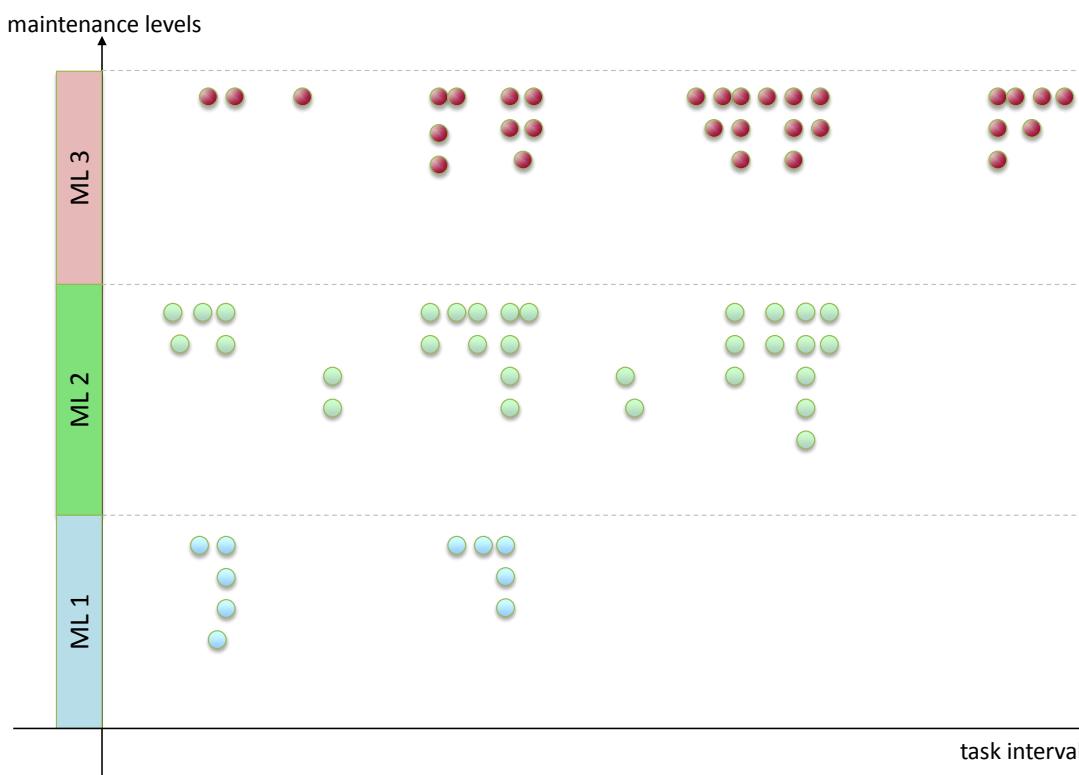
After all valid PMTRs are decided as scheduled maintenance task in master task packages, a further MTA for each task package must evaluate whether:

- potential interdependency of single activities/maintenance tasks exists (eg, one maintenance task must be closed and confirmed by an inspector, before another maintenance task can start)
- a potential time saving will result by performing time consuming tasks for gaining access only once (eg, after opening panels and doors, all tasks are performed within the corresponding area before closing the panels and doors again)
- task packaging must consider the possibility of performing tasks in parallel to enable a more realistic estimation of the duration and the effort of a complete scheduled maintenance task package

5.3.2

Impact of maintenance level on master task packages

After completion of the S4000P analysis, a set of PMTRs with different interval types and numerical interval values is identified, harmonized with other PMTR sources, consolidated and documented within the LSA database. PMTRs with PE intervals and PO thresholds (if they exist) are significantly influenced by different ML, identified during the MTA of the individual PMTR. An example for a first distribution of PMTRs is given in [Fig 8](#).



ICN-B6865-S3000L0110-002-01

Fig 8 Initial distribution of PMTR depending on maintenance level

Each bullet in [Fig 8](#) represents one PMTR with a specific interval type (eg, operation hour, cycle, month) and numerical interval value to be performed at predicted MLs.

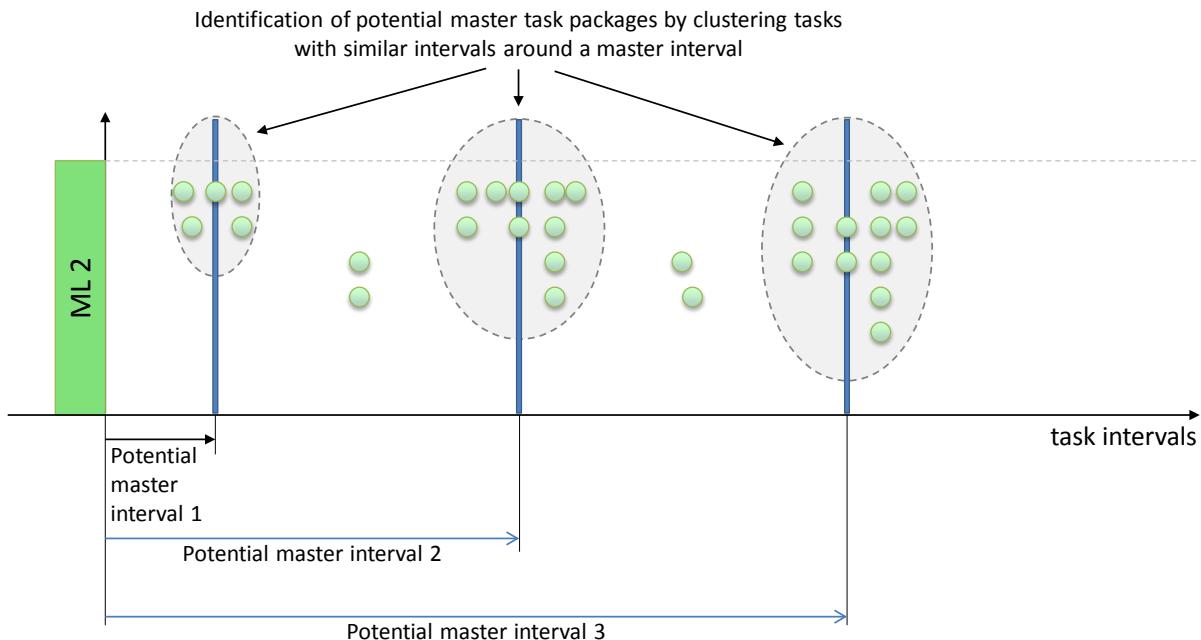
Note

Examples for ML definitions are given in [Chap 11](#).

The analyst must consider the following aspects during the harmonization process:

- The interval types of master task packages (eg, based on calendar time) must typically be defined in accordance with the Product usage requirements of the user. In practice, it is necessary to harmonize all identified PMTRs around an MRO plan which allows only specific interval types for scheduled maintenance.
- PMTR interval types for Product components need to be adapted as far as possible to the leading maintenance task intervals of the complete Product by using corresponding conversion factors for a conversion calculation
- Harmonization of PMTRs often requires a change of original interval types and/or numerical interval values, originally determined by the S4000P analysis including the subsequent harmonization and consolidation of those results
- PMTR harmonization must be performed over the different MLs of the Product. PMTRs can be shifted to another ML dependent on the capabilities of each ML.

When starting the interval packaging, the distribution of the PMTRs within one ML can already show a concentration of some PMTRs within limited ranges of intervals. To support the determination of the master intervals, the analyst can use these ranges of similar intervals. Refer to [Fig 9](#).



ICN-B6865-S3000L0112-001-01

Fig 9 Clustering of PMTRs to support the identification of master intervals

5.3.3

Predefined master task packages

The LSA database can also be used and evaluated for all PMTRs/LSA tasks determined (eg, for each Product operating day).

In many cases, three predefined master task packages are applicable for a PMTR allocation:

- Before starting the Product operational/mission phase
- After an intermediate stop of Product operation or mission (prior to the restart, if applicable)
- At the end of a Product operation or mission

In addition, further predefined master task packages allow the allocation of PMTRs.

For example, the Product overhaul/depot level maintenance is defined after 15 years of the 30 years Product usage period. That covers all PMTRs in LSA with the interval of 15 years or with an allocation to Product overhaul/Depot Level Maintenance (DLM).

5.4

5.4.1

Adaptation of a PMTR interval type and/or numerical interval value

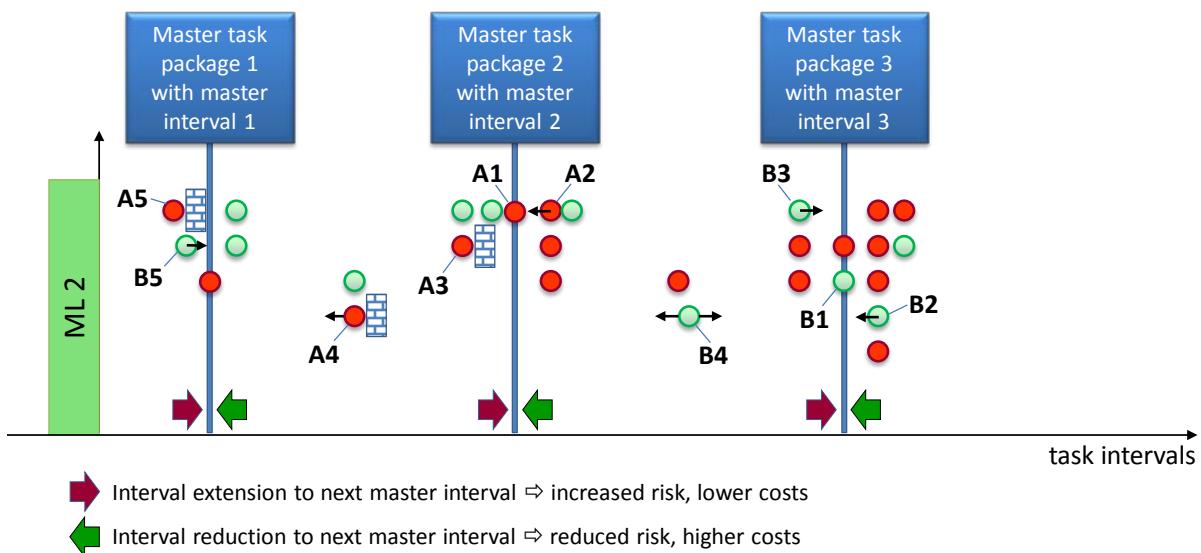
General aspects

Potential changes of PMTR interval type and/or numerical interval values must follow clear rules which must be traceable for regulatory authorities and/or departments responsible for quality at the Product manufacturer. For this purpose, it is strongly recommended that a guideline/PPH be prepared and agreed by the involved parties.

After the final determination of the ML and the master intervals for each master task package, the adaptation of individual PMTR intervals must be performed. In [Fig 10](#), examples are given for use cases for different interval change options. The general consequences which must be considered, are:

- Interval extension causes higher risk of FC occurrence, but reduces maintenance costs
- Interval reduction causes lower risk of FC occurrence, but increases maintenance costs

- Task related to safety, ecological or legal restrictions (examples A1-A5)
- Task related to operational or economic restrictions (examples B1-B5)



ICN-B6865-S3000L0047-002-01

Fig 10 Principles for interval modification

Some PMTRs can have the same interval type and numerical value as the master task package interval. But some of the PMTR intervals types need to be changed. Most of the interval values of PMTRs need to be extended or reduced. However, it is important to remember that extension of intervals and the resulting reduction of the maintenance task frequency can cause a higher risk of FC occurrence.

The examples within [Fig 10](#) consider different criticality of a corresponding FFE for a PMTR and the different needs for adaptation (extension/reduction):

- The examples A1 to A5 are relevant for PMTRs, which prevent FFE related to safety, law conformity or environmental integrity
- The examples from B1 to B5 are relevant for PMTRs, which prevent FFE related to operational/mission availability and/or economic impacts

For each example, adaptation rules are described. Refer to [Table 2](#).

Table 2 General interval adaptation rules

Example	Adaptation rules
A1	PMTR interval equal to the master interval, no adaptation required
A2	PMTR interval higher than the nearest master interval and within a range close to the nearest master interval. Adaptation of interval is possible and agreement by regulatory authorities is expected. Increase of costs with high probability within an acceptable dimension, user involvement is recommended.
A3	PMTR interval lower than the nearest master interval and within a range close to the nearest master interval. However, adaptation of interval not allowed because of the related FFE criticality.

Example	Adaptation rules
A4	PMTR interval outside a close range to the next higher or lower master interval. Adaptation only to a lower master interval is possible. Agreement by regulatory authorities is expected. Increase of costs due to a significant higher task frequency must be considered, user involvement is recommended.
A5	PMTR interval lower than the nearest master interval and within a range close to the nearest master interval, which is the lowest master interval available. However, direct adaptation of this interval not allowed because of the related FFE criticality. This situation requires additional analysis to identify a smaller master interval (eg, daily inspection), to recalculate the interval by safety department or, in worst case, consideration of Product design change.
B1	PMTR interval equal to the master interval, no adaptation required
B2	PMTR interval higher than the nearest master interval and within a range close to the nearest master interval. Adaptation of interval possible. Increase of costs with high probability within an acceptable dimension, user involvement is recommended.
B3	PMTR interval lower than the nearest master interval and within a range close to the nearest master interval. The related FFE criticality allows adaptation of the interval, but the increased risk for the occurrence of an FFE must be agreed by the user.
B4	PMTR interval outside a close range to the next higher or lower master interval. Adaptation to either a lower or a higher master interval possible. Increase of costs because of a significant higher task frequency in case of interval reduction must be considered and must be agreed by the user. Increased risk for the occurrence of an FFE because of a significant lower task frequency must be considered and agreed by the user.
B5	PMTR interval lower than the nearest master interval and within a range close to the nearest master interval, which is the lowest master interval available. Adaptation to the lower master interval possible. Increased risk for the occurrence of an FFE because of a lower task frequency must be considered and agreed by the user.

5.4.2

Summary of PMTR interval adaptation

Intervals of PMTRs, which prevent FFEs related to safety, law conformity or environmental integrity must not be extended. If any PMTR interval of these FFE categories is below the lowest grouping interval for the specific maintenance level, the task needs further investigation, which can result in:

- Introduction of a new master task package with a lower interval value
- PMTR modification (change of PMTR type or recalculation of numerical value)
- Maintenance level change
- Product design change requirements

Intervals of PMTRs, which prevent FFEs related to operational/mission availability or economic impact can be modified for task packaging in both directions, extension or reduction. All modifications must be agreed by the user. The risk of economic loss or additional downtime must always be balanced against potential cost savings.

Note

The ISMO process described in S4000P takes into account these rules based on the FFE category for each individual PMTR. The Product in-service experience in combination with

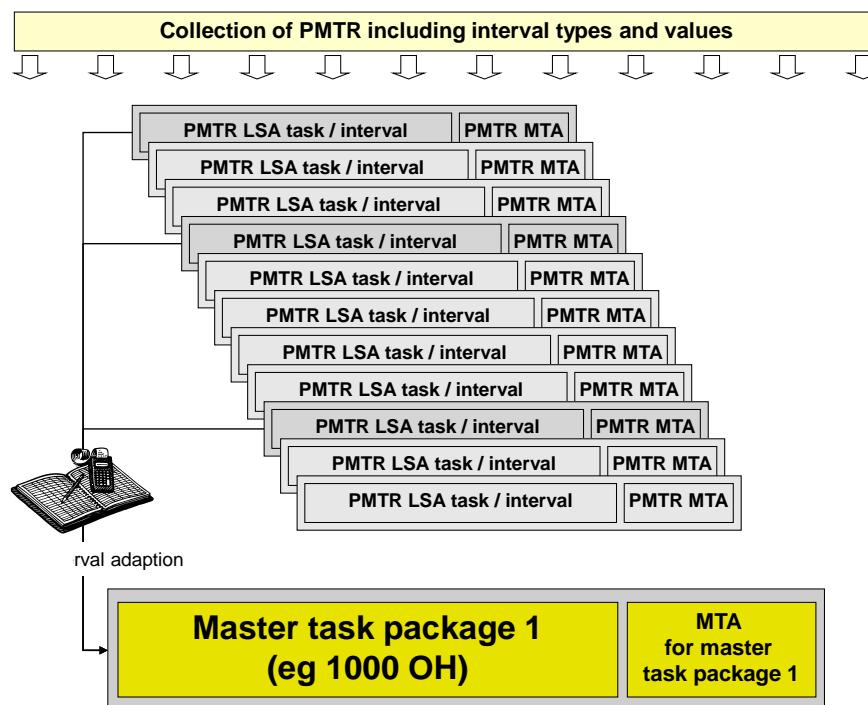
ISMO allows an easier adjustment and optimization of all PMTRs linked to FFE on operation, mission and economy.

For FFE on safety, law conformity or environmental integrity, no in-service data/experience can be expected. The optimization of those PMTRs must consequently be based on other investigations, calculations and/or further analysis.

5.5

Maintenance task analysis for master task packages

After allocation of all PMTRs to master task packages, an MTA for the complete packages is required. The master task packages can contain a large number of single PMTRs with or without adaptations of interval type and/or numerical interval value. Refer to [Fig 11](#).



ICN-B6865-S3000L0048-002-01

Fig 11 Integration of single PMTR LSA tasks into master task packages

These scheduled maintenance tasks in the master task package are not always performed sequentially, but can also be performed in parallel or partially interdependently. Required resources such as personnel and support equipment must be harmonized for the complete package and are not just a simple collection from the single LSA tasks.

If a master task package is finalized, the original PMTRs (documented as LSA tasks against the corresponding LSA candidates) are no longer the basis for logistic evaluations, because all master task packages become now subject to those calculations. However, the original PMTRs must be kept and maintained in the LSA database because of the need for data updates and traceability.

At any stage of the project, it must be possible to trace back from an individual scheduled maintenance task defined in a master task package back to the original PMTR which is documented at the LSA candidate in the LSA database. Refer to [Fig 7](#).

It must also be possible to trace back from each PMTR within the LSA database to the corresponding PMTR source (eg, based on S4000P, MSG-3, RCM, CMR).

These traceability requirements ensure the complete consistency of the scheduled Product maintenance, starting from the first analysis steps, followed by activities from S3000L to the final generation of the technical publications for a Product.

In addition this allows the application of the ISMO process for continuous optimization of Product maintenance including the Product's Life Cycle Costs (LCC). Refer to [Chap 14](#) and S4000P.

6 In-service maintenance optimization

With the ISMO process, S4000P provides a logic based approach for analysis and optimization of all individual scheduled maintenance tasks with intervals predicted by the OMP for a Product.

The ISMO process takes into account the experience accumulated during the Product in-service phase from the individual user and/or from other users of the same or a comparable Product.

After having completed an ISMO process, the updated set of PMTRs must be processed like the original set of PMTRs available prior to Product entry into service. In accordance with the process described in this chapter, the OMP can be updated.

For more information and details related to the ISMO process, refer to S4000P.

Chapter 11

Level of repair analysis

Table of contents

	Page
Level of repair analysis	1
References.....	1
1 General	2
1.1 Introduction	2
1.2 Objective.....	2
1.3 Scope.....	2
2 LORA Candidate selection aspects.....	2
2.1 Candidate selection	2
2.2 Repair or discard decision	3
3 Identification of optimum maintenance level by ELORA	3
4 Data gathering for LORA	4
4.1 Determine parts hierarchy	4
4.2 Determine LORA candidates.....	4
4.3 Collect unit prices	4
4.4 Collect maintenance related cost data	4
4.5 Other maintenance relevant data	6
5 Performance of ELORA and preparation of results.....	7
5.1 ELORA baseline run.....	7
5.2 Sensitivity analysis.....	7
5.3 Maintenance solution recommendation.....	8
5.4 LORA Report	8
5.5 Maintenance solution documentation in the LSA database	8
6 Example for maintenance level definitions	9
6.1 General	9
6.2 Level 1	9
6.3 Level 2	9
6.4 Level 3	10

List of tables

1 References	1
2 Overview of typical cost elements	5

References

Table 1 References

Chap No./Document No.	Title
Chap 3	LSA Business process

1 General

1.1 Introduction

Level of Repair Analysis (LORA) is an analysis method to assist in establishing an optimized maintenance solution based on a support philosophy agreed with by the customer. This includes the determination of where repairable LSA candidates are removed, replaced, repaired or discarded. Repair decisions must consider both economic and non-economic factors such as costs, reliability values, maintainability/testability constraints or availability goals for the Product. The results of the analysis influence maintenance tasks and corresponding task resources like support equipment, personnel or spare parts.

The two main aspects of the LORA process are the technical assessment and commercial (costs) balancing. The mainly commercial approach, It is recommended to perform an Economical LORA (ELORA) in conjunction with LCC analysis activities, because the costs for maintenance activities are a crucial part of the complete LCC. Only the combination of technical and commercial aspects will lead to a proper LORA result.

The LORA is a two-step process:

- Determination of whether the Item under Analysis (IUA) is a repair or discard item (supporting the repair/discard decision by the customer)
- Identification of the optimum maintenance level for the required maintenance tasks, also called ELORA

These two steps are described more detailed in [Para 3](#) and [Para 4](#).

The requirement for a LORA is documented in the LSA PP or similar documents. Depending on the individual phase of a project, a LORA can be performed for different purposes using different methods.

The LORA process can be repeated as often as required during the design and the in-service phases as the design matures. This determines whether the maintenance solution remains as predicted or must be changed due to design changes or other factors. The results of the LORA process can also be used to drive supplier selection by illustrating various options, and can lead to the adoption of phased support.

1.2 Objective

The purpose of a LORA is to provide the customer with decision guidance for the selection of the best balanced maintenance solution. Different customers can come to individual decisions depending on specific project requirements. In the early stage of a project, LORA aspects must be considered in order to influence the design (eg testability features, modularity, accessibility requirements) as well as to support customer strategy decisions (eg 2-level maintenance strategy, single source principle).

1.3 Scope

This chapter is directed at contractor and customer logistics personnel who perform LORA activities. Results of the LORA will be documented in appropriate LORA reports. It is recommended to use the LSA database to document the derived maintenance solution per LSA candidate.

2 LORA Candidate selection aspects

2.1 Candidate selection

The identification of potential LORA candidates can be based on the LSA Candidate Item List (CIL), which is established in accordance with the project's LSA candidate selection process. For details, refer to [Chap 3](#).

The first step in the LORA is to decide whether an LSA candidate is potentially repairable. If yes, a decision must be made on, which of these LORA methods is most feasible to use:

- Best engineering judgment only
- Simplified LORA (based on information collection and consideration of values)
- LORA based on mathematical models (supported by commercial software packages)
- Simulation (supported by commercial software packages)

In either case, the justification for the selection of the method to be used must be documented.

In addition to these factors, other additional constraints or fixed customer decisions can include:

- Exclusions concerning repair due to danger of destruction or hazardous/unsafe situation
- LORA relevant input by system integrator or Original Equipment Manufacturer (OEM)
- Cut off values or design driven influences (a set of predefined cut off values can reduce the number of ELORA candidates). These cut off values are usually based on technical and economic factors such as low task frequency, low unit price, repair costs.
- Customer imposed maintenance constraints external to the LORA
- Existing maintenance infrastructure of the customer derived from ORD/CRD
- Customer requirements derived from ORD/CRD
- Contractual requirements or LSA GC decisions
- Clear understanding of the requirements of LORA by other disciplines

These external influences or requirements must be addressed, discussed and agreed at the LSA GC.

2.2

Repair or discard decision

A Repair/Discard decision is the prerequisite to any ELORA. It can include non-economic as well as economic factors. It is, in effect, a screening process aimed at minimizing the number of ELORA candidates using a logical process of elimination. Each LSA CIL has numerous items that are potential ELORA candidates. The Repair/Discard decision must be made in order to identify which of these items are candidates for further ELORA analysis. This is achieved by determination of the repair feasibility for the items.

For each potential item, crucial technical questions must be answered first:

- Does the design of the item allow repairs in general?
If so, this item is a potential ELORA candidate.
- Is the repair of the item limited to certain maintenance levels?
If so, this item is a potential ELORA candidate. The analysis must be limited to the possible maintenance levels.
- Does the design of the item disallow repairs in general?
If so, this item must be removed from the ELORA candidate list. However, the proper maintenance level for replace and discard must be identified and documented carefully.

This decision, and the logic leading to this decision, must be documented carefully for future traceability. For items that have been identified as potentially repairable, the next stage is to further determine the method of performing the ELORA. This stage presents the opportunity to have any external influences/requirements taken into account (eg existing maintenance solutions used by the customer, cut off values or design driven influences).

3

Identification of optimum maintenance level by ELORA

The repair/discard decision process results in a list of remaining ELORA candidates. These candidates are identified as potentially repairable. The objective of the ELORA is to determine the optimum maintenance solution for these candidates. This is carried out using mainly economic cost factors such as spare parts, support equipment required to diagnose and perform repairs, personnel, training, facilities, technical documentation, and Packaging, Handling, Storage and Transportation (PHST). Additionally, non-economic factors such as availability requirements, reliability, and maintainability can be taken into account.

There are many different mathematical ELORA models available and supported by different commercial software packages. An alternate approach that is often used within a simplified LORA, is to develop a project specific or tailored model that addresses the requirements and constraints (for both customer and OEM) such as available infrastructure, personnel or technical capabilities.

It is of vital importance that all LORA analyses within a project use the same LORA model. Failure to do so will lead to differing results.

4 Data gathering for LORA

It is important that close attention is paid to the collection of LORA data. A LORA data set can contain simple data such as:

- Spare parts
- Facilities
- Support equipment required to diagnose and perform repairs
- Personnel and required skill levels
- Extended training

A LORA data set can also contain comprehensive data giving in-depth details of the requirements and in particular the costs associated with those requirements.

Once the data elements have been agreed in the LSA GC, the LORA data can be gathered from a variety of sources such as drawings, technical documents, contracts, supplier information and commercial department.

4.1 Determine parts hierarchy

Failure modes in LORA are assigned to the hierarchical structure of the system. Failures at the top level of the system are assumed to be primarily caused by failures of the items (or subsystems) at the next lower indenture level. Similarly, failures of the items in the 2nd level of indenture are mainly initiated by failures of the items in the next lower indenture level. For this reason, it is impractical to perform a LORA until hierarchical relationships are defined. For Products that are currently in production, the hierarchy and subsequent LORA candidate items can be determined from existing LSA data following the Product breakdown.

4.2 Determine LORA candidates

After the parts hierarchy is established, the analyst must determine which parts are considered in the analysis. At first glance it seems logical to include all of the parts, however, non-repairable or consumable parts such as nuts, bolts or gaskets will add no value to the analysis and, therefore are not included. Candidate items can be determined by establishing a rule set based on existing repair or maintenance codes, unit price, repair level, or other criteria.

4.3 Collect unit prices

Unit prices for all items are required to fill an ELORA data model and to perform the analysis. Sources for unit prices can be LSA databases, provisioning records or other sources. Unit price for all ELORA candidate items is necessary to determine the economic feasibility of repair. It is essential that for ELORA calculation purposes the correct unit prices are used. Unit prices can differ dramatically from the production phase to the in-service phase (spare part price can be very different from production unit price). Sometimes precise values are difficult to obtain and can only be estimated.

4.4 Collect maintenance related cost data

Maintenance cost data includes labor time, training, spare parts and consumables, support equipment, facilities, technical documentation and PHST aspects. Concerning these data it must be distinguished between initial and recurring costs. These data are used in the calculation of cost at each possible maintenance level.

An overview of typical cost element types for consideration in an ELORA is given in [Table 2](#) (additional data can be required depending on the project requirements).

Table 2 Overview of typical cost elements

Cost element	Initial	Recurring
Personnel		
Labor time costs		X
Training costs (teachers and staff)	X	X
Training equipment costs (eg simulators, classroom equipment, training facilities)	X	X
It is recommended that costs of direct labor and indirect labor (eg line managers, inspectors, work preparation department) be considered.		
Spare parts/consumables		
Spare parts/consumables provision	X	X
Spare parts/consumables storage		X
Disposal costs		X
Support equipment to diagnose and repair		
Support equipment provision and replacement	X	
Support equipment maintenance		X
Support equipment upgrade	X	
Development of customized software (eg test software)	X	
Support of customized software for support equipment		X
Disposal costs		X
Facilities and infrastructure		
Building of facilities and infrastructure	X	
Maintenance of facilities and infrastructure		X
Costs of operation for facilities and infrastructure		X
Costs for modification of facilities and infrastructure	X	X
Technical documentation		
Documentation costs (eg user handbooks, maintenance and training manuals, illustrations, spare part catalogues)	X	X
PHST aspects		
Packaging costs	X	X
Stock keeping costs (eg maintenance tasks and handling during storage, special storage containers, administration of stock)	X	X
Transportation costs (transport to maintenance facilities and back to user)		X

Cost element	Initial	Recurring
Others		
Average repair costs at industry	X	X
Additional costs (documentation, administration, preparatory work and post processing)	X	X
Cost per reorder action	X	X
Cost per requisition	X	X
Disposal costs	X	X
Discount rate	X	X
Holding cost percentage	X	X
Interest rate	X	X

4.5

Other maintenance relevant data

Besides the direct costs related data there is other information of interest for ELORA calculations. This include but are not limited to:

- Time and duration information
 - Reliability information (Mean Time Between Failure (MTBF), Mean Time between Unscheduled Removal)
 - Task frequency of corrective maintenance task
 - Scheduled maintenance interval
 - Task duration
 - Logistics down time (waiting time for logistic resources)

Note

Data for repair time is often difficult to obtain, particularly for new systems. LSA databases sometimes contain this information. Some developmental systems specify desired and maximum times to repair a failure. This can be used as a repair time for items that are directly removed and replaced. For the repair of lower indenture items, it can be necessary to use the repair time for a similar item on another system.

- Availability values
 - Minimum availability of IUA (usually, operational availability is specified)
 - Availability of spare parts in stock
 - Availability rate of personnel for maintenance activities
- Stock relevant values
 - Procurement lead time
 - Pipeline transit times
 - Repair turnaround times
- General information
 - Number and type of contractor industrial facilities
 - Number of operational sites
 - Number of system operated per site
 - Distances between sites
 - Inflation/discount rate
 - Repair policy (eg 2-level maintenance strategy)

History shows that the majority of cost influencing decisions must be made in a very early stage of the project, even if the information is very limited and the design is not yet stable. This

emphasizes that the LORA must be repeated throughout the life cycle so that the requirement for adoption of the maintenance solution can be identified in a timely manner.

5 Performance of ELORA and preparation of results

After successful collection and harmonization of data with the customer, the ELORA is performed. A simple approach is to evaluate and balance the given information by best engineering judgment. If more complex mathematical calculation models are used, a commercial software package can be selected. The data set to be collected depends on the requirements of the specific software package. The results of the calculation runs must be prepared in an ELORA report and distributed according to established rules.

5.1 ELORA baseline run

The first step of an ELORA, using a supporting software package, is the baseline run using the gathered ELORA data. This baseline run will result in a first set of information regarding costs for the individual maintenance levels. The result can be clear cut or ambivalent between the different maintenance levels. In either case, it is recommended to confirm the baseline run results by executing a sensitivity analysis.

5.2 Sensitivity analysis

A sensitivity analysis must be performed on various cost significant parameters. The results of the various sensitivity runs will indicate whether the maintenance solution from the baseline run is stable or not. Parameters used in a sensitivity analyses include:

- Major cost influencing parameters of the IUA itself (eg unit price, MTBF)
- Any parameters that are critical to cost intensive logistic requirements (eg facilities, infrastructure, support equipment, highly educated personnel)
- Any parameters using assumptions or estimates because there was no data available (eg historic or similar data from other Products)

Once the parameters for the sensitivity analyses have been selected, the next step is to establish an appropriate numerical range for the parameter. Finally, when all the requirements have been established, an ELORA is performed for each parameter variation. Changing single parameters will keep the number of runs to a minimum. Multiple parameter changes at the same time will lead to nested runs and therefore are usually to be avoided because evaluation of the results can be confusing.

The output of the sensitivity analyses can be used as a baseline to establish a preliminary maintenance solution. Additionally, it can be used to influence other logistic disciplines and/or design, in order to change their input to drive the equipment to enable an intended maintenance solution.

A sensitivity analysis is usually conducted to complement baseline results. Sensitivity runs are performed for cost significant parameters that have a low confidence level or to accomplish a trade-off study. For a sensitivity analysis of low confidence values, the most common method is to perform runs using the worst and best case data. If there is no shift in maintenance solution result between these two runs, then no additional executions using intermediate values are required.

An example of this is sensitivity for the MTBF of an item. Suppose an estimated value of 50000 hours is used in the baseline run. Preliminary analysis revealed that MTBF's of similar items range from 15000 hours to 150000 hours. The first sensitivity runs performed would include values of 15000 and 150000 hours for the MTBF of this item. If the most cost effective repair level for both sensitivity runs is the same as the baseline run, no further sensitivity analysis for this MTBF is necessary. However, if a change in repair level occurs for one or both runs, additional executions using values of MTBF between the high and low value can be necessary to identify the break points.

5.3 Maintenance solution recommendation

For a concluding maintenance solution recommendation, it is necessary to consider the cost aspects resulting from an ELORA, as well as technical aspects that can have a major impact on the recommendation of the maintenance solution to the customer (eg destruction of the IUA during critical repairs, dangerous repair tasks). Additionally, the customer's preferences must be considered.

It is also important to harmonize the maintenance activities in order to avoid a split of similar maintenance activities to different maintenance sites. For example, it is important that the allocation of a piece of electronic equipment for replacement and subsequent repair of different components are not split between industry and customer industrial sites.

5.4 LORA Report

A report documenting the analysis results must be generated. The LORA report must contain, at a minimum:

- Agreements reached between the customer and/or the contractors regarding LORA
- A brief description of the system/equipment under analysis
- Comprehensive list of any assumptions and estimations including their rationale
- Data sources
- Comprehensive list of the LORA data elements
- Input values
 - Operational environment data used for the process
 - Breakdown of IUA
 - Cost and maintenance data
- Realization of mathematical methods applied within a simplified ELORA
- Utilization of ELORA software package
- Result of baseline run
- Results and explanations of sensitivity analysis runs
- Trade-off studies
- Consolidation of technical and cost aspects
- Concluding recommendation of the maintenance solution including discard decision

It is recommended that a report structure and layout be developed early in the analysis process. Otherwise, important information and logic can be overlooked during the course of a lengthy analysis.

For good traceability, it is recommended to use a standardized format within the LORA report for the maintenance solution recommendation and related customer decision. The customer decision will be documented in the LSA database.

5.5 Maintenance solution documentation in the LSA database

The content of the maintenance solution is related to the corresponding maintenance activities as documented in maintenance tasks in the LSA database. Each task for an LSA candidate is be provided with attributes that reflect the maintenance solution. This information must include the following, at a minimum:

- Is the maintenance task to be performed scheduled or unscheduled?
- For scheduled tasks: interval or threshold data
- The maintenance level at which the task must be performed
- The place at which the maintenance tasks are to be performed
- Special remarks or warnings (eg if the Product is used under special conditions, inspection interval must be reduced from 6 to 3 months)
- References to standard repair tasks
- Data source information for traceability (eg identification of LORA report)

It is recommended that the basic data set that reflects the maintenance concept be agreed during the LSA GC, in order to enable a common understanding. Proper documentation within the LSA database enables a strong reporting ability for different aspects of the maintenance concept.

Examples:

- Complete overview of the maintenance activities at all maintenance levels
- Summary of maintenance activities performed at certain maintenance levels
- Overview of scheduled/unscheduled maintenance activities
- Expected effort at certain maintenance levels
- Support for quality checks concerning completeness and conclusiveness of the maintenance solution

6

6.1

Example of maintenance level definitions

General

This example is based on three levels of maintenance, which indicates the capability of personnel, availability of special facilities, time limits and the environmental conditions to be assumed in determining the functions to be accomplished at each maintenance level.

6.2

Level 1

The goal of level 1 maintenance is to ensure the Product remains available. This implies a fast and easy exchange of LRUs and/or the replacement of modules, performed on the Product by organizational personnel when a mal-function occurs.

Level 1 activities include:

- Servicing activities
- Usage preparation and role changes
- Pre- and post- inspections
- Functional checks
- Trouble shooting
- Preventive maintenance
- Corrective maintenance (repair by replacement and system adjustment)
- Loading of software (operational and engineering) and data retrieval
- Simple modifications

6.3

Level 2

The goal of Level 2 maintenance is to maintain the highest possible level of availability. Operating site maintenance activities are extended to the repair of subassemblies, modules and LRUs after their replacement at maintenance Level 1. Testing on test-benches or integration tests can be included. Subsequently, Level 2 maintenance can be performed either on Product or at specific repair shops.

Level 2 activities include:

- Repairs down to module and subassembly level
- Moderate structural repairs
- Major scheduled inspections
- Moderate modifications
- Technical assistance to the Level 1 organization
- Software servicing concerning engineering data
- Preservation of complete Product

Level 2 activities include corrective and preventive maintenance and specific maintenance activities that will be performed both on Product (during maintenance the Product will be unavailable for use), or at specific repair shops. Level 2 also includes tasks beyond those

accomplished at Level 1 in order to facilitate the return of the equipment to its full operational state. Level 2 maintenance will be performed at facilities capable of accommodating the maintenance tasks, including the need to use special equipment or specialized repair shops, and will be performed by appropriately trained and specialized personnel.

6.4

Level 3

The goal of Level 3 maintenance is to ensure the highest possible availability of the Product, as well as providing engineering support for operational aspects. All repairs and overhaul activities beyond Level 1 and Level 2 capabilities must be ensured. Major modifications to improve the design and/or operational activities will be prepared and, if necessary, embodied at this level.

Level 3 activities are expected to include:

- Repairs down to full reconditioning
- Repairs requiring special or rare skills or support equipment
- Major structural repairs
- Major scheduled inspections
- Extended modifications and update programs
- Technical assistance to the Level 1 and Level 2 organizations
- Software modification
- Preservation of the complete Product

Level 3 maintenance must ensure a maximum of autonomy for the user organizations. International cooperation can allow the set-up of an effective and economic authorized level of maintenance. It is recommended that single source repair be considered the best solution.

Level 3 maintenance will be performed in appropriately equipped facilities or component repair facilities of the customer or the contractor (or subcontractors). Level 3 requires appropriately trained and specialized personnel. Level 3 maintenance also involves the return of defective items (suspected or confirmed) to the OEM for repair/overhaul/retest.

Note

The Level 3 activities as the "highest" maintenance level can also be subdivided into two or more levels (eg Level 3 and Level 4). This can be used when required to clearly separate the user operated activities from the contractor operated activities at industry facilities. In this case, the maintenance strategy would contain four levels of maintenance (or possibly more).

Chapter 12

Maintenance task analysis

Table of contents

	Page
Maintenance task analysis.....	1
References.....	2
1 General	3
1.1 Introduction	3
1.2 Objective	3
1.3 Scope.....	3
1.4 Terms, abbreviations and acronyms	3
2 Task justification	4
3 Categorization of activities.....	5
4 Task documentation	5
4.1 General aspects.....	5
4.2 Practical considerations.....	6
5 Task structure	9
5.1 Rectifying and supporting tasks.....	9
5.2 Structure of supporting tasks.....	10
5.3 Structure of a rectifying task - referencing method.....	11
5.4 Task preconditions, pre-work and post-work.....	12
5.5 Narrative description.....	12
6 Task frequency	12
6.1 Maintenance activities due to inherent equipment failures	12
6.2 Task frequency for supporting tasks.....	14
6.3 Maintenance activities due to damages or special events	15
6.4 Scheduled maintenance activities	15
7 Task resources and task duration	16
7.1 Resources.....	16
7.2 Resources out of references	18
7.3 Harmonization of support equipment and spare parts	18
7.4 Task location aspects	18
7.4.1 Location in conjunction with the Product itself.....	18
7.4.2 Location in conjunction with the required facility/infrastructure	18
7.4.3 Location in conjunction with the zone within the Product itself	19
7.5 Product and system availability during maintenance performance	19
7.6 Support solutions (task variants)	19
7.7 Task duration	20
7.8 Parallel activities within maintenance tasks	20
8 Training requirements.....	22
9 Examples of assigning rectifying tasks to LSA candidates	22
9.1 Example 1: Major failure of equipment, non-repairable.....	24
9.2 Example 2: Mainboard failure of equipment 401 - Option 1	26
9.3 Example 3: Mainboard failure of equipment 401 - Option 2.....	28
9.4 Example 4: Failure of device 01, repairable at operational site	29
9.5 Example 5: Complex maintenance procedure on several levels	32

List of tables

1 References	2
--------------------	---

2	Terms, abbreviations and acronyms	3
3	Examples for possible categorization of tasks	5
4	Assembly view versus breakdown view	6
5	Component types.....	7
6	Typical situations of task documentation requirements.....	8
7	Typical rectifying tasks for typical events	9
8	Formula symbols (task frequency rectifying tasks)	13
9	Formula symbols (task frequency supporting tasks)	14
10	Formula symbols (task frequency scheduled maintenance tasks).....	15
11	Assignment of task resources	16
12	Summary of resources	17
13	MET and labor time	20
14	Comparison of resources for sequential or parallel subtasks	21
15	Breakdown of equipment example	23
16	Explanation of formats in the schematical LSA representations	25
17	Different spare part triggers.....	32

List of figures

1	Event-task correlation	4
2	General equipment example	6
3	Remove procedure including preliminary work	10
4	Remove procedure without preliminary work activities	10
5	Typical rectifying task - repair procedure	11
6	Task resources	17
7	Parallel activities and their resources/durations	21
8	Equipment example	22
9	Breakdown of Equipment 401	23
10	Major failure of equipment, non-repairable.....	24
11	Schematical LSA representation of example 1	26
12	Mainboard failure of equipment 401 - rectified by equipment 401 replacement	26
13	Schematical LSA representation of example 2	27
14	Mainboard failure of equipment 401 - rectified by equipment 401 repair	28
15	Schematical LSA representation of example 3	29
16	Failure of device 01, repairable at operational site.....	29
17	Schematical LSA representation of example 4	31
18	Major failure of equipment, only repairable at higher level.....	32
19	Schematical LSA representation of example 5	34

References

Table 1 References

Chap No./Document No.	Title
Chap 7	LSA failure modes and effects analysis
Chap 8	Damage and special event analysis

Chap 9	Logistic related operations analysis
Chap 10	Scheduled maintenance analysis
Chap 21	Terms, abbreviations and acronyms
S1000D	International specification for technical publications using a common source database

1 General Introduction

The requirement for a maintenance task must be analyzed from different perspectives. The first step is to divide up the entire task into appropriate work steps, or subtasks. This is necessary to apply a useful structure to a complex and time-consuming activity. It is not unusual for repair or replace procedures to contain hundreds of subtasks. The sequence of each of the subtasks must be clear. The decision to divide up an activity depends on the requirements for the depth of information. Depending on the use case, it can be sufficient to break down an activity into 10 subtasks with some general information. In other use cases, where there is a requirement to perform a deep analysis, the same task, which was broken down in just 10 subtasks before is now divided up into 40 subtasks. Additionally, an analysis of the tasks concerning the resources is required. Everything that is needed to perform the maintenance task must be identified within the Maintenance Task Analysis (MTA).

1.2 Objective

This chapter is a guideline for the analysis of an identified maintenance task with regard to its logistic requirements. This mainly includes spare parts and consumables, support equipment, personnel, facilities and task duration information. Additional information such as task criticality, task location, training needs, pre- and post-conditions or safety requirements must also be taken into consideration.

1.3 Scope

This chapter is directed at analysts who perform analysis activities within the LSA process to identify corrective and preventive maintenance tasks. After the identification, a deeper analysis of the maintenance activities is required. It is recommended that the establishment of close co-operation between disciplines such as maintainability, testability and reliability, together with the personnel to perform an MTA, is created. An ideal solution would be to combine these duties and assign them to one analyst.

1.4 Terms, abbreviations and acronyms

Definitions that are specific to MTA are given in [Table 2](#). The complete set of LSA terms, abbreviations and acronyms is given in [Chap 21](#).

Table 2 Terms, abbreviations and acronyms

Term	Definition
Supporting task	A supporting task is a part of a complete maintenance activity. As a standalone task it cannot rectify an event, such as failures, damages, special events or thresholds. A supporting task contains subtasks in terms of work steps.
Rectifying task	A rectifying task is any maintenance activity that resolves an event, such as failures, damages, special events or thresholds. A rectifying task contains subtasks in terms of referenced supporting tasks and/or definite work steps.

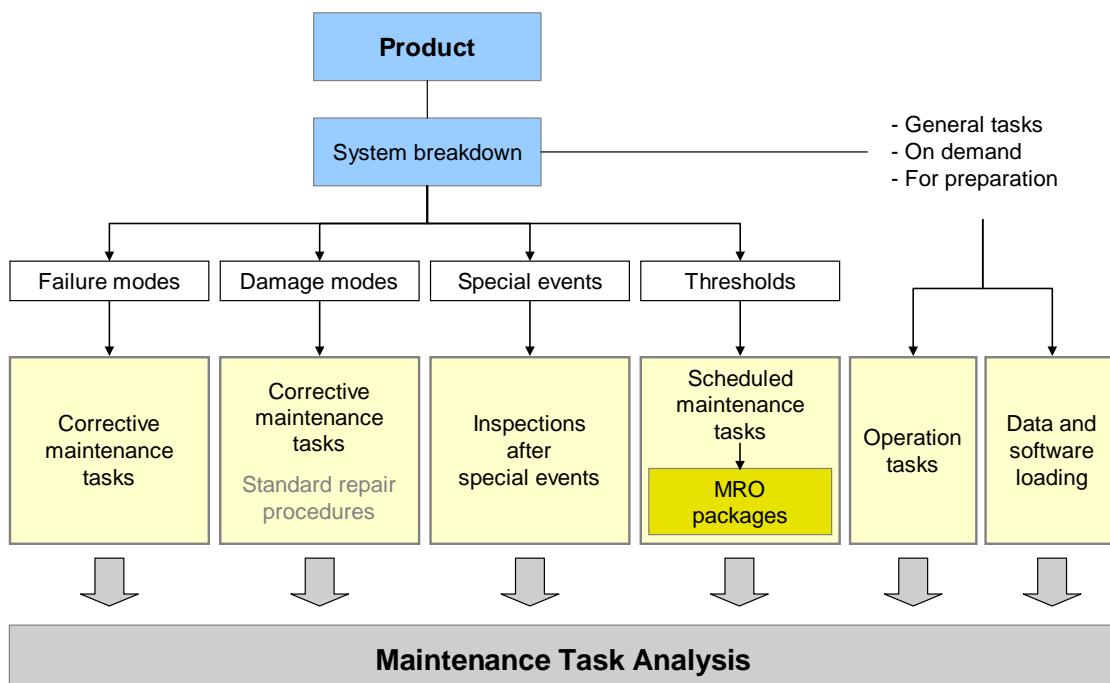
2 Task justification

Any maintenance task or operational activity that is identified as required will cause effort within support engineering and logistics disciplines. For that reason, a justification for the performance of each activity in the environment of maintenance and operational tasks is required. The principle of event driven maintenance is one of the main aspects.

Maintenance tasks and related operational tasks to be considered by an LSA are justified by corresponding logistics analysis activities. Failures, damages, special events and thresholds are the primary drivers for maintenance activities and are described in their specific chapters. Refer to [Chap 7](#), [Chap 8](#) and [Chap 10](#).

Additionally, there are tasks which originate from other sources (eg handling and servicing requirements or software and data loading). Refer to [Chap 9](#).

For proper documentation of the connection between the justifying source and the related maintenance activity, a corresponding structure in the LSA database must be established. [Fig.1](#) shows an overview of the relationship between maintenance/operational tasks and their justification.



ICN-B6865-S3000L0049-001-01

Fig 1 Event-task correlation

Deciding whether a maintenance task that is clearly justified by a maintenance relevant event or a general supporting task can be difficult, particularly when analyzing operation support or software loading tasks. For example, a task for towing an aircraft can have several different justifications:

- Towing out of the hangar to prepare for a mission
- Towing for refueling purposes
- Towing to the maintenance hangar in case of a failure

In this case, it is not possible to indicate clearly whether the towing task is part of a maintenance activity or an operational activity. However, this task must be considered as a specific supporting task, which is required for different needs. Documentation of these activities and who is responsible for them must be agreed upon between the customer and the contractor, within

the LSA GC, in order to avoid later misunderstanding or even the neglect of such activities. The integration of these activities within the Product breakdown is described in [Chap. 3](#).

3 Categorization of activities

For unique task identification, the use of a coding system such as the data module coding given in S1000D for technical publication is recommended (refer to S1000D). The coding of the activity is categorized in two data elements, information code and information code definition. Taking into account that technical documentation and the LSA are strongly interconnected, it is recommended that the S1000D task definitions also be used in the LSA database. This avoids the requirement for a translation matrix from the LSA task coding and nomenclature to the coding and naming within a technical documentation system. This approach improves commonality, traceability and, in the end, ILS quality.

In the description of a code for a special activity, there must always be an appropriate verb, which clearly identifies the activity. Avoid the usage of general terms. [Table 3](#) shows some examples from S1000D.

Table 3 Examples for possible categorization of tasks

Information code	Information code description
250	Clean
251	Clean with chemical agents
256	Polish and apply wax
341	Manual test
342	Automatic test
520	Remove procedure
720	Install procedure
752	Data loading
921	Remove and replace procedure

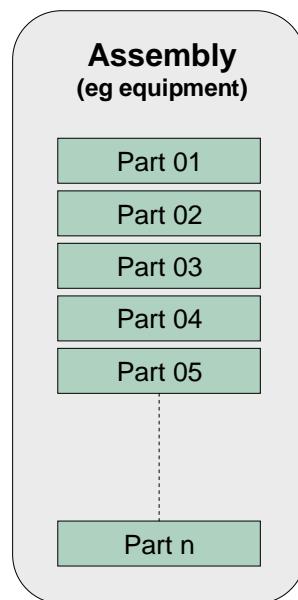
Note

With the help of the information code of S1000D, activity groups are defined. There are more general information codes, such as 250 for clean. Additionally, more specific procedures concerning cleaning activities can be applied with a more detailed information code such as 251 for cleaning with chemical agents. Refer to S1000D.

4 Task documentation

4.1 General aspects

The selection criterion for LSA candidate items is described in [Chap. 3](#). Maintenance concepts for breakdown elements or parts which are selected as LSA candidates can be documented in different ways. A simple general example is given in [Fig. 2](#) to begin clarification of the different methods that can theoretically be used.



ICN-B6865-S3000L0050-001-01

Fig 2 General equipment example

The assembly in [Fig 2](#) above contains n different parts. Theoretically, the assembly itself and each part within the assembly can fulfill criteria to be selected as a potential LSA candidate. From a maintenance point of view, there are different possibilities to document the identification of maintenance tasks concerning the repair of the assembly:

[Table 4](#) shows the extreme situation of an either-or decision. As the documentation of an LSA candidate must be considered as a cost intensive activity, it is recommended to be economical with addressing breakdown elements and/or parts to be selected as an LSA candidate.

Table 4 Assembly view versus breakdown view

Assembly view	Breakdown view
One LSA candidate selected = assembly itself, eg equipment	n LSA candidates selected (each part can become an LSA candidate, because it can be replaced)
n maintenance tasks to be documented: repair assembly by replacement of part 1 to part n	n maintenance tasks to be documented: replace part 1 to part n

4.2

Practical considerations

The situation in reality is more complex than that described in [Para 4.1](#). The parts from the simple example above can be repairable themselves or be discard items. Being repairable can make a specific part within a piece of equipment an LSA candidate also. In this situation, having an LSA candidate item within another LSA candidate item cannot always be avoided.

One of the decisive questions will be which component of the equipment has what kind of properties concerning the possibility of repair at which maintenance level. This has a large impact on the storekeeping method of the different items. To clarify terminology, it can be distinguished between several types of repairable, shown in [Table 5](#).

Table 5 Component types

Component type	Description
Non-repairable	A part that will not be repaired at any maintenance level. The reason can be technical and/or economical. The methods of replacement within the storekeeping must be clarified.
Repairable on industry maintenance level	An equipment/part which can be repaired only on industry level. For such items, only the replacement will be described in the LSA database. It is recommended to identify the possibility of repair on industry level in the LSA database. A detailed description of repair tasks is not required. The methods of replacement within the storekeeping must be clarified.
Repairable on customer maintenance level, depot site	A part/equipment that can be repaired on a specialized customer level. For such items, the replacement tasks and the repair procedures can be described in the LSA database. The level of technical documentation for the maintenance activities must be clearly defined within the LSA guidance conference. The methods of replacement within the storekeeping must be clarified.
Repairable at customer maintenance level, operational site	A part/equipment that can be repaired at customer level at an operational site. For such items, the replacement tasks and the repair procedures must be described in the LSA database. The level of technical documentation for the maintenance activities must be defined clearly within the LSA guidance conference. The methods of replacement within the storekeeping must be clarified.

This rough classification of equipment/parts additionally depends on the influencing events on the items. There can be failures that lead to the disposal of the complete equipment/part, and other failures will result in the equipment/part being repaired. Each event must be analyzed to identify which maintenance activity, at what maintenance level, must follow to rectify the event while being aware that a definitive decision is not always possible.

There will be events that can be rectified on several maintenance levels. In this case, not only the types of influencing events are significant, but also the economic aspects must be considered carefully. For example the result of a Level of Repair Analysis (LORA) can be the replacement of the complete equipment (with a follow-on discard) because of economic aspects, however the equipment would be repairable. Also, mixed scenarios are possible. For example, a certain percentage of a specific failure is rectified at the customer maintenance level, and the balance will be returned to the industry maintenance level because of capacity constraints at customer maintenance level.

In practice, many different situations concerning maintenance driving events (especially failures) and the following required chain of maintenance activities can occur. For better understanding, [Table 6](#) gives an overview of typical examples. Additionally, an example of a rather simple piece of equipment is given in [Para 9](#) (different maintenance situations and how they can be documented in the LSA).

Table 6 Typical situations of task documentation requirements

Event	Rectifying task	Possible follow-on task	Remark
Failure of a non-repairable component of equipment	Repair equipment by replacement of faulty component at operational site	Disposal of faulty component	Non-repairable component is required as a spare part If the documentation of the discard of the faulty component is required, a specific disposal procedure would be documented against the component (as the appropriate LSA candidate) as a separated rectifying task
Failure of a repairable component of an equipment	Repair equipment by replacement of faulty component at operational site	Repair faulty component at operational site	End item is not waiting until the completion of the repair of the component Component is required as a spare part
Failure of a repairable component of an equipment	Repair equipment by direct repair of faulty component at operational site	None	End item is waiting until the completion of the repair of the component Component is not required as a spare part, because the repaired component will be re-installed
Failure of repairable component of an equipment, component is repairable at customer depot site	Repair equipment by replacement of faulty component at operational site	Forwarding component to customer depot site Repair faulty component at customer depot site	Repairable component is required as a spare part at operational site If the documentation of the repair of the faulty component as a follow-on task is required, it would be documented against the component (as the appropriate LSA candidate) for the repair
Failure of repairable component of an equipment, component is repairable at industry level	Repair equipment by replacement of faulty component at operational site	Forwarding component to industry	Repairable component is required as a spare part at operational site Documentation of repair on industry level not required
Failure of equipment, only repairable at industry or customer depot site	Replace faulty equipment at operational site	Forwarding equipment to industry Repair equipment at industry or customer depot site	Equipment is required as a spare part at operational site If the documentation of the repair of the faulty equipment at customer depot site as a follow-on task is required, it would be documented against the equipment (as the appropriate LSA candidate) for the repair
Major failure of equipment, no repair possible	Replace equipment at operational site	Discard faulty equipment	Equipment is required as a spare part on operational site If the documentation of the discard of the faulty equipment is required, a specific disposal procedure would be documented against the equipment (as the appropriate LSA candidate) as a separated rectifying task

It is not possible to document all combinations of events and the follow-on repair/replace activities within this chapter. The examples given are typical ones. Depending on additional aspects (eg abilities at the different maintenance levels, scrap rates of equipment or components, end item waiting for repair or not), the sequence of activities can be influenced in multiple ways. The examples are given to clarify the principle of summarizing maintenance activities within a few full LSA candidates. A number of follow-on tasks on higher maintenance levels need not to be analyzed in detail. Only the identification of such possible tasks can be important. In most cases, a failure of equipment is rectified by two different task types:

- Replacement of the entire equipment (represented by a specific breakdown element)
- Repair of equipment by replacement of parts (as a follow-on task)

The follow-on tasks describe the repair procedures that must be analyzed in detail, in order to decide where these activities must be linked to and how deep of a detailed description is required. Another aspect to be considered is that not every replace task can be performed at a customer operational site. For this reason, it can be necessary to transport the complete Product/end item itself to a higher level maintenance site. In this case, the replacement of defective equipment will be performed at industry level or at a customer depot site.

5 Task structure

Establishing a common understanding of how to create a proper maintenance task structure , without any problems, can be difficult." Aspects such as, the categorization of maintenance activities, establishing a hierarchy of maintenance tasks and subtasks, and using information within existing task with the help of an intelligent referencing system, all need to be taken into account

5.1 Rectifying and supporting tasks

One important step that will give a structure to the tasks is to divide them into two classes of tasks.

- Rectifying tasks ("event solver")
- Supporting tasks

Each maintenance activity is driven by an event. This event can be a failure, damage, a special event or a deadline, which has expired. All of these events require a maintenance action to be completed. Each task that is able to resolve such an event can be defined as a rectifying task. Typical examples for rectifying tasks are given in [Table 7](#):

Table 7 Typical rectifying tasks for typical events

Event	Rectifying tasks
Failure	Repair or replace procedures
Damage	Repair or replace procedures
Special event	Inspection/fault location
Threshold	Scheduled maintenance action (eg scheduled replace of a time change item)

As opposed to the rectifying task, a supporting task cannot be the solving task for an event. The supporting task as a standalone task can never resolve a failure or damage and can never be the standalone activity after a special event. Also, operational needs can cause specific activities which belong to the group of supporting tasks. Typical supporting tasks are:

- Test procedures
- Gain access/undo access
- Remove and install procedures

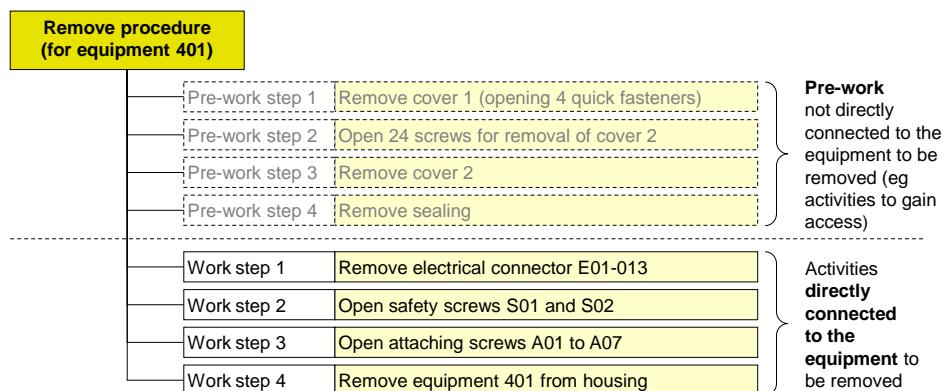
- Assemble and disassemble procedures
- Jacking up of a vehicle

Note

Supporting tasks can be complex and time-consuming, therefore, it must be possible to divide these tasks into work steps.

5.2 Structure of supporting tasks

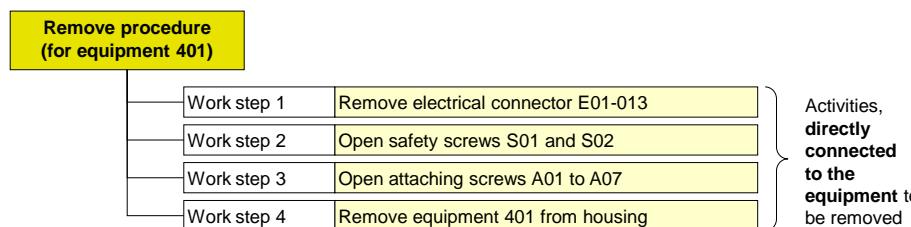
With the help of supporting tasks, the analyst is able to create a library of task modules, which can be used later to support the creation of the more extended rectifying tasks. The supporting task can be divided into work steps. [Fig 3](#) and [Fig 4](#) show a simple approach for the arrangement of a rather simple remove procedure:



ICN-B6865-S3000L0051-002-01

Fig 3 Remove procedure including preliminary work

As shown in [Fig 3](#), this example contains both the pre-work and the work steps indicating that the pre-work steps must be completed before the real work concerning the IUA, begins. This situation should be avoided when building supporting tasks. All of the preconditions to perform the work steps concerning the IUA itself are considered as completed when starting with the first work step. For this reason, the situation shown in [Fig 3](#) reduces to a task structure described in [Fig 4](#).



ICN-B6865-S3000L0052-001-01

Fig 4 Remove procedure without preliminary work activities

This second way of documenting the supporting tasks is the recommended one. The work steps are described with the help of subtasks within the supporting task. References to other existing supporting tasks must not be used in order to avoid nested references.

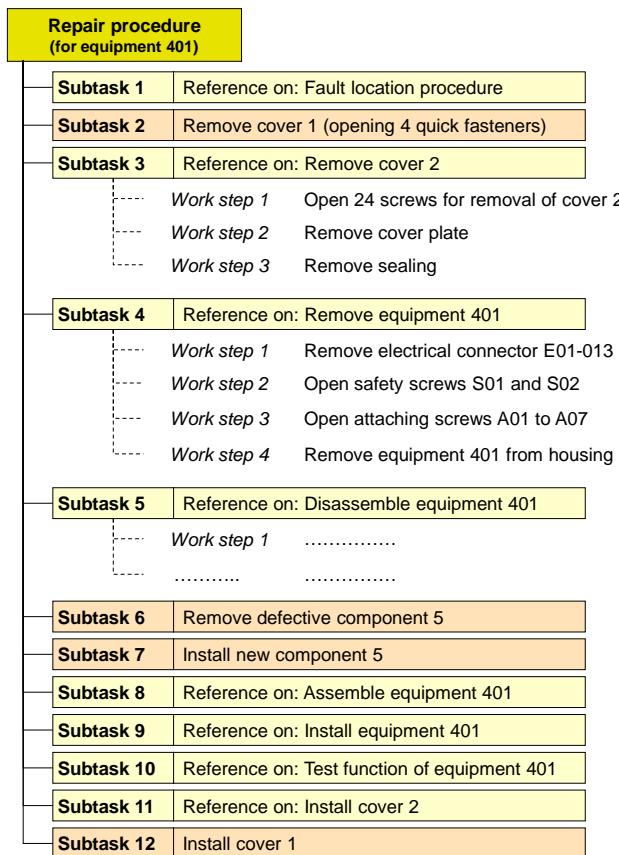
Note

The procedures shown in [Fig 3](#) and [Fig 4](#) can equate to a procedural data module. Refer to S1000D.

5.3

Structure of a rectifying task - referencing method

After the creation of a library of supporting tasks, the next step is to arrange the rectifying tasks, this means those procedures that can serve as "event solvers". [Fig 5](#) shows the structure of a typical repair procedure of equipment. Within this repair procedure, there are a number of subtasks. Some of them are subtasks that are already described on another breakdown indenture level (eg the removal of the cover 2). Some of them are subtasks that are already described on the same breakdown indenture level (eg the removal or the installation of the equipment 401). Both are possible, however, it is not necessary to describe the removal of the cover 2 again within the repair procedure. This is already done at the breakdown level of the cover 2. Therefore, it is recommended to use a reference to this existing task. Never use copies of existing task structures. It is possible that special supporting tasks can be referenced many times within a project. Therefore, for updating reasons, the original information must be documented only once within the LSA database. In case of modifications of the supporting task, the change needs only to be performed once. All references are updated automatically.



ICN-B6865-S3000L0053-002-01

Fig 5 Typical rectifying task - repair procedure

[Fig 5](#) shows a typical arrangement of a rectifying task. There are both subtasks and references to existing supporting tasks within the sequence of the entire repair procedure. In an extreme case, a rectifying task only contains references to existing supporting tasks.

Note

The use of references is recommended for all activities, which are always the same, independent of where and how often the IUA is installed on the Product (eg disassemble or assemble tasks are normally always the same, when the items are removed from the Product). Installation and removal tasks on the other hand can be very different, if an item is installed in several locations in the Product.

5.4

Task preconditions, pre-work and post-work

For each task, a number of preconditions must be fulfilled in order to carry out the activity. Such preconditions are of special interest for technical publications. Preconditions concerning safety must be documented within the preliminary requirements section of procedural data modules. The following aspects can be included:

- General preconditions:
This area describes the general preliminary work that must be carried out in order to achieve the conditions, so that the real task can be started.
- Safety preconditions:
This area covers all aspects necessary to achieve conditions in which the task can be carried out in a safe environment (warnings and cautions).
- Personnel preconditions:
In this area, all personnel aspects are covered. This includes the requirement of additional abilities or certifications of the person as well as the need for special personal requirements to achieve safety regulations.

The activities that must be carried out before reaching the required preconditions must appear in the sequence of the complete maintenance task (supporting or rectifying) as corresponding subtasks.

Another aspect is the identification of pre-work and post-work activities. In extended repair or replace procedures containing a large number of subtasks, there can be a number of subtasks which are necessary for the preparation of the subject repair or replace procedure (eg gain access or failure location procedures). Such subtasks for preparation can be marked in the LSA database as pre-work. The same applies to post-work activities, which do not belong to the real repair or replace procedure, but have to be performed in order to actually close the complete maintenance task (eg cleaning of support equipment, paperwork).

5.5

Narrative description

The narrative description of a maintenance activity is primarily a matter of technical documentation. For this reason, performing one and the same activity twice must be avoided. An extended description of a task within the LSA is not desirable. A short description of the work steps can be given, but a brief wording is recommended. The LSA must identify the required maintenance activities and analyze the identified tasks. The use of narrative texts in the LSA as a draft for technical documentation should be verified carefully. Technical documentation normally must obey strict rules concerning language, formatting and layout.

6

Task frequency

An important piece of information concerning maintenance activities is the frequency of the analyzed rectifying task. The frequency that a maintenance activity will be performed within a year is of central significance for the planning of logistic resources. The task frequency is directly linked to the frequency of the triggering event. The frequency of some events can be foreseen rather well, while others can be only estimated with the help of statistic methods.

6.1

Maintenance activities due to inherent equipment failures

Equipment failures statistically occur after a logistic time parameter called the Mean Time Between Failure (MTBF). This is a value for the average duration between two failures of equipment. With the assumption of a constant MTBF over the entire usage time and a single installation of equipment, a predicted task frequency of a rectifying task can be calculated with a simplified formula:

$$TF_{rec} = \sum_{i=1}^k FMR_i \cdot \frac{AOR}{MTBF} \cdot \lambda_{corr} \cdot \lambda_{MB}$$

Table 8 Formula symbols (task frequency rectifying tasks)

Formula symbol	Explanation
TF_{rec}	Task frequency of the rectifying task (1/year)
AOR	Annual operating requirements [operating hours/year] For equipment that works in continuous operation, AOR is 8760 hours per year. The measurement base of the AOR can also be something other than operating hours (eg distances or number of cycles).
$MTBF$	Mean Time Between Failure [operating hours]. The measurement base of the $MTBF$ can also be something other than operating hours (eg distances or number of cycles).
FMR	Failure mode ratio The failure of equipment can be distributed between several components. Each component can be responsible for the failure of the whole equipment with a certain ratio.
λ_{corr}	Correction factor for $MTBF$ (in case of special conditions at special installation areas). This correction factor can differ at each installation location of equipment. In the case of a multiple installation of equipment, this correction factor can be of special relevance.
λ_{MB}	Correlation factor to convert different measurement bases of AOR and $MTBF$ (eg an $MTBF$ is given not in operational hours, but in case of a vehicle in a specific distance measurement base like kilometers, miles).
i	Index for the identification of the FMR of the single failure mode within the analyzed equipment.
k	Number of different failure modes to be considered for the accumulation.

The formula above can be considered to be the simplest possibility. To consider the need for correction (eg due to environmental conditions which influence the reliability of the item under analysis), the correction factor λ_{corr} can be used in the formula above. To consider different measurement bases of AOR and $MTBF$, the correlation factor λ_{MB} can be used (eg to convert a covered distance of a vehicle to a time measurement bases like operating hours). If no correction or correlation is required, both values equal 1.

A more precise task frequency calculation is only possible with more complex formula taking into account other influence factors such as:

- No failure found factors (eg Built in Test (BIT) cannot duplicate)
- $MTBF$ can be a function of time $MTBF(t)$, then integral calculation has to be used
- $MTBF$ can be of different types (eg predicted, allocated, measured), the one to be used must be selected carefully
- Correction because of induced failures can be required

The logistic value $MTBF$ is of special interest when different types of equipment are analyzed. The distribution of the $MTBF$ value over the entire life cycle of equipment can have a different behavior depending on the type of equipment. $MTBF$ values can change over the life cycle of equipment. Mechanical equipment with deterioration effects has a behavior that is different than for example that of electronic equipment. Such examinations will be made by reliability analysis activities. With the help of mathematical functions, different failure distribution behavior can be described in detail. In this context refer to special literature concerning reliability.

In case of multiple installation of equipment it is recommended that the task frequencies for each installation point be calculated separately. For a required accumulation of task frequencies (eg for maintenance tasks being performed in a special repair shop) the calculated single task frequencies can be summarized. Depending on the installation area, the frequency of the failure of equipment can be dramatically different.

6.2

Task frequency for supporting tasks

For supporting tasks, a task frequency cannot be calculated in the same way as for rectifying tasks. Supporting tasks are not linked directly to any event like the rectifying tasks are. However, supporting tasks are normally "called" by rectifying tasks using the referencing method. By adding the task frequencies of the referencing rectifying tasks, a "sum up" task frequency of any supporting task can be calculated with the help of the following simplified formula:

$$TF_{supp} = \sum_{i=1}^n \lambda_Q \cdot TF_{rec,i}$$

Table 9 Formula symbols (task frequency supporting tasks)

Formula symbol	Explanation
TF_{supp}	Task frequency of the supporting task (1/year)
$TF_{rec,i}$	Task frequency of the referencing rectifying task (1/year)
λ_Q	Number of "calls" of the rectifying task for the supporting task to be calculated
i	Index for the referencing rectifying tasks
n	Number of the referencing rectifying task

A task frequency of supporting tasks must be used with care since the related task requirements are already covered by the rectifying tasks themselves which reference the supporting tasks. Care must be taken to avoid double counting any requirements.

Due to the fact that a supporting task can be referenced by different and many rectifying or operational tasks, the calculation of a task frequency for the supporting tasks can be of interest concerning statistical figures that can be used for:

- Identification of maintenance drivers to be assessed for potential improvement of design
- Quantification of maintenance activities depending on the real usage/maintenance of an item

Example: Opening and closing of covers

The remove and install tasks of a cover can be referenced many times by different rectifying tasks from many different LSA candidates. It can be important to know how often a cover is really removed and installed within one year because of the need for some spare parts after a certain number of removal and installations. The remove and install of the cover due to inherent failure or because of damage is normally only a small fraction of the real removal and installation activity. The dominating fraction results from the need to gain access to other equipment behind the cover.

Because of a frequent removal and installation of a cover, the design must be user friendly (eg by the usage of quick release fasteners) and damage tolerant.

6.3

Maintenance activities due to damages or special events

Failures of equipment for other inherent reasons cannot be predicted in the same way as described in [Para 6.1](#) since the task frequency can only be estimated. If there is any experience concerning the occurrence of damages or special events, with the help of statistical investigations for example, the results can be used to get an idea of how frequent these unpredictable events occur and how frequent the corresponding maintenance activities will arise.

However difficult it is to forecast the frequency of these events, it is important to get an estimation based on the best information available. All maintenance activities, which are triggered by these events, have their corresponding requirements concerning material and personnel. To estimate the resources and the capacity utilization, these unpredictable events must be taken into account.

6.4

Scheduled maintenance activities

In the case of scheduled maintenance, no additional calculation or estimation is required because the interval of the maintenance activity is fixed by the corresponding Scheduled Maintenance Analysis (SMA) and by maintainability activities by assembling proper scheduled maintenance packages.

Intervals given by scheduled maintenance can be converted to task frequency values based on how often they occur per year. Each interval that is given by a value with a specific measurement base must be related to the annual operating requirements. The task frequency can be calculated with the following simple formula:

$$TF_{\text{sched}} = \frac{AOR}{Int} \cdot \lambda_{\text{corr}} \cdot \lambda_{\text{MB}}$$

Table 10 Formula symbols (task frequency scheduled maintenance tasks)

Formula symbol	Explanation
TF_{sched}	Task frequency of a scheduled task (1/year)
AOR	Annual operating requirements [operating hours/year] For equipment that works in continuous operation, AOR is 8760 hours per year. The measurement base of the AOR can also be something other than operating hours (eg distances or number of cycles).
Int	Interval with corresponding measurement base (same as AOR)
λ_{corr}	Correction factor for $MTBF$ (in case of special conditions at special installation areas). This correction factor can differ at each installation location of equipment. In the case of a multiple installation of equipment, this correction factor can be of special relevance.
λ_{MB}	Correlation factor to convert different measurement bases of AOR and $MTBF$ (eg an $MTBF$ is given not in operational hours, but in case of a vehicle in a specific distance measurement base like kilometers, miles).

Note

Scheduled maintenance can have a more complex structure concerning the determination of different thresholds, intervals and triggers for a scheduled maintenance task (refer to [Chap 10](#)). In this case, calculation of a simple task frequency value can often be impossible.

7

7.1

Task resources and task duration

Resources

The resources necessary to perform a maintenance task must be defined at an adequate level within the task itself. Generally, it must be possible to identify when a resource is needed within the sequence of the task. Additionally, it can be of interest for training purposes, which personnel needs appropriate skills to use special support equipment. Based on the principle of the structure of tasks, which can be arranged by subtasks and references to supporting tasks, resources are allocated accordingly. It is recommended to follow the basic principle that each resource is linked to that activity, when the resource actually is needed. [Table 11](#) gives an overview of possible resources and the method of linking it to the corresponding activity.

Table 11 Assignment of task resources

Resource	Description
Spare parts	Spare parts are assigned to the subtask, where they are actually required. To get an overview of the required spare parts for the entire task, all spares from the subtasks and from the referenced supporting tasks must be summarized.
Consumables	Consumables are assigned to the subtask, where they are actually required. To get an overview of the required consumables for the entire task, all consumables from the subtasks and from the referenced supporting tasks must be summarized.
Support equipment	Support equipment is assigned to the subtask, where it is actually required. If support equipment is required during the complete task, it can be assigned to the task instead of each single subtask. To get an overview of the required support equipment for the entire task, all support equipment from the task/subtasks and from the referenced supporting tasks must be summarized. Information can also be added about which personnel use the support equipment. In complex subtasks, more than one person can be actively using several pieces of support equipment. For training requirements, it must be possible to associate support equipment with the relevant personnel resource. The assignment of support equipment often identifies the need only. In this case, the identification initiates a process to procure or to develop corresponding support equipment based on an adequate specification. Therefore, the following cases are possible for the analyst in the MTA concerning support equipment:
	<ul style="list-style-type: none"> – Selection of existing and proven support equipment – Selection of a specification for support equipment – Identification of the need to develop a new specification of completely new support equipment
Personnel	Personnel is assigned to the subtask, where it is actually required. If personnel is required during the complete task, it can be assigned to the task instead of each single subtask. To get an overview of the required personnel for the entire task, all personnel from the task/subtasks and from the referenced supporting tasks must be summarized.

Resource	Description
Facilities	Facilities are assigned to the subtask, where they are actually relevant. If a facility is required during the complete task, it can be assigned to the task instead of each single subtask. To get an overview of the required facilities for the entire task, all facilities from the task/subtasks and from the referenced supporting tasks must be summarized.
Technical documentation	Required technical documentation (eg original manufacturer's documentation) is assigned to the subtask, where it is actually relevant. If technical documentation is required during the complete task, it can be assigned to the task instead of each single subtask. To get an overview of the required technical documentation for the entire task, all technical documentation from the task/subtasks and from the referenced supporting tasks must be summarized. Refer to S1000D.

Note

Although the S3000L data model enables the material resources to be linked to both task and subtask, it is recommended that they are assigned to the subtask, where it is actually required. In some cases it can be reasonable to link a resource on task level, eg if the resource is in use during the entire task and the task contains a large number of subtasks.

On the level of the entire task, all resources from the subtasks and referenced supporting tasks must be summarized and harmonized. This summary can be presented by reports concerning the different resources (or even for all resources in one report). An example is given in [Fig 6](#).

Install procedure for equipment 401							
Subtask	Spares	Consumables	Support equipment	Personnel	Special facilities	MET	Labor time
Install equipment 401 into housing	Seal [1x]	Adhesive C [as required]	None	Electrician 1 Electrician 2	None	1 min	1 min 1 min
Close attaching screws A01 to A07	Washer [7x]	None	None	Electrician 1	None	3 min	3 min
Close safety screws S01 and S02	None	None	Torque wrench [1x]	Electrician 1	None	2 min	2 min
Install electrical connector E01-013	None	None	None	Electrician 1	None	0,5 min	0,5 min

ICN-B6865-S3000L0054-002-01

Fig 6 Task resources

The summary for the install procedure from [Fig 6](#) on task level is shown in [Table 12](#):

Table 12 Summary of resources

Resource type	Resource	Quantity
Spare parts	Seal Washer	1 7
Consumables	Adhesive C	As required
Support equipment	Torque wrench	1

Applicable to: All

S3000L-A-12-00-0000-00A-040A-A
Chap 12

Resource type	Resource	Quantity
Personnel	Electrician	2
Special facilities	none	
Technical documentation	not required	
Mean Elapsed Time (MET)	6,5 minutes	
Labor time	7,5 minutes	

7.2

Resources out of references

Within the methodology of referencing supporting tasks, it must be considered that the resources of the referenced tasks will be taken over to the referencing rectifying task. At first glance, this seems to be a proper approach. However, this approach can result in a false estimation of personnel requirements. Normally, simple work steps are performed by personnel with a lower qualification (eg the removal of access covers). For that reason, these personnel are normally documented within these simple supporting tasks. If the supporting task is referenced within an extended repair or replace procedure, there must be the option in this context to change the real executing personnel to another person (with another/higher qualification), who is also able to perform the referenced activity and who is involved in the complete task anyway.

Similar to personnel, the use of support equipment originating in referenced supporting tasks should be analyzed carefully. Perhaps it is possible to replace the support equipment from the original supporting task by another, which is also used within the complete rectifying task at another position and which can also fulfill the needs of the original support equipment. In this case, it is possible to reduce the amount of different support equipment that must be used within the rectifying task.

7.3

Harmonization of support equipment and spare parts

To reduce administrative effort and to avoid needless costs, it must be ensured that all identified support equipment and spare parts will be harmonized on Product level. The list of support equipment must be analyzed whether there are options to replace several pieces of equipment by a single equipment or if there are support equipment of different manufacturers with exactly the same functionality. The mixture of spare parts from different manufacturers should be avoided where identical items of supply are installed in different locations.

7.4

Task location aspects

To cover all aspects concerning the location at which a task is performed three different location information types can be distinguished.

7.4.1

Location in conjunction with the Product itself

The location of a task in conjunction with the Product can be aligned with the S1000D data element Item Location Code (ILC). The terminology "On" and "Off" tasks can be referred to frequently. This means the location of the maintenance activity with regard to the Product. An "On"-task concerning location means that the activity is performed directly on the Product (eg in or directly at an aircraft). A removal task of an LRU is always an "On"-task, because the LRU is always removed directly from the Product. The disassemble task after the removal in a special maintenance shop is a typical example for an "Off"-task concerning this type of location. Refer to S1000D.

7.4.2

Location in conjunction with the required facility/infrastructure

Another piece of information concerning location of a task is the attribute of a required facility. Additional to the "On" and "Off" information from [Para 7.4.1](#) the required facility is an attribute at

the subtask level. The selection of a facility object as the location of a subtask does not fix automatically the location information concerning "On-" and "Off-" activities. Examples for infrastructure or facilities can be the following:

- Maintenance hangar
- Individual shops such as electronic shop or engine shop
- Movement area
- Out of area

7.4.3

Location in conjunction with the zone within the Product itself

If the Product is divided into physical areas (zones) this information can be added as an additional attribute (eg a zone identifier) to the subtask. In this case, the location aspect is related to the layout of the Product. The information can be used to estimate the amount of activity within each local zone of the Product.

Note

Information concerning zones can also be used for the documentation of the result of a zone analysis in conjunction with an SMA. In this case, the zones will be documented in the form of non-hardware breakdown elements. The identified scheduled maintenance tasks can be documented against these breakdown elements (refer to [Chap 10](#)).

7.5

Product and system availability during maintenance performance

Further important information concerning the Product or different systems within the Product is the availability during maintenance activities. This information can be documented with the help of corresponding codes on different breakdown levels.

This information covers the ability to operate the Product during maintenance procedures. It can be distinguished between:

- Product not available during maintenance
In this case, the Product cannot be used during, for example, a repair procedure or an inspection. The Product must wait until the maintenance action is finished before it is available again. A typical situation can be the removal of a defective component and the repair of the defective component (eg in a special maintenance shop). After successful repair, the repaired component will be re-installed into the Product. During the entire repair activity on the defective component, the Product was waiting.
- Product available during maintenance with reduced capability
In this case, the Product can be used during, for example, a repair procedure or an inspection with reduced capability.
- Product fully available during maintenance
In this case, the Product can be used during a repair procedure or an inspection with full capability. A typical example is the replacement of a defective component by a new one. Upon completion of the replace procedure, full capability is recovered. However, the defective component can be repaired in a special maintenance shop, but this repair does not have any influence on the availability of the Product. After the successful repair, the repaired component will be stored for later use.

7.6

Support solutions (task variants)

Maintenance activities can differ depending upon the environmental conditions during the performance of the task. In this case, the task itself can be nearly the same, however additional support equipment can be required or the task itself is different from the original one. Reasons for variants of tasks can be:

- Performance of the task under different environmental conditions (climatic aspects)
- Performance of the task at different places
- Performance of the task for another customer with different preconditions concerning personnel and equipment of the operational site

- Permanent repair or temporary repair
- Differences from peace time to war time scenarios

Task variation brought about by different conditions and/or configurations of the Product, can be reflected by using applicability, which can also carry over to the technical documentation. Refer to S1000D.

7.7

Task duration

To get an overview about the capacity utilization of personnel and support equipment, time information about the tasks is required. Within the task, the duration of a task and the working time of the personnel must be distinguished. [Table 13](#) explains the difference between Mean Elapsed Time (MET) and labor time.

Table 13 MET and labor time

Time	Description
Mean Elapsed Time (MET)	Duration of the entire task. The duration is addressed against the subtasks. The duration of the whole task can be calculated as the sum of times coming from the subtasks and from the referenced supporting tasks.
Labor time	<p>Summarized duration of personnel work. This time is addressed against the subtasks. If more than one person is working at the same time on a subtask, the labor time for each different skill level must be summarized.</p> <p>For example three persons of the same skill level are working at the same time within a subtask, which takes 20 minutes. In this case for this subtask the following times can be documented:</p> <p>MET: 20 minutes Labor time: 60 minutes</p>

Regarding times/durations of maintenance tasks, it must be considered, what type of duration is documented in the MTA. Taking into account the non-productive times to estimate personnel requirements is an especially important aspect. In the LSA database, normally so-called "spanner-in-hand" times are documented. This means that all preliminary work concerning preparation of the real maintenance activity (eg paperwork, fetching of support equipment or material) normally is not included. To take into consideration non-productive times, estimates can be performed outside the LSA.

Another aspect is the consideration of logistic delay times. Any delay caused by waiting times for logistic resources (eg support equipment or facilities not immediately available, waiting for personnel, waiting for materiel) is usually not documented in the LSA. To take into consideration logistic delay times, estimates can be performed outside the LSA.

Note

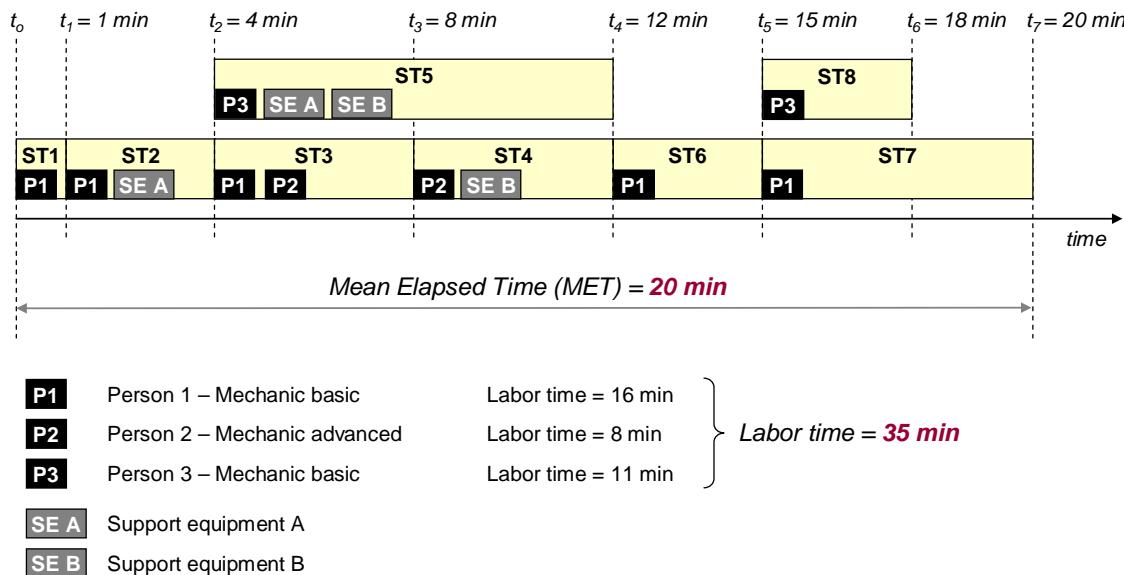
Waiting times caused by integral parts of a maintenance task such as the curing of materiel or the drying of paint should not be considered in the same way as non-productive times or logistic delay times. Another aspect is the need for authorized inspection before continuing the entire maintenance task with the next step. These well predictable procedures can be documented within a maintenance task as an own subtask with its specific duration and logistic requirements.

7.8

Parallel activities within maintenance tasks

For more complex maintenance tasks containing many subtasks which are performed by different persons, the simultaneous performance of work steps must be considered. This

simultaneity of activities of course has an influence on MET for the complete maintenance task, as well as for the task resources concerning support equipment and personnel. Refer to [Fig 7](#):



ICN-B6865-S3000L0055-001-01

Fig 7 Parallel activities and their resources/durations

The parallel subtasks **ST5** to **ST3/ST4** require the support equipment **SE A** and **SE B**. For subtask **ST5** the same support equipment as in subtask **ST2** can be used because these activities are not performed in parallel. The consequence for the complete task is the requirement of one support equipment **SE A**. For support equipment **SE B**, there is a different situation. Because of the parallel use of support equipment **SE B** in the partly parallel subtasks **ST4** and **ST5**, there is a requirement of two support equipment **SE B** for the complete maintenance task.

A similar situation holds for the personnel. The person with the qualification mechanic basic is required in parallel for the subtasks **ST3** and **ST5** as well as for **ST7** and **ST8**. This causes the requirement of an additional person **P3** with the same qualification as person **P1**.

[Table 14](#) shows the differences between a simple approach using a sequential methodology (one subtask after the other) or the possibility of parallel activities.

Table 14 Comparison of resources for sequential or parallel subtasks

Step by step sequence	Parallel activities
Mean Elapsed Time: MET = 35 min	MET = 20 min
Support equipment requirements: SE A: 1 SE B: 1	SE A: 1 SE B: 2
Personnel requirements: Mechanic basic: 1 (P1) Mechanic advanced: 1 (P2)	Mechanic basic: 2 (P1 and P3) Mechanic advanced: 1 (P2)

To document the parallel and sequential performance of subtasks, the elements *subtaskTimelineEvent* and *subtaskTimelineLag* contained in the [Chap 19](#) data model can be used to document the relationship of subtasks. This can also be used to create a draft version of a working plan by, for example, deriving a graphic schedule of a maintenance task from the LSA database content.

8 Training requirements

The identification of training requirements concerning maintenance activities can be derived from the maintenance tasks documented in the LSA database. In this context it must be distinguished between special training needs and training needs which are covered by the abilities which can be attained by normal professional education. However, especially in many international projects, the aspect of national situations concerning education can make it difficult to identify general training needs for each partner nation.

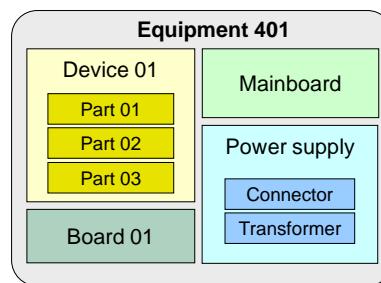
To support the Training Needs Analysis (TNA), there is a need to document special training requirements to perform a maintenance task. For this reason, the task must be analyzed concerning requirements on the subtask level (eg the use of a complex support equipment which requires training) as well as on the task level to judge the complexity of the task as a whole. The aspects to be considered are:

- Responsible section or department for the maintenance task
- Selected skill level of personnel performing the subtask
- Need for team training due to the complexity of the task
- Use of special support equipment which requires training
- Required special training to perform the task/subtask
- Required training methods
- Required experience to perform the task/subtask
- Complexity of the task as a whole

The information concerning training can be collected at task or subtask level. At the subtask level, it is possible to analyze each activity in detail and to assign special support equipment to the affected person performing the activity. Training information collected at task level are valid for all subtasks, eg determination of a general skill level required for the task. Summarizing the results of the training information for the complete task is the last step.

9 Examples of assigning rectifying tasks to LSA candidates

The following examples are provided to give a better understanding how to assign tasks to the corresponding LSA candidates. For that reason, a simple example of equipment is used for the following typical cases.



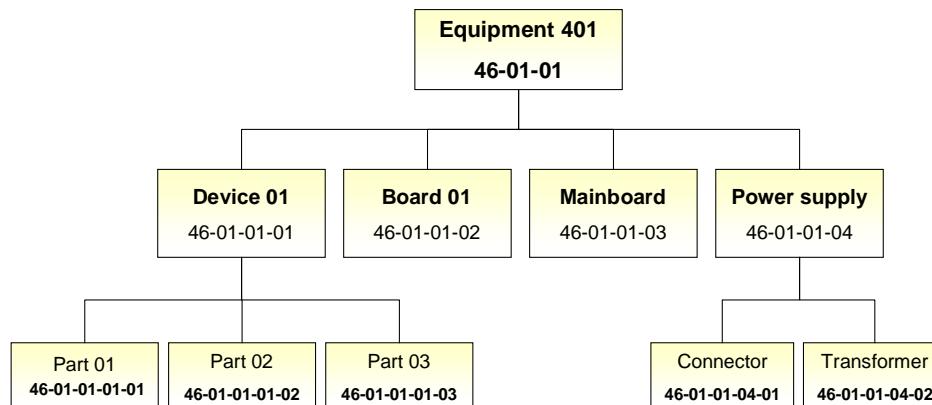
ICN-B6865-S3000L0056-002-01

Fig 8 Equipment example

The equipment example shown in [Fig 8](#) and contains the components with corresponding properties shown in [Table 15](#). The hierarchical Product breakdown is shown in [Fig 9](#).

Table 15 Breakdown of equipment example

BEI	Name	Component type	Next higher BEI
46-01-01	Equipment 401	Equipment (LSA candidate)	-
46-01-01-01	Device 01	Repairable at customer operational site	46-01-01
46-01-01-01-01	Part 01	Repairable at industry	46-01-01-01
46-01-01-01-02	Part 02	Non-repairable	46-01-01-01
46-01-01-01-03	Part 03	Non-repairable	46-01-01-01
46-01-01-02	Board 01	Repairable at industry	46-01-01
46-01-01-03	Mainboard	Non-repairable	46-01-01
46-01-01-04	Power supply	Repairable at customer depot site	46-01-01
46-01-01-04-01	Connector	Non-repairable	46-01-01-04
46-01-01-04-02	Transformer	Repairable at industry	46-01-01-04



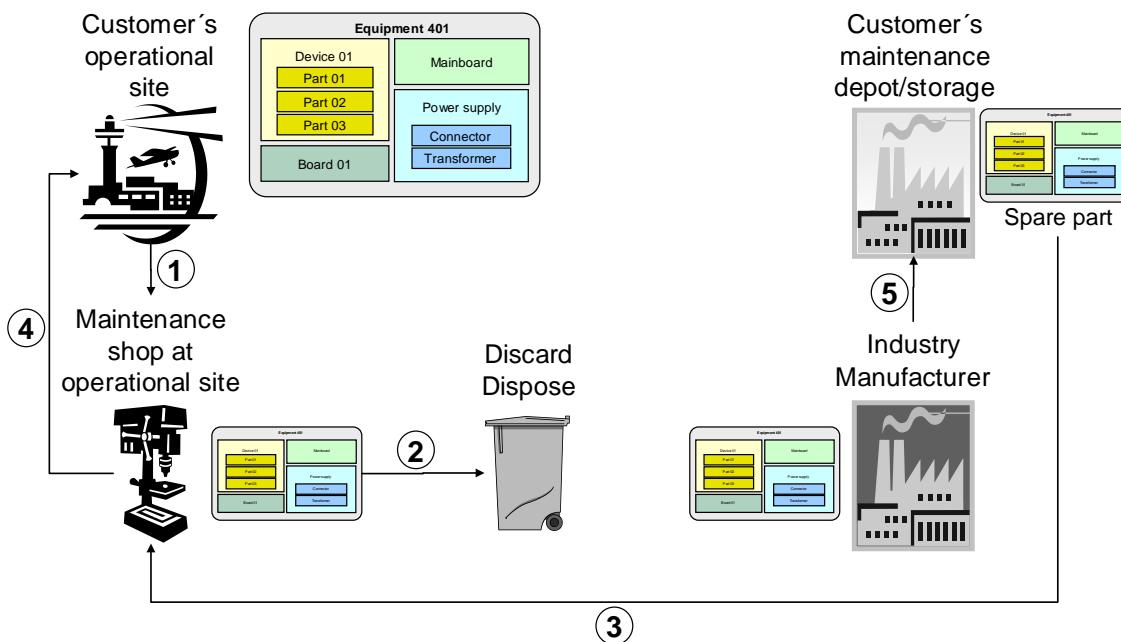
ICN-B6865-S3000L0102-001-01

Fig 9 Breakdown of Equipment 401

The following paragraphs establish a better understanding how maintenance activities can be presented within the LSA database. From a rather simple situation like an equipment replacement to an extended one with several activities at several maintenance levels the examples illustrate the diversity of maintenance tasks and to the need of careful documentation of all information which is crucial to complete a maintenance activity. For each event driven situation there are several potential solutions to rectify the failure.

In the examples, typical possible solutions are shown. In the figures, which show the representation of the LSA database content, the information is reduced to the essential steps in terms of LSA.

9.1 Example 1: Major failure of equipment, non-repairable



ICN-B6865-S3000L0057-002-01

Fig 10 Major failure of equipment, non-repairable

Event:

Failure of equipment 401 which cannot be repaired by replacement of components, a complete equipment change is required

LSA candidate: Equipment 401

Complete maintenance procedure description:

Refer to [Fig 10](#).

- 1 Remove faulty equipment 401 at customer's operational site
- 2 Test faulty equipment 401 at maintenance shop (need for a new equipment 401 to replace the faulty equipment 401 identified).
Discarddispose faulty equipment 401 (follow-on task)
- 3 Receive spare part (complete equipment 401) from customer's maintenance depot
- 4 Install new equipment 401 at customer's operational site (replacement of equipment 401)
⇒ Failure rectified, Product can be used again
- 5 Re-order new equipment 401 from industry to restock customer's maintenance depot

For schematical LSA representation, refer to [Fig 11](#).

Rectifying task for the event:

Replace equipment 401

Follow-on maintenance related tasks:

Discarddispose faulty equipment

Other follow-on activities:

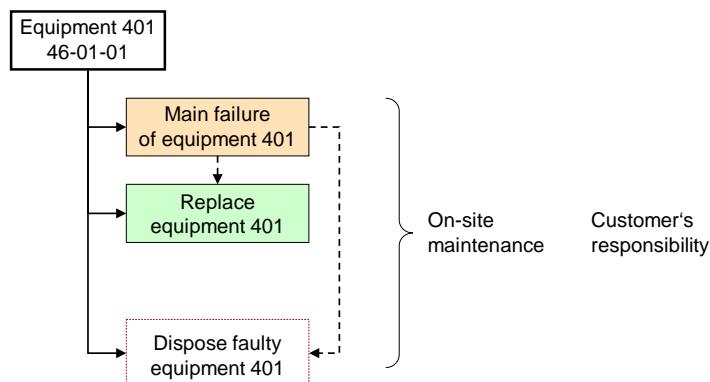
Re-order new equipment 401

The following table explains the usage of formats in the schematical LSA representations.

Table 16 Explanation of formats in the schematical LSA representations

Format	Explanation
White box, black letters	Breakdown element
Orange box, black letters	Failure at LSA candidate level
Green box, black letters	Rectifying task for the failure at LSA candidate level
White box, black letters, red dashed frame	Follow-on task triggered by a failure at LSA candidate level
Arrows, black	Relations from: Breakdown element to a subsequent breakdown element Breakdown element to the related failure at LSA candidate level Breakdown element to the rectifying task
Dashed arrows, black	Relations from: Failure to rectifying task Failure to follow-on task
Arrows, red	Relation from rectifying task to supporting task (only for example 4)

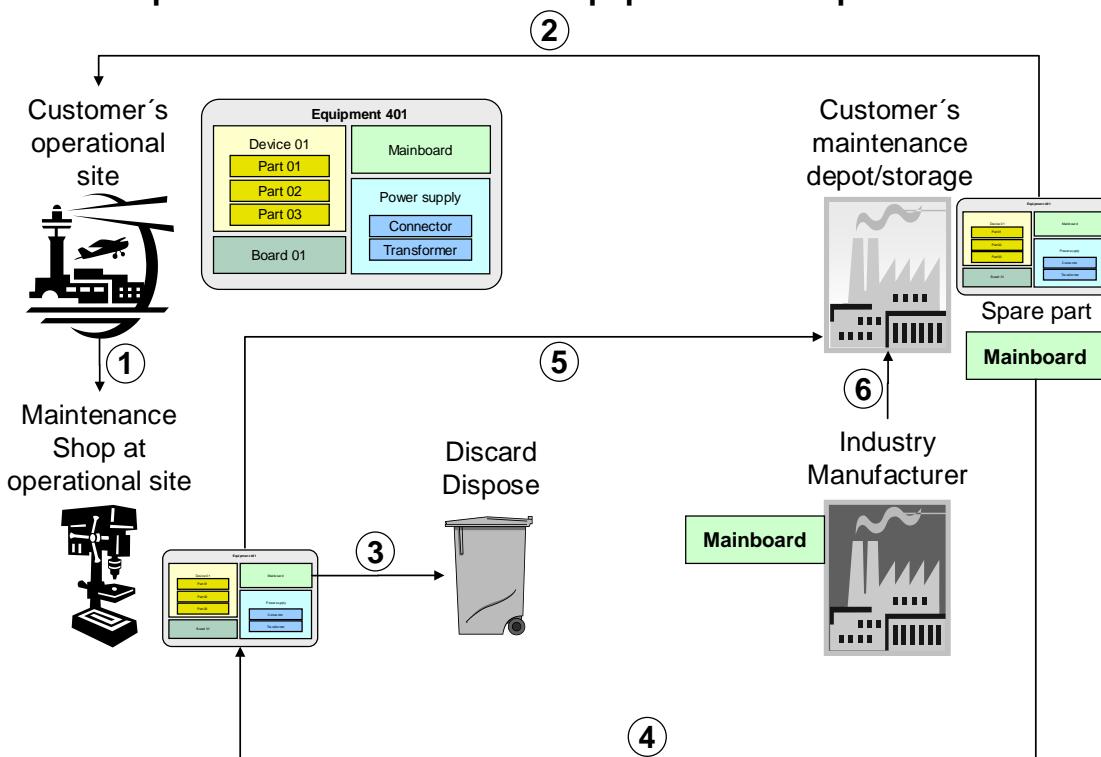
Schematic LSA representation of rectifying and follow-on tasks:



ICN-B6865-S3000L0103-002-01

Fig 11 Schematic LSA representation of example 1

9.2 Example 2: Mainboard failure of equipment 401 - Option 1



ICN-B6865-S3000L0058-002-01

Fig 12 Mainboard failure of equipment 401 - rectified by equipment 401 replacement

Event:

Failure of mainboard (component itself is not repairable)

LSA candidate: Equipment 401

Complete maintenance procedure description:

Refer to [Fig 12](#).

- 1 Remove faulty equipment 401 at customer's operational site
- 2 Receive spare part (complete equipment 401) from customer's maintenance depot
Install new equipment 401 at customer's operational site (replacement of equipment 401)
⇒ Failure rectified, Product can be used again
- 3 Remove faulty mainboard from equipment 401
Discard/dispose faulty mainboard
- 4 Receive spare part (new mainboard) from customer's maintenance depot
Repair faulty equipment 401 by installation of new mainboard
- 5 Send back repaired equipment 401 to customer's maintenance depot
- 6 Re-order new mainboard from industry to restock customer's maintenance depot

For schematical LSA representation, refer to [Fig 13](#).

Rectifying task for the event:

Replace equipment 401

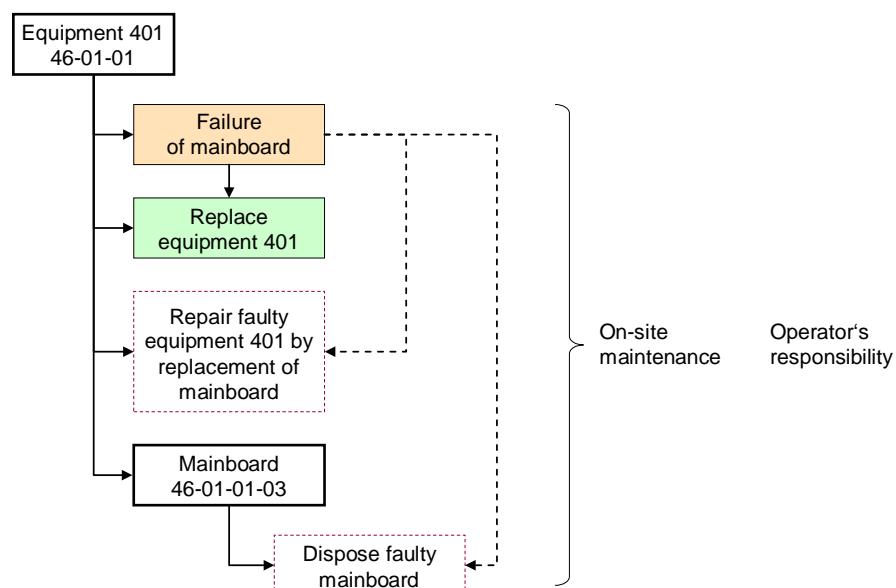
Follow-on maintenance related tasks:

Dispose faulty mainboard
Repair faulty equipment 401 by installation of new mainboard

Other follow-on activities:

Re-order new equipment 401

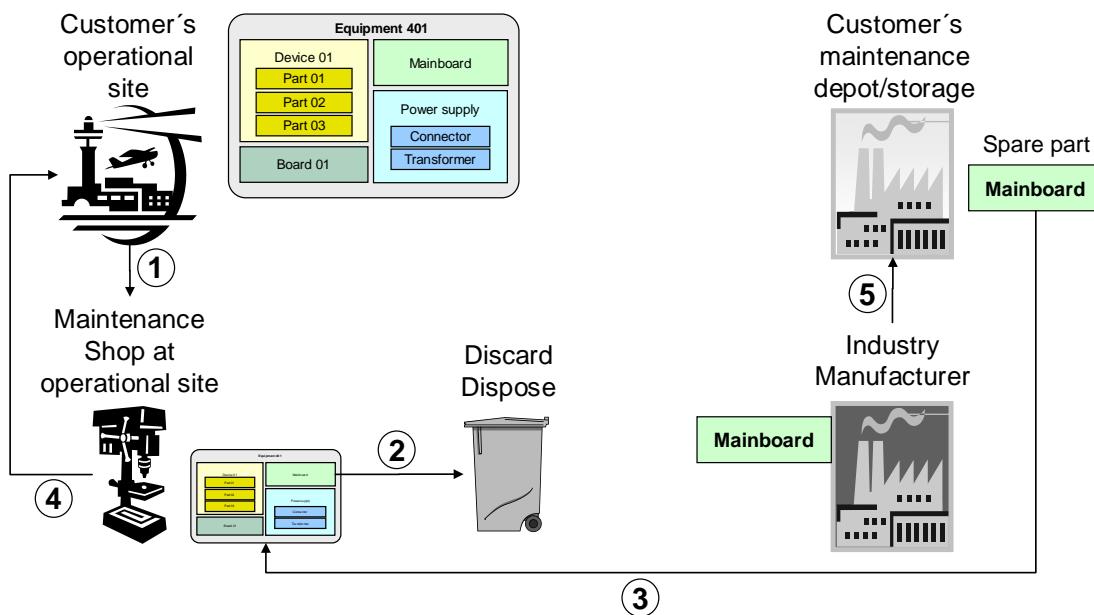
Schematical LSA representation of rectifying and follow-on tasks:



ICN-B6865-S3000L0104-001-01

Fig 13 Schematical LSA representation of example 2

9.3 Example 3: Mainboard failure of equipment 401 - Option 2



ICN-B6865-S3000L0059-002-01

Fig 14 Mainboard failure of equipment 401 - rectified by equipment 401 repair

Event:

Failure of mainboard (component itself is not repairable)

LSA candidate: Equipment 401

Complete maintenance procedure description:

Refer to [Fig 14](#).

- 1 Remove faulty equipment 401 at customer's operational site
- 2 Remove faulty mainboard from equipment 401
Discard/dispose faulty mainboard
- 3 Receive spare part (new mainboard) from customer's maintenance depot
Repair faulty equipment 401 by installation of new mainboard
- 4 Re-install equipment 401
⇒ Failure rectified, Product can be used again
- 5 Re-order new mainboard from industry to restock customer's maintenance depot

For schematical LSA representation, refer to [Fig 15](#).

Rectifying task for the event:

Repair equipment 401 by replacement of mainboard

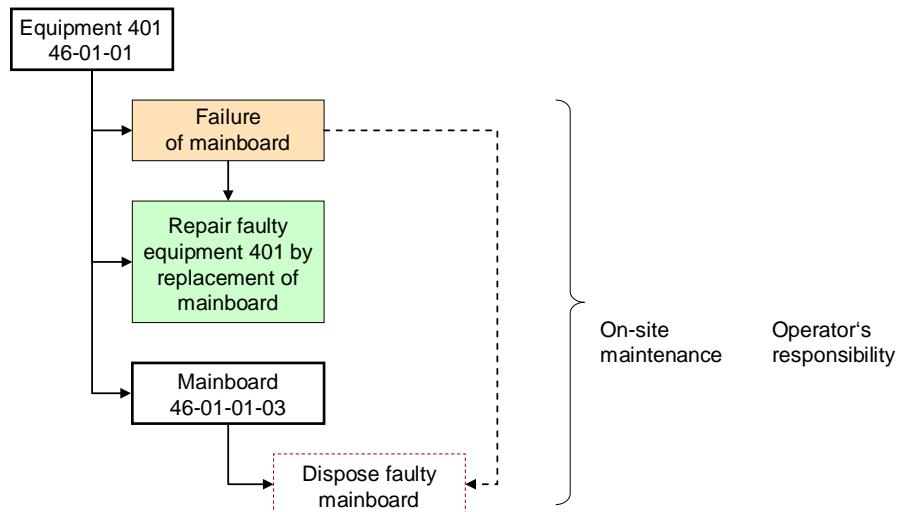
Follow-on maintenance related tasks:

Discard/dispose faulty mainboard

Other follow-on activities:

Re-order new mainboard

Schematic LSA representation of rectifying and follow-on tasks:

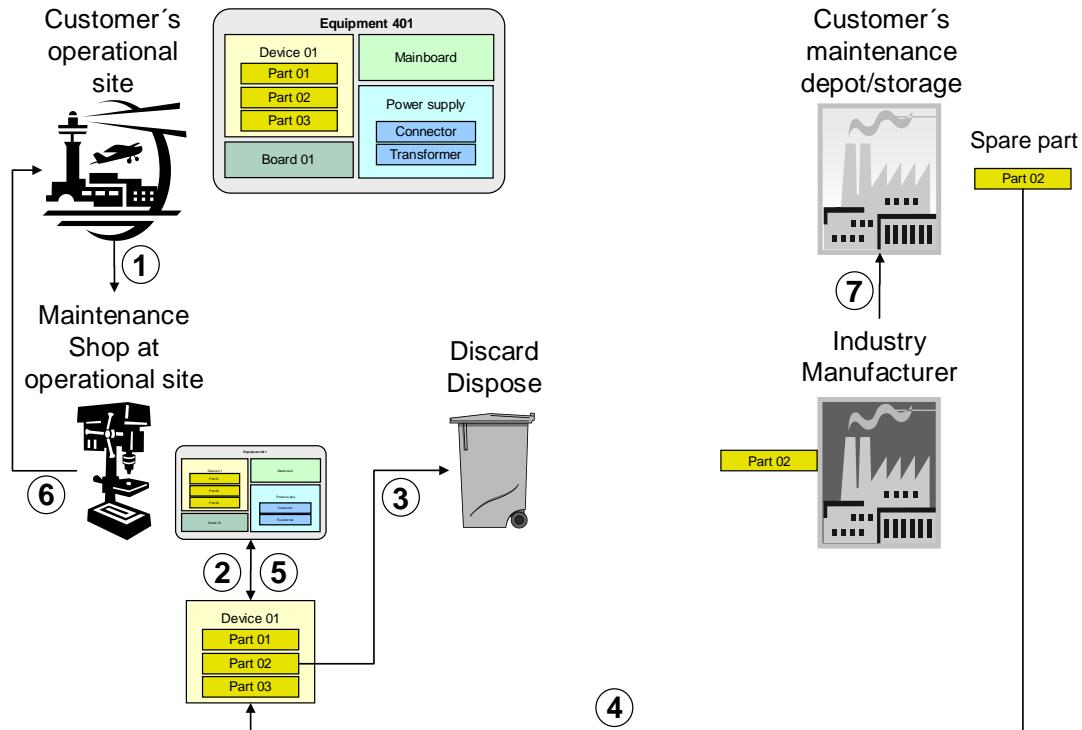


ICN-B6865-S3000L0105-001-01

Fig 15 Schematic LSA representation of example 3

9.4

Example 4: Failure of device 01, repairable at operational site



ICN-B6865-S3000L0060-002-01

Fig 16 Failure of device 01, repairable at operational site

Event:

Failure of Device 01 (component itself is repairable at operational site)

Applicable to: All

S3000L-A-12-00-0000-00A-040A-A

Chap 12

LSA candidate: Equipment 401 and device 01

Complete maintenance procedure description:

Refer to [Fig 16.](#)

- 1 Remove faulty equipment 401 at customer's operational site
- 2 Remove faulty device 01 from equipment 401
- 3 Remove faulty part 02 from device 01
Discard/dispose faulty part 02
- 4 Receive spare part (new part 02) from customer's maintenance depot
Repair faulty device 01 by installation of new part 02
- 5 Re-install device 01 into equipment 401
- 6 Re-install equipment 401
⇒ Failure rectified, Product can be used again
- 7 Re-order new part 02 from industry to restock customer's maintenance depot

For schematical LSA representation, refer to [Fig 17.](#)

Rectifying tasks for the event:

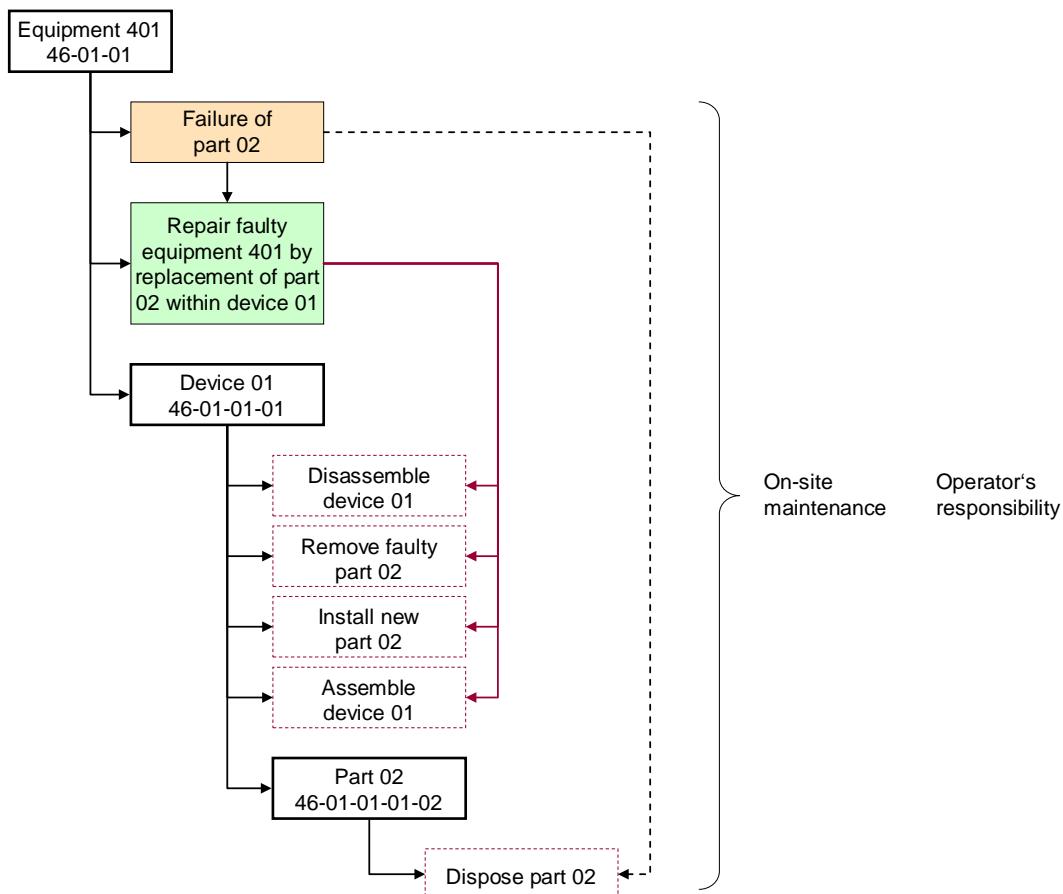
Repair equipment 401 by re-installation of repaired device 01
Repair device 01 by replacement of part 02

Follow-on maintenance related tasks:

Discard/dispose faulty part 02

Other follow-on activities:

Re-order new part 02

Schematical LSA representation of rectifying and follow-on tasks:


ICN-B6865-S3000L0106-001-01

Fig 17 Schematic LSA representation of example 4
Note

Example 4 demonstrates a possible solution for a more complex situation as in the previous examples. On closer examination the example describes a classical problem how to document rectifying maintenance tasks. Device 01 and equipment 401 are both repairable. At the same time device 01 is a component of equipment 401. That means both can serve as typical LSA candidates which can be repaired by the replacement of components. This situation can be described as "full LSA candidate contains another full LSA candidate". The way to handle this situation as described in example 4 is one possible solution. The failure is documented against equipment 401 (in this case, testability and detectability properties must allow the failure detection and localization on the level of equipment 401). Consequently, the rectifying task is also documented against the equipment 401 (here *Repair faulty equipment 401 by replacement of part 02 within device 01* and the subsequent re-install of the repaired device 01). The detailed activity within device 01 (remove, disassemble, assemble, re-install), can be documented as supporting tasks within device 01 as a partial LSA candidate (the rectifying task will refer to these supporting tasks). The exchange of part 02 will be documented as a work step within the rectifying task to trigger the need for the spare part itself (in this case part 02).

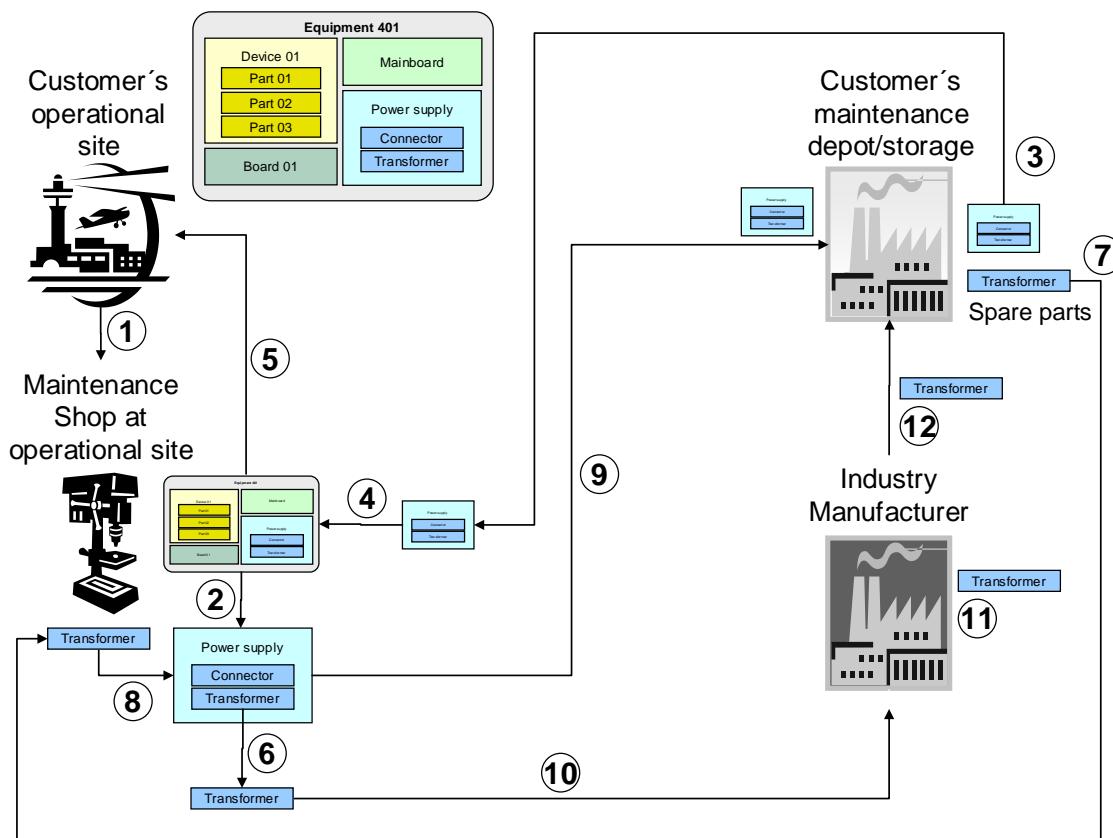
Depending on the situation that the entire Product is waiting for the re-installation of a faulty equipment/component after any repair activity, the representation within the maintenance tasks in the LSA database is of different character. The need for spare parts is influenced on this decision (wait for repair or not). In general, two main situations can occur, shown in [Table 17](#):

Table 17 Different spare part triggers

Product waiting situation	Consequence
Product is waiting for repair of equipment	The entire equipment is not needed as a spare part, because the same equipment will be re-installed (after a successful repair) into the Product.
Product is not waiting for repair of equipment	The entire equipment is needed as a spare part, because a new equipment will be installed to repair the Product to enable the reuse the Product as soon as possible. The repair of the equipment will be carried out later, when the Product is already in use again.

9.5

Example 5: Complex maintenance procedure on several levels



ICN-B6865-S3000L0061-002-01

Fig 18 Major failure of equipment, only repairable at higher level

Event:

Failure of power supply (component itself is repairable at operational site by replacement of part, part itself is repairable at industry site)

LSA candidate: Equipment 401 and power supply

Complete maintenance procedure description:

Refer to [Fig 18](#).

- 1 Remove faulty equipment 401 at customer's operational site
- 2 Remove faulty power supply from equipment 401 at customer's operational site
- 3 Receive spare part (new power supply) from customer's maintenance depot
- 4 Install new power supply into equipment 401
- 5 Re-install equipment 401
⇒Failure rectified, Product can be used again
- 6 Remove faulty transformer from power supply
- 7 Receive spare part (new transformer) from customer's maintenance depot
- 8 Repair faulty power supply by installation of new transformer
- 9 Send back repaired power supply to restock customer's maintenance depot
- 10 Send back faulty transformer to industry
- 11 Repair faulty transformer at industry site
- 12 Industry to send back repaired transformer to restock customer's maintenance depot

For schematical LSA representation, refer to [Fig 19](#).

Rectifying tasks for the event:

Repair equipment 401 by replacement of power supply

Follow-on maintenance related tasks:

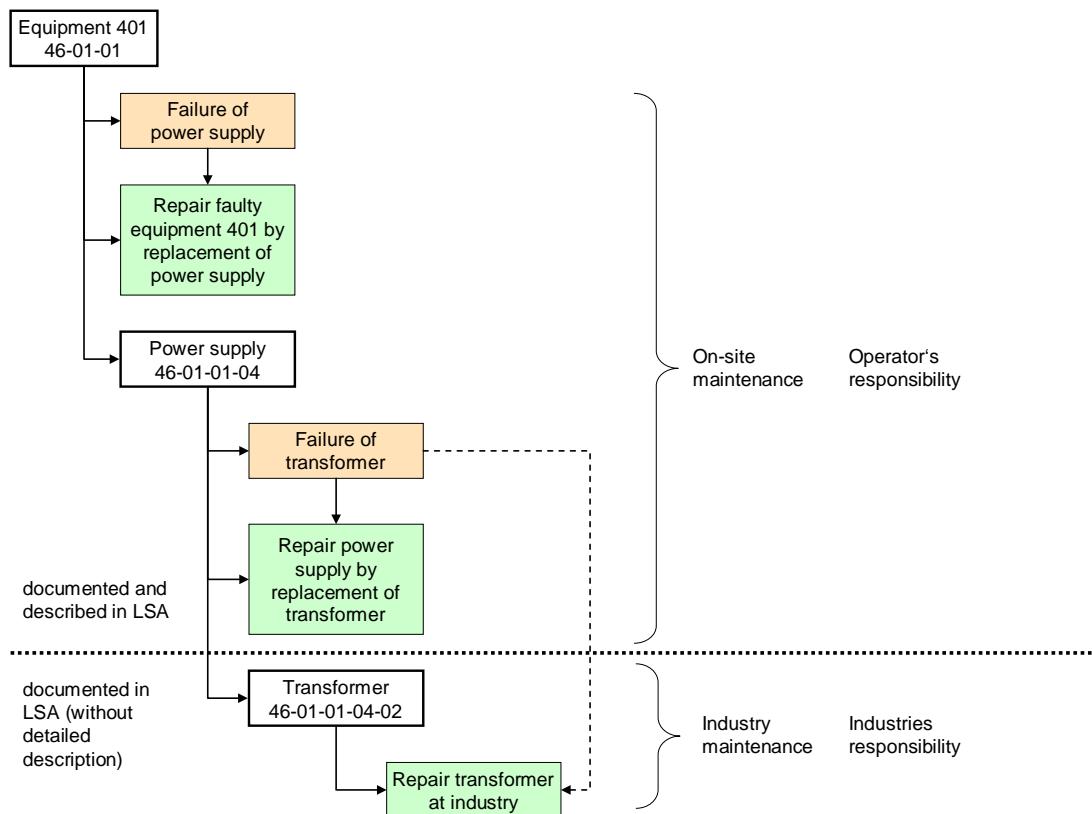
Repair power supply by replacement of transformer at operational site

Repair transformer at industry site

Other follow-on activities:

Send back repaired power supply from operational site to customer's maintenance depot

Send back repaired transformer from industry to customer's maintenance depot

Schematical LSA representation of rectifying and follow-on tasks:


ICN-B6865-S3000L0107-001-01

Fig 19 Schematical LSA representation of example 5

Chapter 13

Software support analysis

Table of contents

	Page
Software support analysis	1
References	2
1 General	3
1.1 Introduction	3
1.2 Objective	3
1.3 Scope	4
2 Software support analysis in the different project phases	4
3 Breakdown concept	5
3.1 Functional and physical breakdown principles	5
3.2 Physical software categories	6
3.2.1 Field-loadable software	6
3.2.2 Shop-loadable software	6
3.2.3 Resident software/firmware	6
3.3 LRU and SRU aspects	6
3.4 Functional software categories	6
3.5 Data	7
4 Software support analysis	7
4.1 SSA process - commonality with the LSA business process	7
4.2 SSA candidate selection	8
4.2.1 Selection of physical candidates	8
4.2.2 Selection of functional candidates	8
4.3 Maintenance relevant events for software (support initiators)	9
4.3.1 Operational events	9
4.3.2 Technical events	9
4.3.3 Software improvement requirements	9
4.3.4 Software failures	10
4.4 FMECA/FMEA aspects	11
4.5 SMA for software	11
4.6 LORA aspects	11
4.7 Software support tasks	12
5 Software support concept framework	12
5.1 Software support profile	13
5.1.1 Software support levels	13
5.1.2 Software support roles	14
5.1.3 Use cases for software support activities	14
5.2 Support classes	14
5.2.1 Processes	15
5.2.2 Product	15
5.2.3 Environment	15
5.3 Support functions and software related support tasks	15
5.3.1 Operational servicing support	16
5.3.2 Management support	19
5.3.3 Software modification tasks	20
6 Supportability factors	22
6.1 Compatibility matrix	22
6.2 Deployment	22
6.3 Documentation	22
6.4 Loading/unloading and installation	23

6.5	Transportation on hardware carriers	23
6.6	Expansion capability	23
6.7	Modular software design	23
6.8	Modification frequency	24
6.9	Recovery	24
6.10	Safety integrity	24
6.11	Security	25
6.12	Size	25
6.13	Skills	25
6.14	Software distribution	26
6.15	Standardization	26
6.16	Technology	26

List of tables

1	References	2
2	Failure classes	10
3	Software support levels	13
4	Tasks for provision of new functionality and/or capability	16
5	Tasks for recovery of a system and for problem reporting in case of failures	17
6	Organizational tasks	18
7	Maintenance support tasks	19
8	Software modification tasks	21

List of figures

1	Software support analysis in the different project phases	4
2	Aspects of an SSC	12

References

Table 1 References

Chap No./Document No.	Title
<u>Chap 3</u>	LSA business process
<u>Chap 10</u>	Scheduled maintenance analysis
<u>Chap 11</u>	Level of repair analysis
S4000P	International specification for developing and continuously improving preventive maintenance
MSG-3	A4A MSG 3 - Operator/Manufacturer Scheduled Maintenance Development

Chap No./Document No.	Title
RCM	Examples include MIL-STD-1843 (USAF) - Reliability-centered maintenance for aircraft, engines and equipment MIL-STD-2173 (AS) - Reliability-centered maintenance for naval aircraft, weapon systems and support equipment UK MoD DefStan 02-45 - Requirements for the application of reliability-centered maintenance - techniques to HM ships, submarines, royal fleet auxiliaries and other naval auxiliary vessels
RTCA/DO-178B	Software Considerations in Airborne Systems and Equipment Certification
SAE JA1004	Software Supportability Program
SAE JA1005	Software Supportability Implementation Guide
SAE JA1006	Software Support Concept

1 General

1.1 Introduction

In modern Products, software aspects are of increasing importance. More and more functionality is supported or realized by complex software packages. Concepts and processes are established for hardware components to guarantee full supportability of the Product during the entire life cycle. The appropriate tool to achieve this goal for hardware is the LSA process. The same requirements apply for software. Hardware and software are of equal importance for the proper function of a Product. For this reason, an analysis methodology called Software Support Analysis (SSA) can be applied in order to develop an adequate Software Support Concept (SSC). SSA is a methodology to analyze software with the purpose of supplying the customer with all necessary information to establish a cost effective SSC. This includes all equipment, tools, personnel, documentation, infrastructure and required skills and training.

An SSC takes into account all activities in order to enable a continued usage of software within a Product. To find a common understanding of software support, a standardized concept for software support documentation is recommended to be part of an LSA program plan, harmonized between the contractor and the customer.

Similar to the analysis activities for hardware, software must be analyzed with regard to its operational and maintenance requirements. Special operational aspects are of particular interest for the end user of a Product because of the influence on the day-to-day business. Aspects covering software modification, such as bug-fixing and software improvement (eg software upgrades), must also be taken into account.

1.2 Objective

This chapter provides the logistic analyst with guidelines for handling the specific requirements concerning software in the environment of maintenance and operation. The interrelation between software and hardware is clearly defined and the method of integrating software aspects into the overall LSA process is explained.

For software itself, a clear distinction between operational and maintenance aspects and real software modification must be established. For example, operational aspects can include the simple act of loading working software or required data to equipment within the Product, whereas software modification covers aspects that deal with the change of the source code to

fix a problem, improve the performance of a software package, or adapt to changes in procedures, data, or systems that affect the software performance.

1.3

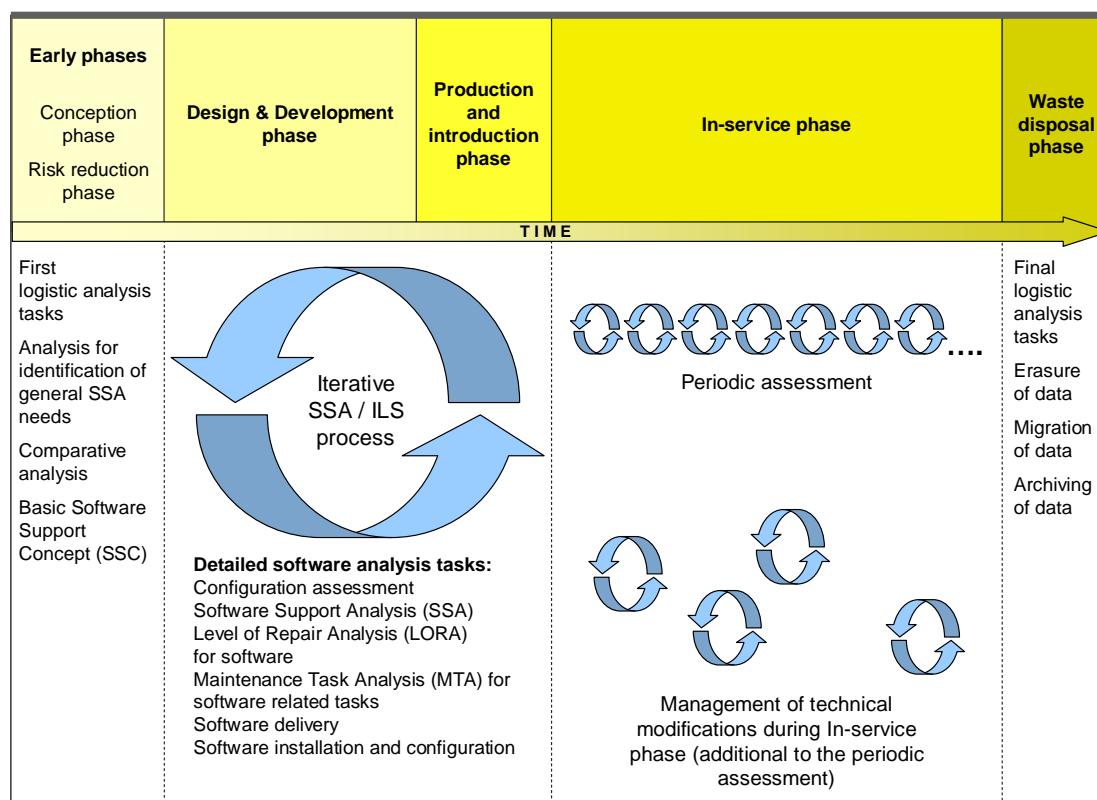
Scope

This chapter is directed at ILS managers and/or LSA analysts for both contractors and customers who analyze Products containing software. With the means of LSA and SSA, an appropriate support concept for such equipment can be established. With the help of the results of the various analyses, the customer's needs concerning supportability, readiness and operability of hardware that contains software can be assured.

2

Software support analysis in the different project phases

Similar to the application of LSA, SSA is an analysis activity, which can be applied throughout the entire life cycle of a Product containing software. In the different life cycle phases various analysis activities must be performed, refer to [Fig 1](#). Depending on the phase, some activities increase in importance, others decrease. For example, during the design and development phase, the application of SSA is of high importance as it can influence software architecture and design in terms of supportability requirements for future needs to load, service, or modify the software packages. In an early concept phase, SSA helps to identify basic software support requirements. During the in-service phase, similar aspects as those for hardware are valid. Technical modifications will force the analyst to repeat the analysis of the logistic concepts for the affected hardware/software. For this reason, an iterative SSA/ILS process will occur at different depths, depending on the extent of the technical modifications. These activities are basically the same as those in the design and development phase and in the production and introduction phase, but normally to a smaller extent.



ICN-B6865-S3000L0062-001-01

Fig 1 Software support analysis in the different project phases

In the disposal phase, handling software and/or data during the disposal of hardware is important and care must be taken to ensure data is dealt with in accordance with the project's requirements. Sensitive data must be erased from scrapped hardware (eg, in a military environment). For archiving purposes, existing data must be preserved from disposed hardware components. If existing data is needed for further operation of newly implemented systems, the possibilities of data migration must be analyzed and concepts for data migration must be established.

3 Breakdown concept

[Chap 4](#) provides guidance on the development and assignment of BEIs for hardware and software with examples of physical and functional breakdowns. It is recommended to follow this guidance and the examples in developing the breakdown relationships of software in the context of the hardware.

For software, it is possible to obtain a breakdown element identifier similar to that of hardware by following the concepts of physical or functional breakdowns. However, due to different characteristics, it can be difficult to associate software within the traditional physical/functional breakdown. This is especially true for advanced systems (eg modular avionics), where the physical location of a specific software can be vague. Often, software is not a clearly defined physical entity, therefore it can be difficult to identify the appropriate indenture level for the software item within a physical or functional breakdown. Even in the case of a clear assignment of software to hardware, the question can arise as to where the software can be allocated:

- Is it part of the hardware element, where the software is loaded?
- Is it part of the hardware element, where the software is executed?
- Is it part of the hardware element, where it physically resides?

The breakdown, in this context, is different from the traditional physical/functional hardware breakdown and it is important to establish a consistent approach. The physical breakdown can be used to identify software within a Product breakdown for operational purposes, and provide the necessary software view for traditional maintenance purposes. The functional breakdown is a software engineering view that shows how software will be modified and integrated, and provides the relationships between the different software elements that must be considered when a redesign is performed.

3.1

Functional and physical breakdown principles

Software items identified within a physical breakdown are normally elements that can be handled by the operator. This is the software that must be considered for operational aspects such as, for example, loading/unloading. The software can reside on different indenture levels of the Product breakdown.

The allocation of physical software to a specific indenture level is actually driven by the need to maintain an adequate configuration control of the Product and also to highlight the logistics impact that such software can have on standard operation and maintenance. Though traditionally software was assigned to a hardware element, new configuration management issues and approaches have changed this concept to the extent that within the LSA, software can be its own replaceable unit (LRU or SRU) or a part/component in its own right. Refer to [Para 3.3](#).

Software items that are identified within a functional breakdown are elements that are used to describe the functional/design aspects that are important for the software designers, especially since such a breakdown usually indicates the need for integration with other software. This kind of breakdown must be considered for software modification support. Therefore, it is important that the functional breakdown follows the software design as closely as possible.

3.2 Physical software categories

There are three categories of physical software: Field Loadable Software (FLS), Shop-Loadable Software (SLS) and resident software (also called firmware). The concept for the three categories is established in the configuration management principles that control the configuration of a specific Product.

3.2.1 Field-loadable software

FLS is any software that can be installed on one or several pieces of equipment on a system/Product without the need to dismount the equipment from its existing installation location. FLS is considered an item in its own right within the Product breakdown and therefore, a modification of FLS results in a change of the FLS part number and a change of the configuration of the system where it resides. However, FLS loading does not cause a change in hardware part number in order to avoid a forced disassembling of a part merely to change the part number on that part. The Product configuration is affected because the function of the Product can change when using a different piece of software.

3.2.2 Shop-loadable software

Shop-loadable Software (SLS) is any software that can be loaded into an LRU, but requires that the LRU be dismounted from its installation on the Product. A replacement of any hardware component is not required. A modification of the SLS not only changes the part number of the SLS, but also the part number of the LRU where it is loaded. This is the same as when an SRU is replaced by a different one, in order to maintain the form, fit and function principle that drives configuration management (when replacing software by different software, the function is changed). In this case the Product configuration is affected, only due to the change in the hardware part number.

3.2.3 Resident software/firmware

Resident software (also called firmware) is any software that can be loaded into an LRU or SRU but requires that LRU/SRU to be dismounted from its installation on the operational Product and requires the replacement of a component. Although such resident software can have its own part number when it is installed on a component or loaded onto an SRU, the part number is changed for the component or SRU where it is installed. This also entails a change to the part number of the next higher assembly.

3.3 LRU and SRU aspects

The easiest way to handle software within an LSA Product breakdown is to identify such software as LRU, SRU or part/component, depending on the software category. This simplifies not only the overall Product configuration management, but it also clarifies how to handle the Product configuration when using FLS and there are no tags on the target hardware regarding the software it contains. This becomes a non-issue when the configuration is handled at the next higher level.

Similarly, a loading task due to a hardware repair or software update is quite easy to integrate into the overall maintenance tasks because a specific (software) SRU or LRU is affected. In principle, it is no different than performing a task on a hardware SRU/LRU and subsequently performing additional tasks at the next higher breakdown level. Therefore, loading software after a hardware repair is very similar to, for example, replacing a consumable.

3.4 Functional software categories

The functional software breakdown follows the software design requirements, as it is mainly oriented towards software modification. The classic Computer Software Configuration Item, Computer Software Configuration and unit level views can be used but, contrary to "traditional" software engineering, it is also necessary to provide the higher levels of abstraction (subsystem and system level), as this indicates interdependencies and need for integration, including the need for system or subsystem level integration rigs. In this context, software assembly items within a functional breakdown can support the grouping of functionality that is documented as a

whole using the same principle as for artificial assemblies within physical breakdowns). Such artificial items can be used to provide a helpful additional level of abstraction without changing the overall structure and functionality of the software design.

In the event that a major version of the software is generated that significantly changes the software structure, functionality or supportability elements (eg, change of programming language of a specific module) that can entail a modification of the support tasks and/or support infrastructure, it is recommended that a breakdown element revision identifier is used. It does not provide any added value to perform the full analysis simply because there is a change in an algorithm, but it can be essential if, for example, the software risk class changes.

3.5

Data

Data can be handled in a similar way to software, considering it from both the physical and functional point of view, with the peculiarity that the physical data can be loaded/unloaded but also prepared. This is different from the pure software aspect, since software modification is a design activity, while a data preparation can be considered an operational activity. However, differentiating between the two is not always clear cut.

In general, data can be included in the LSA database in the same way as software, including preparation if applicable, but would be also be included in the functional breakdown in order to ensure the configuration and dependencies are kept. For example, a software modification can entail a modification in the mission preparation systems.

Whether data is considered as software or as a special category within the LSA database must be discussed with and agreed by the customer. Though standards such as RTCA/DO-178B include data under the software heading, there are merits to treating it as a separate element for logistics purposes.

4

Software support analysis

SSA is a consistent methodology to guarantee proper software supportability throughout the requirements, specification and design phases in order to define the most cost effective support concept that meets the operational and software modification requirements. Establishing the necessary support infrastructure before the Product enters in the in-service phase must be ensured. The main goals of an SSA are the establishment of supportability requirements concerning software in the early program phases and the influence of the software design to ensure supportability for both, system operation and later modification processes. The outline for a stand-alone software supportability program is described in SAE JA1004, Software Supportability Program.

4.1

SSA process - commonality with the LSA business process

In general, the SSA process covers similar logistic aspects as described in [Chap 3](#) for the LSA process. For a Product that contains software as a basic element of the complete system, a number of software support activities must be applied depending on the situations, which are the trigger for the support. In this area, two main categories of analysis are identified:

- Operational/maintenance aspects (eg loading of data or software to the corresponding hardware, system recovery, data transport and archiving)
- Software design aspects (eg real modification of software for bug-fixing or improvement purposes)

When carrying out SSA, it is necessary to consider the relationship and differences between the two task categories, and the representation of software items in the functional and physical system breakdowns. One aspect is that support activities that do not involve modification are typically analyzed against physical breakdown items. In all cases, the analysis results concerning operational aspects are recorded in the LSA database, including the documentation of tasks such as the loading of data and software to the carrying hardware element within the Product breakdown. For example, the change of equipment can require that software and/or

data must be reloaded to the new equipment and configured after or before installation into the system. The usage of hardware for special purposes can require the loading of a special software or data package to support this kind of operation. Nothing will be modified within the software itself.

The software modification itself is another area in which the software source code is normally directly influenced and changes in the source code are performed. Since the requirements for these activities are different from the operational aspects concerning software, the documentation of these activities can be separated from the operational aspects. The breakdown elements, to which these activities can be addressed, are functional software breakdown elements. Software packages which are analyzed concerning their requirements for real software modification tasks can be treated as SSA candidates similar to LSA candidates for hardware.

4.2

SSA candidate selection

In general, SSA candidates are elements of either physical or functional breakdowns that are subject to any kind of SSA. The candidates must be selected with regard to operational or supportability significance. Taking into account that software support activities are divided in operational/maintenance and software modification categories, candidate selection is carried out within these areas separately.

4.2.1

Selection of physical candidates

4.2.1.1

Software

An SSA candidate selection within a physical breakdown must take into consideration all software items that can be loaded and/or installed separately by the operator. The candidate itself can also be the hardware item that is the carrier of the software.

For distributed software, the candidate can be a subsystem or a system within the physical breakdown. Loading and/or installation tasks are typically documented against the corresponding hardware. Other operational tasks can also be documented against a software breakdown element (eg transportation or distribution of software).

4.2.1.2

Data

Data is normally of electronic nature and is handled in a similar way to the corresponding software. Therefore, the support of software and dependent data can normally be analyzed together (although not in the same way). Each SSA candidate must be analyzed for special support aspects for data. This includes, but is not limited to:

- Is a data preparation process required that needs special software or hardware?
- Is a data validation process required?
- Are special transmission media or networks required?
- Are there special security aspects to be considered for data loading?
- Are there special compatibility requirements with the existing executable software?
- Are there special requirements concerning size and format (eg databases, file formats)?

4.2.2

Selection of functional candidates

An SSA candidate identified in a functional breakdown must take into consideration all software items that are subject to software modification activities. Each software item can be analyzed using the following questions in order to decide whether it is subject to a software modification activity:

- Are there any support initiators to be expected for the software items that require a modification of the software source code?
- Does the software item in the breakdown include all software within the system or within the equipment?
- Does the software item use a separate design?
- Does the software item require special hardware and/or software tools for development?

- Does the software item contain proprietary software such as run-time libraries or COTS elements?
- Are there different versions of the same software item for usage on different platforms?
- Are there deviations of the software items to the general design environment?

4.3

Maintenance relevant events for software (support initiators)

Similar to the methodology of LSA for hardware, the starting point of documenting the maintenance relevant aspects is the identification of any event that initiates maintenance activities. These events can also be described as software support initiators. The event driven methodology can also be applied to SSA, however, the events concerning software are somewhat different from the events for hardware. They can be grouped according to the implication of:

- Operational events
- Technical events
- Software improvement requirements
- Software failures

4.3.1

Operational events

This type of event is related to an operational requirement. Operational events can be documented within the related hardware item. The same applies to the corresponding operational task. Typically, the aspects covered are:

- Hardware maintenance/repair of computing hardware

After hardware maintenance, different types of tasks can be necessary (eg, reloading of data and/or software, installation and configuration of software packages on new installed equipment).
- Usage (mission) preparation

Besides the installation of additional hardware, the preparation for usage of a Product can require the installation of supporting software packages or the loading of special mission data.
- Post mission requirements

After usage of a Product, data created during the mission must be unloaded or archived. Software packages that were required for a special mission must be de-installed and the Product must be reconfigured to a standard configuration.

4.3.2

Technical events

Besides the hardware maintenance, other technical events can trigger software support activities. These events normally require adaption of the existing software packages to the modified environmental preconditions. Typically, the aspects covered are:

- Changes of the parent hardware
- Changes of the parent software (eg upgrade of operating systems, of corresponding database systems or of firmware)
- Changes in technology of inter-operating systems
- Changes in technology of network interconnectivity

4.3.3

Software improvement requirements

Software is typically subject of a permanent improvement process. Normally, software release changes take place in particular intervals depending on the complexity and size of the software package. Besides the fixing of real software failures, the primary inputs for new releases are dedicated end user requirements or changes in the environmental preconditions. Improvement

in this context means the introduction of new features to improve or extend functionality or the usage comfort of an existing software package.

4.3.4

Software failures

System failures initiated by software are the main support initiators. The reaction to these failures depends on the severity of the failure. Not every failure caused by software will initiate a software modification activity.

For software failures, it is recommended that a classification concerning severity be introduced. For each failure class, a sequence of actions can be defined within an SSA. In every case, proper documentation is required to support later diagnosis.

Table 2 Failure classes

Class	Description
Minor	<p>In general, these are failures concerning the user comfort. In many cases, these events are not real failures but only poorly designed features of the software or they are real failures that can be bypassed by other features of the software or an alternate handling.</p> <p>However, it is recommended to collect, document and report these minor events to the software support organization. This information can support the incorporation of improvements into a future release of the software package. Especially the correction of marginal insufficiency can considerably increase user's acceptance.</p>
Medium	<p>These are events that cause decreased functionality of the system. If this type of event occurs, the user is not able to continue the work as expected, however, the Product as a whole works without severe disturbance (eg no shutdown is needed). The ongoing usage of the Product is possible with reduced performance or even with no restrictions.</p> <p>These events normally need a timely correction, or a defined workaround to deal with the failure until the final correction is made with the help of a failure correction procedure. Normally, a software modification is necessary.</p>
Major	<p>A major software failure is considered as an unacceptable event. The failure causes the shutdown of the entire Product or the necessity to operate the Product under very restricted conditions. The normal capability of the Product is significantly degraded.</p> <p>These events need a correction implemented as soon as possible. In case of emergency situations, the software support organization must react as soon as possible to recover the entire Product to full operation. The reaction times must be discussed with and agreed by the customer.</p>

A software failure cannot be treated exactly like a hardware failure. In the case of a hardware failure, any relevant maintenance action such as a repair or a replace can be clearly defined. In case of a software failure, it is not normally possible to identify the required activity immediately. For example, a restart of the Product can be sufficient to recover and the failure will not occur again. On the other hand, the failure can be severe and the Product performs a shutdown and must not be restarted in order to avoid the repetition of the failure and a further corruption of the entire Product.

An important aspect can be the corruption of data or executable code. This event can be initiated, for example, by a virus infection or by corrupted carrying hardware (eg the loss of functionality of a hard disk). In this case, there is no real inherent failure in the software source code. This event is handled as external damage.

4.4

FMECA/FMEA aspects

The results of a technical FMECA/FMEA are relevant for software since the FMECA/FMEA represents an analysis method carried out for the overall equipment/system (covering hardware and software aspects). For any identified failure mode, a decision must be made as to whether the associated functionality is dependent on or even provided completely by software. During the design/development of the software, this failure mode information can be used to design a software architecture that eliminates, or at least mitigates, the possibility of the software failing in such a manner as to cause a specific functional failure of the Product. In addition, the design can be instrumented so as to ensure that any such potential software failures are detected and recorded, including their associated information, and that the Product continues operation in a safe mode.

During the in-service phase, system failures due to software failures are drivers for software modification activities. A principle of the logistics FMEA is to group all of the software failures together that lead to one specific system failure, and even to group the specific system failures together that lead to the same software modification request. The analysis is based on the function that the software performs. However, it is typically not practical to carry out the FMECA/FMEA of the single functional failure effects down to the level of specific defects in the source code.

Another consideration during the in-service phase is not only to provide the necessary information to the software engineers about a specific failure, but also to prevent unnecessary maintenance tasks due to software failures. A software specific failure that is reported by built-in-test capability and which can be identified by the operator, will reduce the number of disassemblies (as you cannot rectify a software failure by replacing the equipment) and number of cases of no failure found related to hardware.

4.5

SMA for software

SMA such as S4000P, MSG-3 or RCM is not applicable to software packages in the same way as for hardware (for hardware, refer to [Chap 10](#)). Software failures are the result of unintended effects of the software design. Such failures cannot be avoided by the means of SMA since they correspond to design flaws. However, potential software failures can affect safety. It can be possible that the result of SMA carried out for a piece of hardware equipment which contains software, identifies scheduled maintenance tasks that are of operational character concerning software. However, the analysis remains an SMA for hardware, taking into account software elements. A stand-alone software package will not normally be a candidate for SMA.

An SMA on the software itself is a periodic assessment of the software during its life and not the scheduled maintenance tasks performed on it. Software, to a great extent shows a bathtub like failure profile with initially many failures. Bugs are fixed with successive software releases leading to a stable behavior. During later releases, failure rates may increase again due to the increasing degradation of the software as it is modified beyond its initial scope. A periodic assessment of the failure rate can determine whether the software is reaching its end of life due to increasing software failures due to modifications and alterations or because the added functionality can no longer be properly processed by the underlying hardware, thus forcing either a software rewrite or perhaps even a complete system redesign.

4.6

LORA aspects

The identification of that support level that a task will be performed can be provided by a Level of Repair Analysis (LORA) procedure (refer to [Chap 11](#)). For operational aspects, the performance of maintenance tasks related to software or data loading can be covered within a LORA for the hardware related tasks, or at least by a process that is very similar to it. The triggers for such tasks can sometimes not be the traditionally identified tasks, but can be due to external factors such as a new software release, or the need for specific operational software or data loading. If these tasks occur frequently enough, the result of the LORA can actually be different than the one that would be obtained when associating these tasks to the corresponding hardware.

For real software modification, detailed information about equipment (software development environment concerning hardware and software developer tools) and personnel capabilities is required. Also, contractual warranty and software ownership aspects can influence the determination of an appropriate support level for software modification activities. For logistics management tasks, the process also becomes important in order to determine the best location to perform tasks such as, for example, generation of software or data media.

4.7

Software support tasks

After the identification of the relevant support initiators (events), the appropriate tasks must be identified (operational/maintenance tasks or modification tasks). The goal is to define all operational, maintenance or modification relevant tasks, their related resources and further task characteristics, such as duration, man power requirements, preconditions or safety conditions.

Guidelines for the analyst regarding which type of software support tasks can occur resulting from different events are given in SAE JA1005, Software Supportability Implementation Guide. Refer to [Para 5](#).

5

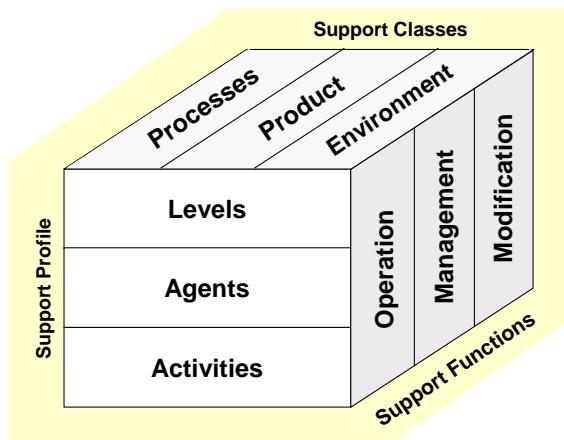
Software support concept framework

The achievement of a proper supportability framework is one of the most important design principles for software, in a similar way as for hardware. This requires the establishment of an SSC in an early stage of the software design process. The earlier the design pays attention to support requirements, the better the supportability will be. It should be easy to modify software if change requirements are addressed from different sources. Also loading, installation and configuration procedures should be as easy as possible.

To take into account all of these aspects, it is recommended to give a guideline to the logistic analyst using this specification. The guideline must cover all of the important areas concerning software support and must support contractual agreements between the customer and the contractor, which are agreed during the LSA GC. All of the aspects can be tailored to the required extent of the project.

Note

There are different perspectives of an SSC, all of which are interconnected and influence each other, refer to [Fig 2](#).



ICN-B6865-S3000L0063-001-01

Fig 2 Aspects of an SSC

The extent of a software development or procurement project depends on whether the process to establish an SSC can be a part of the overall LSA business process (refer to [Chap 3](#)).

Aspects of software support can be covered by the procedures of the general LSA business process. Relevant documents such as an Operational Requirements Document (ORD) or a

Customer Requirements Document (CRD) can also cover software support aspects concerning the required usage information. The LSA GC also covers the software support aspects and the relevant and contractual agreements must be written in the documents LSA Program Plan (LSA PP) and/or Guidance Conference Document (GCD). The candidates for a detailed SSA can be selected from the Candidate Item List (CIL). The rules for SSA candidate selection are established here, too.

It is recommended that for large and complex systems with a considerable amount of software, software supportability is treated as its own area. A separate SSA program plan can be created and implemented, and a separate Guidance Conference (GC) for software aspects can be held. However, the interaction between hardware and software development must be considered. Software is designed to address functionality to hardware components. Therefore, both must work hand in hand and processes that guarantee proper cooperation must be harmonized, documented, reviewed and put into practice. Refer to SAE JA1006, Software Support Concept.

5.1 Software support profile

The software support profile takes into consideration the location of the software support and the organizations and related persons that are the relevant players in a software support environment. Additionally, the processes regarding how the different instances work together in an integrated way must be discussed with and agreed by the customer.

5.1.1 Software support levels

The levels of support can be addressed for software support in a similar way to that for the maintenance levels within the LSA for hardware. The support levels for software can be different from the hardware support levels and can be used as input to the LORA decision process for the software. Also, classification is similar to the maintenance levels for hardware, but some special aspects concerning software can be considered additionally. Refer to [Table 3](#).

Table 3 Software support levels

Support level	Description
Level 1 (Operational level)	This support level describes the location at which software is in operational use. Within this level, normally only simple support tasks can be carried out. These tasks are covered by the operational servicing support.
Level 2 (Intermediate support level)	An Intermediate support level can cover all support activities which are related to operational servicing support but also to management support. User support in the form of help desks and hotlines can also be covered by intermediate support level. An example for intermediate support level is the Service Repair Vehicle concept. The Service Repair Vehicle is able to perform, for example, communications, download of software or firmware update in a flexible way (like a quickly available help desk). It is located near to the operational area but is not part of the operational site.
Level 3 (High support level)	This support level is normally located at a central site. Activities from the management support can be covered here as well as modification support. It depends on the software packages themselves (eg complexity, size, modularity, programming techniques) and on the equipment (eg tools, personnel, facilities) of the support sites as to how far the capability to modify software packages is available.

Support level	Description
Level 4 (Software vendor and/or original developer)	The vendor level is normally the best skilled support level for the high demanding tasks concerning software modification. In many cases, the vendor is the original developer of the software, equipped with all necessary developer tools and personnel. Often, this level is selected for modification support because the user itself does not have the required capabilities for these complex support tasks. Additionally, aspects of responsibility and liability must be considered, which can force the user to address software modification to the original vendor.

5.1.2

Software support roles

As a first step to clarifying software support roles, customer (buyer and users) and supplier (vendor and supporters) roles must be clarified. As a second step, the roles concerning software usage and support must be allocated to personnel who are able to offer the required services. These roles can be:

- Simple user without any rights to modify or to recover any software system (not a real support role, but in general only a service receiver). The main task of a simple user is to keep the software operationally running and report any system/software failures.
- Power user with some basic knowledge and the ability to perform a system recovery. This kind of role can be involved in the software support by performing simple operational support actions in order to keep the software running, such as installation and loading of data. Normally a power user is not involved in any software modification activities.
- System administrators are power users on a higher level. They must be able to cover all activities concerning operational support and eventually management support. Normally system administrators are not able to modify software itself, but software configuration like changing of performance parameters and loading of additional software packages or installation of updates are typical administrative tasks. System administrators are normally concerned with the direct support for the simple users or power users in working with the software packages. Normally a system administrator is not involved in any software modification activities.
- Service provider, who is able to provide qualified operational support and additional services, such as a hotline. Service providers can be located onsite with the customer and vendor, and in some cases, are able to provide modification support.
- Vendor or original software developer, who is able to provide modification support up to the highest support level. Vendor support is normally located off site.

All these roles must be equipped with the required skills, capabilities and rights necessary to carry out their tasks. The roles, including their profile, is documented in the SSC. The limits of competence must be clearly defined, documented and accepted by all affected personnel.

5.1.3

Use cases for software support activities

Use cases should be described for the different support roles and support levels, indicating how the different resources at different locations work together and which processes are the basis for the cooperation. This support scenario must be agreed by both, the customer and the contractor, and then implemented and controlled.

5.2

Support classes

The quality aspect of an SSC is extended to all objectives involved. It is recommended that quality standards be used and implemented to ensure that rules are adhered to. This is accomplished by three software support classes.

5.2.1 Processes

All processes being established must be discussed between the customer/user and the software support provider. A written documentation of how these processes are implemented must be provided. Process characteristics including inputs, outputs, controls and resources must be clearly addressed:

- Is there any written documentation of the process (eg flowcharts, descriptions, requirements, responsibilities)?
- Is an accepted standard used to describe the relevant processes (eg international or company internal)?
- Are there any real measurable performance control parameters available and what are the measures for performance control?
- Are inputs and outputs, procedures, requirements and an effective control mechanism defined and documented?

5.2.2 Product

Supportability aspects should be addressed as early as possible in the software Product characteristics. Inherent quality characteristics of software depend on how well or badly the software has been designed. Requirements for a good software design include many aspects starting with the software specification and ending with a perfect user handling, supported by a proper graphical user interface. Typical quality aspects are:

- Modularity
- Changeability and expandability
- Simplicity and easy handling
- Stability and consistency
- Performance concerning processing speed
- Instrumentation (in the sense that the software is purposely instrumented for an understanding of how it is operating)

The better these aspects are considered in the development process, the easier it is to handle and to support the software package. However, even well designed software can cause problems by implementing a bad support concept.

5.2.3 Environment

Environmental aspects which affect quality start with the personnel being involved in the usage and/or support of software. To implement proper software support, it is essential that personnel characteristics are thoroughly addressed:

- Who is the right person for a software support job?
- What are the required skill levels and how much experience is recommended?
- How many persons are required to guarantee a trouble-free software support?
- What is the level of motivation of the personnel?
- What employee turnover can be expected?

Not only personnel aspects influence the quality of the environment. Simple aspects such as the quality of physical facilities, computer workplace equipment and size of offices can affect the quality of the support services themselves.

5.3 Support functions and software related support tasks

Maintenance tasks related to software can be divided into different categories. This depends on:

- Is the task simple software loading/unloading task or a simple software transportation task (eg from a device A to a device B)?
- Is it necessary to remove hardware from a system to load/unload software?
- Is the maintenance task connected to any case of failure recovery or documentation of any trouble related to software problems?

- Is any case of software modification involved?

Note

Software modification leads to a wide area of special support tasks, which must be addressed in detail.

5.3.1

Operational servicing support

The operational servicing tasks describe all activities associated with the day-by-day operation of the system. However, these tasks can have a different quality and depth of impact on the usage of the system (eg downtime, test requirements). In general, these tasks are rather simple tasks carried out by the user on support Level 1 or Level 2, which only require the software package to be available on a compatible medium plus a corresponding installation manual. The method of installation must be taken into account. On a stand-alone system, a system administrator will have some other requirements than with a large computer network system. In modern network architectures, software and data installation and loading will be carried out with the help of software deployment systems supported by corresponding tools.

[Table 4](#), [Table 5](#) and [Table 6](#) give an overview of what kind of tasks can occur, including the further definition of some terms concerning the handling of software packages. These tasks will be grouped in three subcategories:

- Tasks for the provision of new functionality and/or ability
- Tasks for recovery of a system and for problem reporting in case of failures
- Organizational tasks

Table 4 Tasks for provision of new functionality and/or capability

Type of task	Description
Software installation	Installation of a software package means the pure provision of a possible functionality realized by the means of software by carrying out an installation routine.
Software de-installation	De-installation of a software package means the definite removal of a possible functionality realized by the mean of software by carrying out a de-installation routine or by simple deleting of files from a storage device.
Software loading	Software loading is a task that is one step higher than the pure installation. In addition to the pure installation routine, which is part of the software loading, the configuration of software must be considered. In many cases, this step of software loading is more complex than the pure installation.
Software configuration	In this context, the term configuration means the change of internal parameters (eg operational parameters, granting of user rights, performance parameters, path information, database connections) for the operation of the software. This action requires no new installation of the software package itself. For the change of configuration parameters of the software, the functionality of the software package itself is sufficient. For the configuration, it is important to have sufficient access rights (eg administrator account for login into a software) to realize these changes.

Type of task	Description
Data loading	<p>Normally, data will be loaded with the help of the existing functionality of an installed software package in order to provide special capabilities to the Product. The border between software and data in this case is flexible, depending on the type and format of the data that must be loaded into the existing system. Data loading tasks are also typical tasks for preparation of operation or of a mission.</p> <p>Data does not change the basic functionality of the installed software package. For example, the maps in a navigation system can be considered as data, the software of the navigation system uses this data, for example, for the graph of a city map. With the loading of another data set (eg package of another country), the functionality of the navigation system will not change, it will stay a navigation system. However, the usage can possibly change, by the allocation of data.</p>

A special aspect of installation and loading of any software/data is the embedded software or firmware respectively. In this case, the allocation of new functionality or capabilities can be a more demanding activity. Normally, some special tools and support equipment (eg, to gain access) are required to transfer the new software code or the new data to the storage devices of embedded software (eg resident storage chips on a computer main board). These requirements are identified within the Maintenance Task Analysis (MTA) of the installation or loading task for the embedded software or firmware.

For each change of software by installation/loading it is recommended to define how proper function of the Product after the change can be ensured. The necessity of a functional test and a confirmation of proper work by an adequate skilled specialist must be investigated thoroughly and, if required, documented (eg, in the installation instructions).

Table 5 Tasks for recovery of a system and for problem reporting in case of failures

Type of task	Description
Software recovery	<p>Software recovery includes all activities of basic diagnostic and simple recovery actions such as a reboot/restart of a system after a system or software shutdown. Normally, these activities can be carried out safely by low skilled personnel. Even in case of a successful recovery to full system operation, a diagnosis of the system is strongly recommended. All available data concerning the problem occurrence and the recovery actions should be carefully documented for evaluation. This can lead to a number of outcomes to immediately rectify the cause of the problem or to sustain full system operation by alternate means. If the underlying problem cannot be solved within a short time, a temporary downgrading of system capability can be considered.</p>

Type of task	Description
Recovery problem reporting	<p>In any event of a system failure involving software, the generation of a problem report is recommended, even after a successful simple recovery, such as restarting or reloading. This problem report contains as a minimum:</p> <ul style="list-style-type: none"> – What was the system condition/configuration/version (software and hardware platform) at the time of failure? This information is important in the fault investigation on transitory effects, where the ability can be needed to reproduce the failure effect in an appropriate test environment or reference system. – Type of software failure? – Was a software recovery action successful (yes/no/partial)? – Severity of failure? <p>Problem reports must be communicated to the relevant personnel who would investigate the problem further to establish its priority and determine any workarounds and any needs for temporary operational limitations.</p>
Post operational data extraction	<p>Software support tasks carried out immediately after operation will generally be concerned with data extraction (for operational and/or engineering analysis) and fault investigation, and non-logistics functions such as the sanitation of classified software/data. With respect to data-related functions, a range of data administration tasks can be required (eg the provision and upkeep of data which is routinely needed by software a program to fulfill its operational role). Fault investigation tasks will relate to both activities on the prime system and to the deeper-level testing carried out on remote reference systems which replicate or simulate the prime system's environment.</p>

All tasks related to recovery of a system do not normally contain any software modification activities. However, recovery of a system can require the involvement of an updated software package.

Problem reporting can be a starting point of the need for modification. Even if recovery tasks were successful, it can be necessary to modify software to rectify the failures that occurred to avoid the repetition of the failure situation.

Table 6 Organizational tasks

Type of task	Description
Software delivery	<p>In the context of operational servicing tasks, software delivery covers the receipt of new software by the user from the distribution organization. Receipt implies a number of activities including both quality aspects and validation aspects:</p> <ul style="list-style-type: none"> – Checking condition and completeness of delivery – Checking configuration status information – Checking license information – Checking compatibility between delivered software and the system

Type of task	Description
Operational configuration control	This task takes into consideration the compatibility of software to special hardware. A specific software package may only work on certain variants of hardware items. On the other hand it can be possible to install/load different software packages to equipment, depending on the required functionality. The user must know what is compatible. Establishing configurations that are not tested or even not allowed must be avoided.

5.3.2

Management support

The management support tasks are tasks that cannot be considered as pure operational support. However, a modification of software is not part of these tasks either. This kind of support is located somewhere between the operational support and the real software modification tasks. The activities can be carried out across all levels of support and by users as well as by support organizations. Refer to [Table 7](#).

Table 7 Maintenance support tasks

Type of task	Description
Problem reporting	<p>This type of problem reporting covers more aspects than the recovery problem report. The implementation and upkeep of a management system for failure reporting and for corrective activities (eg a bug tracker system) is a central activity in this area. Within this system all actions are monitored and organized:</p> <ul style="list-style-type: none"> – Prioritization related to the severity of problems – Request for software modification – Monitoring of software modification and bug fixing procedures <p>An additional aspect can be the continuous monitoring of system effectiveness and the development of support costs. To guarantee best software performance and supportability, it can be necessary to change software because of these kinds of problems.</p>
System configuration control	<p>In addition to the operational configuration control, the system configuration control is of increasing importance. The interconnection of many computer systems in large networks within a complex Product requires configuration control at a higher level. The first step is that it must be ensured that no software package is installed on non-compatible hardware equipment. This is covered by operational configuration control.</p> <p>The second step is for system configuration control must ensure that all Product components that carry software/data and that are connected, work together properly without any problems. In modern Products this can become a complex challenge. Establishing a compatibility matrix for software/hardware is recommended. Only tested configurations of an assembly of hardware and software components can be operated under safe conditions.</p>

Type of task	Description
Delivery and installation	In the context of management support tasks, these activities are carried out by the contractor or by the supporting organization. In particular, after the modification of software and the creation of a new software release, the modifying organization will be concerned with the best way of providing the new software Product to the affected users, which can be a single client or in case of widespread Products, thousands of clients. Packaging and distribution paths will be of special interest in the case of sensitive or security relevant software.
User support	<p>User support covers a wide range of activities carried out by the supporting organization. The interactions between the supporter and the user can be:</p> <ul style="list-style-type: none"> – Support for installation, set-up or operation of software – User help desk/hotline – User queries, which are answered by the supporting organization by providing result reports – User training

5.3.3

Software modification tasks

These tasks are related to a longer-term process of implementing changes to software. Normally these activities are carried out by a supporting organization of the contractor at a high level of support (manufacturer or software vendor). However, modification of software can also be carried out by the customer, particularly changes of lower criticality. This requires corresponding equipment as well as the appropriate skilled personnel. Software modification tasks cover the software modification activity itself, including coding, change implementation, and testing as well as the related tasks, such as problem investigation and configuration control. Refer to [Table 8](#).

The reasons for the necessity to change software can be of different quality. These are:

- Corrective changes
- Adaptive changes
- Perfective changes

In general, the most critical reason for the necessity to change the software is the occurrence of real failures, which leads to an unacceptable situation within the operation of the entire Product. Such a failure will normally lead to a corrective modification of software. Also, the necessity of adaptive changes can be a critical situation, but this kind of change can normally be predicted and planned for in a better way than for a corrective activity due to a real software failure. The time schedule for adaptive changes is not normally as critical as that for a corrective action. However, in modern systems, it becomes more and more difficult to predict the adaption needs for software over a longer period.

Perfective changes are for increasing the user-friendly properties of the software or for increasing the functionality. Normally these changes are implemented through an upgrade concept. The requirements of the users will be collected during a certain time period and will be implemented within a new release of the software package.

Table 8 Software modification tasks

Type of task	Description
Problem investigation	<p>The initial concern of problem investigation is usually to determine how the affected software can be further operated with the actual failure (eg with lower capability). It must be ensured that a real software bug or a real technical problem exists. Triggering any real software modification because of, for example, user mishandling, must be avoided. After the problem has been correctly identified and any appropriate interim measures put in place, an analysis will follow on how the software can be modified that the problem can be resolved permanently.</p> <p>For a proper problem investigation, adequate resources and tools for failure finding and documentation must be available. This can be a simple programming environment with certain compilers and debuggers and/or a complex environment which allows simulation and failure reproduction.</p>
Change implementation	<p>The implementation of changes into software packages requires carefully considering some crucial aspects. Normally, the supporting organization hosts a complete repository of the entire software development process from the beginning to the actual status. Coding, compilation and testing of software requires special tools, which must be available. Additional appropriately trained and experienced personnel is necessary.</p> <p>Any basic change in technology can cause support problems and this must be taken into account. Examples can be obsolescence or the superseding of tools that are required for software modification or the change of basic operating systems on computers. This can have a large impact on costs or even require the upgrade of entire software packages to a new release when the former software package is not compatible with the new environment.</p> <p>In the SSC, the relevant standards and procedures that are used must be defined and documented.</p>
Change release	<p>A new release of a software package marks the end of its change implementation process. The new software package containing the installable and executable files/code, which is tested and certified, can be now delivered to the customer/user. All additional information is also available for the end user, such as installation instructions and modified documentation.</p> <p>The supporting organization must ensure that the distribution of modified software releases is well organized. This is part of the overall SSC.</p>
Configuration control	<p>Configuration control of software during modification includes managing individual change development and controlling different software versions and documentation. For this reason, the usage of a software development repository is strongly recommended. The compatibility between the new software packages and the existing hardware/software configuration must be ensured.</p> <p>In the SSC, the relevant standards and procedures must be defined and documented.</p>

6 Supportability factors

To establish a proper supportability of software, the analyst must consider a range of factors that can influence software supportability. All of them must be taken into account in terms of the project requirements. Some of the factors are more related to operational support and some of them are related to software modification support. These supportability factors are generally either attributes of the software itself or the associated development process, or of the environment within which the software is operated and supported. For special projects, there can be additional aspects which are not completely covered by these factors, but it is a good starting point for any analytical activity concerning logistic aspects of software. The factors are:

- | | |
|---------------------------------------|---------------------------------------|
| – Compatibility matrix | (refer to Para 6.1) |
| – Deployment | (refer to Para 6.2) |
| – Documentation | (refer to Para 6.3) |
| – Loading/unloading and installation | (refer to Para 6.4) |
| – Transportation on hardware carriers | (refer to Para 6.5) |
| – Expansion capability | (refer to Para 6.6) |
| – Modularity | (refer to Para 6.7) |
| – Modification frequency | (refer to Para 6.8) |
| – Recovery | (refer to Para 6.9) |
| – Safety integrity | (refer to Para 6.10) |
| – Security | (refer to Para 6.11) |
| – Size | (refer to Para 6.12) |
| – Skills | (refer to Para 6.13) |
| – Software distribution | (refer to Para 6.14) |
| – Standardization | (refer to Para 6.15) |
| – Technology | (refer to Para 6.16) |

6.1 Compatibility matrix

In systems where multiple software carrying equipment work together in a network, it is important to guarantee the compatibility of the equipment. It must be ensured that different versions of software on different hardware items are able to interoperate properly in a networked system. On the other hand, the compatibility of different versions of a software package with different hardware items is also of interest. The recommendations on where to document these compatibilities are:

- Software to hardware compatibility can be documented within the LSA database within the physical breakdown containing corresponding software items.
- A compatibility matrix concerning networked equipment carrying software packages, which must document the allowed configurations of the complete system concerning compatibility of software versions/issues is required. It is recommended to establish this matrix outside of an LSA/SSA database.

6.2 Deployment

Software deployment is a crucial aspect for software supportability. Each location, in which software packages are in use, must be sufficiently supported. The linkage of the locations to support capabilities (eg installation support, user helpdesk, troubleshooting and recovery support) must be addressed in detail and agreed by the customer and the support organization.

6.3 Documentation

In general, documentation refers to two main aspects:

- Software developer supporting documentation
- Software user supporting documentation
 - End user documentation (user handbook)

- Administrator documentation (installation/update and configuration handbook for administrative purposes)

In order to ensure software supportability for the developers, documentation must be produced conforming to an agreed standard and must be available to the organization responsible for delivering software support. Any software tools used in the creation of documentation must be included in the support facility and arrangements must be defined for their life cycle support.

For operational aspects, the documentation for the administrator or for any power user is of special interest. The documentation must contain installation and configuration manuals as well as troubleshooting tips in case of some malfunctions, which can be rectified easily. In case of more severe malfunctions, the documentation must contain the relevant data for the supporting organization that must be contacted.

6.4

Loading/unloading and installation

The duration of software loading is an important aspect. For this reason, the software itself and corresponding loading devices must be optimized for proper and fast loading processes. The same applies for the unloading of data or software after special missions or usages of a Product. If a large amount of data must be downloaded from a Product after a mission, a fast and easy download procedure must be available.

An additional aspect of software/data loading is safety. Appropriate test procedures must ensure that the loading or the installation was actually successful and the system works without any problems.

6.5

Transportation on hardware carriers

There are particular aspects that must be addressed if it is necessary to transport software using hardware devices. Which kind of carrier will be selected depends on the type, size and safety requirements of the software. Aspects can be, but not limited to:

- Is it possible to transport the software on chip cards or USB sticks?
- Do you need DVD or even hard disks as a carrier for the software?
- What are the safety aspects (eg is it allowed to transport the software on a carrier, where the files can be modified easily)?

6.6

Expansion capability

Expansion capability is an aspect of system design. Though related to the software and sometimes influenced by the software design, it is usually a hardware characteristic that must be considered within the overall system design. It deals with the degree to which software can be modified without being limited by restrictions on computing resources.

Examples of such constraints on computing resources are:

- Available memory
- Processor performance
- Mass storage capacity
- Input/output bandwidth
- Database capacity
- Network performance

If expansion capability is not taken into account in enough detail, software modification will be limited, modification costs will significantly increase and/or additional activities will be required. Expansion capability is highly important in systems with fixed hardware configurations or embedded real-time applications.

6.7

Modular software design

Modularity is an attribute of the structure of a software design. Different areas of functionality within a software package can be structured with the help of a number of modules. The

possibilities to apply modularity to a software design are determined by the choice of design method, programming tools and programming language. However, in general, the optimum approach to achieve modularity will be one that balances functional and performance requirements against the need to provide an understandable and supportable design. Insufficient modularity can result in increased modification effort and costs because of the need to implement consequential modifications in different parts of the software.

6.8

Modification frequency

Modification frequency (also called change traffic) is a measure of the rate at which software modification is necessary. Before the software is in use, estimates of the modification frequency can be made with the help of comparison with similar applications (any data available from comparable in-service systems on change traffic and effort will be of significant value). Some calculation methods based on the characteristic properties of the software can be used for change traffic estimation. Experience during test and trial phases can also be included in the investigations. Modification frequency influences stability, software integrity and system operation. Change traffic will affect the volume of software support activity. Higher change traffic will require more software modification activity.

High modification frequency involves a higher risk of downtimes of the entire system because of implementation problems or even because of bugs from new software releases. It is essential that a balance between the frequency of new software releases and the needs from adaptive, perfective or corrective requirements is found.

6.9

Recovery

In case of any system halt caused by a software failure, it must be possible to reset the system to full functionality or at least to an acceptable level of degraded functionality in an acceptable amount of time. This must be considered in early phases of software development in order to provide a proper instrumentation for recovery purposes. This can include, but not limited to:

- State of health monitoring
- Fault-tolerant handlers
- Design for debug features

This aspect is particularly important in systems deployed to space or other exposed locations. If software fails in such distant systems, it is crucial to have a potential recovery capability using the available instrumentation for operational real-time upgrade (eg download updated software and/or data).

Recovery procedures must be fully documented and must be well known by the responsible users or administrators. Recovery procedures must be tested before a real emergency occurs. In this case it must be clear what to do and how (eg critical data can be restored from a data backup). The best backup is worth nothing if during an emergency the stored data cannot be restored because of unattended circumstances. It is recommended that personnel are trained to react properly and that proper function of recovery be regularly simulated.

6.10

Safety integrity

The safety integrity required by a software package is determined by the safety criticality of the functions that it provides. The overall safety criticality of a system should be analyzed by the application of an appropriate hazard analysis technique. The partitioning of system functions in the system design is the consequence of the criticality of particular software items. Design must guarantee that software items that implement critical functions are minimized or at least segregated from other parts of the software. System requirements must define safety criticality categories and software safety integrity levels. Constraints and requirements for software design, testing and modification will be associated with each safety integrity level.

However, safety integrity is not only a matter for software design. During transportation, installation and operation of the software, some activities can be necessary to achieve a

sufficient safety level. For example, the granting of user access rights on the basis of allowed software functionalities or data areas is an important safety relevant activity, as well as the proper transportation of critical or secure data.

6.11 Security

The aspect of security primarily concerns the software itself. There are methods that can be applied to meet security requirements for software usage, such as::

- Cyclic redundancy check
- Encryption
- Digital signature
- Activation or license keys

The security classification of data, executable code and documentation can constrain software support activities. The main influence on equipment will be to impose special handling requirements. This can limit access to the software and introduce design requirements which raise the importance of specific software support tasks and equipment. The security classification of a software item will depend on the application and the equipment design. Wherever possible, software must be designed in a way that highly classified software parts are physically segregated from all other software within the system. Security requirements provide criteria for security classification of software items and specify modification and handling constraints associated with such classifications.

Another aspect of security is the operational character. It must be ensured that a software package can operate in a secure environment. This means internal security concerning a proper user access control policy as well as external security. Intruders from outside must be effectively blocked and internal users must be limited and controlled concerning their activities (eg via the internet). These operational security requirements can cause a number of support activities such as:

- Network administration activities (eg granting access rights to specific data areas, file servers or application services)
- Firewall administration activities
- Virus detection and defense activities

Note

Software in military systems can contain operational data that must be prevented from access by third parties. In this case it must be possible for the crew to erase all software and data to prevent further use of the Product and/or access to the data.

6.12 Size

The size of software packages influences supportability parameters, both in terms of the level of change traffic expected and the resources required to implement modifications. The size of the software within a system depends on the application and the design solution. Software requirements must state any constraints on the size of run-time software imposed by the system design. Many software support and supportability projections will be based on software size and complexity. Software development requirements should define aspects for data collection and analysis to measure software size and to verify models or estimates of supportability parameters, which are dependent on software size.

6.13 Skills

The required skill levels for personnel that are involved with software must be considered from several perspectives. The normal user must be able to handle the functions of the software in a professional manner. Operators with administrative responsibilities will require a deeper education concerning the functionality of the software in the corresponding IT environment. The highest level of skill is required for software modification. In this case, personnel with appropriate software engineering skills are needed. Requirements for particular qualification are

associated with the application domain and the technology or the methods used. Skill requirements will be determined by the system design, the software design and the applicable software support policy.

6.14 Software distribution

Software can be distributed on a variety of different media types. Distribution via the internet or other network environments, such as Local Area Network (LAN) or Wide Area Network (WAN) structures, is possible.

There are particular aspects that must be addressed if it is necessary to transport software using hardware devices, such as the storage medium and access restrictions to that medium.

6.15 Standardization

Standardization can be applied to the computing environment within which the software is operated. The same applies to technologies and engineering processes that are used to develop the software and the associated software documentation. Standardization helps to reduce the diversity of tools and methods. This considerably reduces the effort for training of personnel and for the required equipment at the support facilities. Also, for operational aspects, standardization can be helpful (eg loading/unloading or backup procedures can be defined to conform to existing standards).

6.16 Technology

Technology aspects must be considered with respect to the software engineering methods and tools used in development and implementation. This includes:

- Software design methods and supporting tools
- Operating systems
- Programming languages
- Compilers and debuggers
- Software test methods and test environments (hardware and software)
- Project specific tools and techniques

Technology also influences operational aspects. For example, the technology of storage devices within a system can significantly influence support activities or the use of network based technologies can dramatically ease the maintenance of client computers in a client-server application environment.

Chapter 14

Life cycle cost considerations

Table of contents

	Page
Life cycle cost considerations	1
References.....	2
1 General	2
1.1 Introduction	2
1.2 Objective.....	3
1.3 Scope.....	3
2 Different views on costs over the Product life cycle	3
2.1 Life Cycle Cost.....	3
2.2 Total Ownership Cost	3
2.3 Whole Life Cost	4
3 LCC analysis process	4
3.1 General approach.....	4
3.2 Aim and objective of the study.....	5
3.3 Develop the LCC framework	5
3.3.1 LCC breakdown structure	5
3.3.2 Data and assumptions definitions document.....	6
3.3.3 Methods, models and tools.....	6
3.3.4 Risk and uncertainties analysis	6
3.4 LCC analysis outputs and reports	6
4 LCC analysis in the LSA business process.....	6
4.1 Interactions	6
4.2 LCC aspects during early LSA activities	7
4.3 LCC analysis during LSA process	8
4.3.1 Influence on design	8
4.3.2 Configuration assessment.....	8
4.3.3 LORA	8
4.4 Data exchange between LSA and LCC	8

List of tables

1 References	2
2 List of potential LSA data elements to support LCC analysis	9

List of figures

1 Acquisition costs versus O&S costs (iceberg effect)	2
2 Relation between LCC, TOC and WLC	4
3 Life cycle cost framework	5
4 Interactions between LCC and LSA	7

References

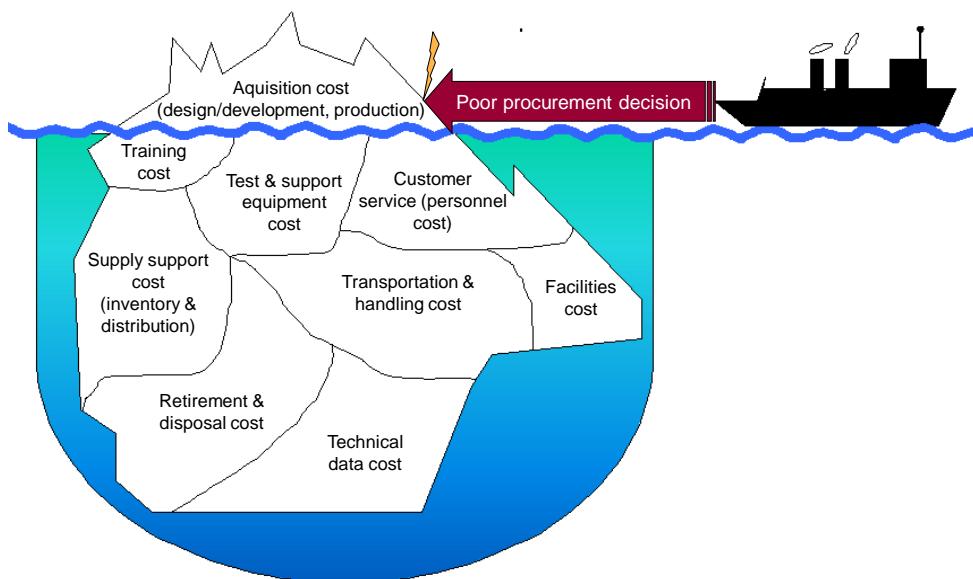
Table 1 References

Chap No./Document No.	Title
Chap 19	Data model
NATO RTO Technical Report 028	Cost Structure and Life Cycle Costs for Military Systems

1 General

1.1 Introduction

The operations and support costs usually exceed significantly all cost of acquisition and, therefore, the decision to purchase a Product should not only be influenced by the Product's initial cost (acquisition cost) but also by the Product's expected operating and support cost over its life and by the Product's disposal cost. The relation between acquisition cost and support related cost for buying decision is shown in [Fig 1](#).



ICN-B6865-S3000L0070-002-01

Fig 1 Acquisition costs versus O&S costs (iceberg effect)

LSA identifies the support system required to meet the customer's performance objectives in terms of Product usage (eg, availability, safety). In that context, LSA requirements support the minimum Life Cycle Cost (LCC) by meeting requirements in areas such as the number of maintenance levels, maintenance support personnel capabilities, maintenance training limitations, technical manuals limitations, reliability and Mean Time to Repair (MTTR) at each maintenance level. The LCC analysis process can be used to estimate the cost impact of LSA requirements. LCC analysis results can hence be used as one decision criteria in the LSA process for:

- Selecting LSA candidate items
- Identifying analysis activities for each candidate item
- Influence on design
- Configuration assessment
- Level Of Repair Analysis (LORA)

To the same extent, the LCC analysis process benefits from LSA as the related requirements the data generated are inputs for updating the LCC model. Therefore, it is vital to maintain a coherency between the LSA database and the Data and Assumptions Definitions Document (DADD) used for the LCC analysis. Refer to [Para 2.3](#).

1.2 Objective

This chapter provides an overview of the LCC aspects and which information developed by an LSA process can be used to support the LCC analysis activities.

1.3 Scope

The scope of this chapter is to provide LSA managers and analysts an understanding of Life Cycle Cost (LCC), its use and interactions with the LSA business process. The LSA managers and analysts need to share information regularly with LCC experts to ensure that their analysis and decisions take into consideration the economical factor.

2 Different views on costs over the Product life cycle

Several terms are used to define the cumulative costs of a Product and its support environment over its life cycle from design to disposal:

- Life Cycle Cost (LCC)
- Total Ownership Cost (TOC)
- Whole Life Cost (WLC)

These terms differ in their scopes (refer to NATO RTO Technical Report 028), which are described in [Para 2.1](#), [Para 2.2](#) and [Para 2.3](#).

For LCC analysis activities, a strong relationship to LSA activities and results can be identified, which is considered within [Para 4](#).

2.1 Life Cycle Cost

LCC consists of all the direct costs plus any indirect-variable costs associated with the procurement, operations, support and disposal of the Product. Indirect-variable costs can include linked costs, such as additional common support equipment, additional administrative personnel and non-linked costs, such as new recruiters to recruit additional personnel. LCC also includes the marginal costs (both direct and indirect) of introducing a new equipment or capability.

LCC does not include notional allocation of costs, whereas TOC and WLC can do. All indirect-variable costs related to activities or resources that are not affected by the introduction of the Product are not part of LCC.

Note

LCC is used as a minimum for the analysis of alternatives and to compare options of alternatives, and often for economic analyses.

2.2 Total Ownership Cost

TOC consists of all elements that are part of LCC plus the indirect, fixed and linked costs. These latter can include items such as common support equipment, common facilities, personnel required for unit command, administration, supervision, operations planning and control, fuel and munitions handling.

TOC represents all costs associated with the ownership of a system except non-linked fixed costs that are related to the running of the organization.

Note

TOC is used for budgeting purposes, determining the use of services between systems, for optimization purposes and for financial analysis.

2.3

Whole Life Cost

WLC consists of all elements that are part of TOC plus indirect, fixed, non-linked costs. These latter can include items such as family housing, medical services, ceremonial units, basic training, headquarters and staff, academies and recruiters.

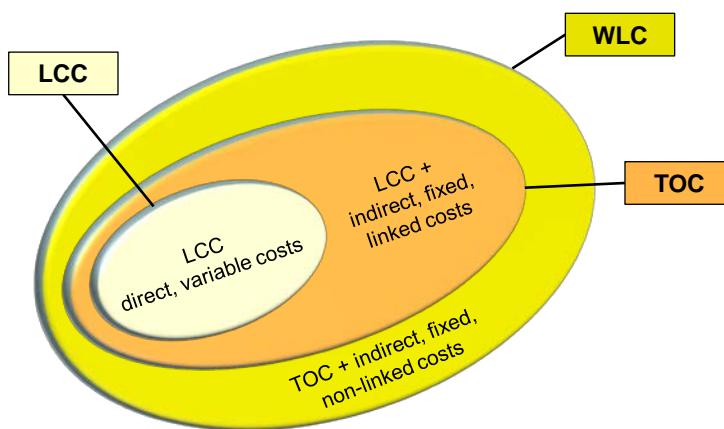
In WLC all costs or expenses that are made by the organization are attributed to the systems or Products they produce.

As WLC represents the total budget provision including such element as headquarters costs, it allows the visibility of the complete allocation of funds.

Note

WLC is used for a strategic view and high-level studies.

The relationship between the different cost classes described in [Para 2.1](#), [Para 2.2](#) and [Para 2.3](#) is shown in [Fig 2](#).



ICN-B6865-S3000L0071-002-01

Fig 2 Relation between LCC, TOC and WLC

3

LCC analysis process

3.1

General approach

Although it can be necessary to adapt the LCC analysis process to the Product being considered, there are common steps that can be essential in all the LCC analysis processes. The depth of the analysis for every step needs to be tailored to the requirements of each project.

The approach needs to be tailored to support the questions to be answered, the costing requirements and the availability of suitable data. With some variation (to the level of detail), the same basic approach to LCC analysis can be applied to all projects regardless of their specific requirements. This approach requires some steps to achieve a successful LCC analysis, which are:

- Definition of aims and objectives of the study. Refer to [Para 3.2](#).
- Development of the LCC framework. Refer to [Para 3.3](#).
 - Establishment of the program content, the costing boundary within the LCC breakdown structure
 - Identification of methods and models needed according to the objective and the life cycle phase
 - Identification, collection and documentation of data and assumptions necessary to populate the LCC model
 - Analysis of risks and uncertainties

- Report and presentation of the results. Refer to [Para 3.4](#).

3.2 Aim and objective of the study

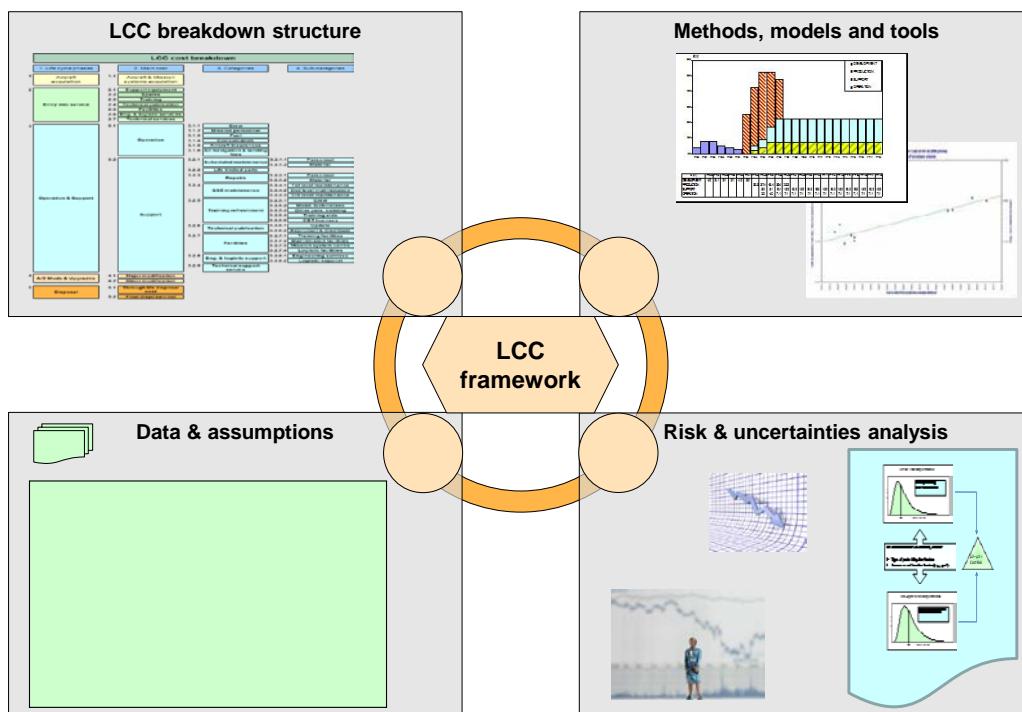
It is essential to define what is going to be estimated and understand what the estimates will be used for (eg, setting budgets, support options evaluation, trade off decisions, pricing). Examples for the range and variety of objectives can be:

- the acquisition of a new transport aircraft to fill a capability gap
- the acquisition of an information management system for a medium sized company
- the modification of a large fleet of vehicles due to obsolescence

The aim and the objectives of the study will have a major impact on the way an LCC analysis is conducted. A different type of question will result in a different way of conducting the study. This will often be implicit in the type of study being undertaken, but it needs to be clearly and unambiguously defined if the LCC analysis is to provide useful and meaningful results.

3.3 Develop the LCC framework

The different components of an LCC framework are shown in [Fig 3](#).



ICN-B6865-S3000L0072-002-01

Fig 3 Life cycle cost framework

3.3.1 LCC breakdown structure

The LCC Breakdown Structure (LCCBS) identifies all cost items affecting the total LCC of the Product. Cost items need to be defined in a very specific way to avoid ambiguity. For that purpose, a cost items needs to be defined according to 5 dimensions:

- Resources
- Activity
- Product
- Time
- Location

Example:

Cost of technicians (resources) to maintain (activity) the engine (Product) during 25 years (time) at company X (location).

3.3.2 Data and assumptions definitions document

The DADD records the LCC analysis ground rules and assumptions. This includes organizational, program, technical and economic data. The LSA can provide technical and organizational data regarding support. For this purpose, it is obvious that LSA data must be maintained within the in-service phase to enable LCC analysts to use the LSA as a dependable data source.

3.3.3 Methods, models and tools

There are many software packages on the market to support an LCC analysis, but it is recommended to select the most appropriate tools depending on:

- the specific Product to be analyzed
- the objectives of the LCC analysis
- the life cycle phase

As several methods can be used, it is useful to compile all results in an aggregation tool so that an overview on the LCC can be maintained. The use of tailored LCC models can be preferred in some cases.

3.3.4 Risk and uncertainties analysis

Risk and uncertainties analysis can be included within the model so that the possible variances in program estimates in terms of cost can be fully understood.

3.4 LCC analysis outputs and reports

The results of the LCC analysis can be documented and presented by as many reports and graphs are required to allow users to clearly understand both the outcomes and the implications of the analysis.

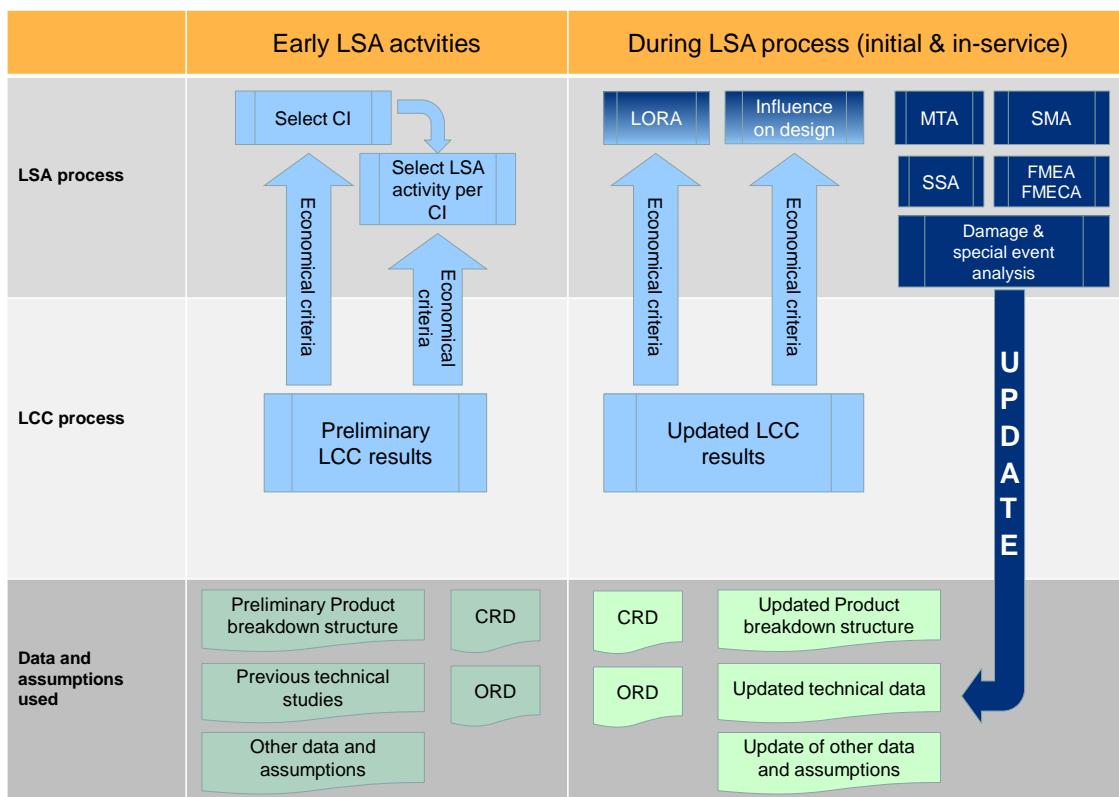
The final LCC analysis report must contain at least the purpose and scope, a summary of the LCC framework, results, analysis of the results including sensitivity and trade-off analysis and conclusions and recommendations.

4 LCC analysis in the LSA business process

4.1 Interactions

LCC analysis work starts in the early phases, before LSA begins. Therefore LCC results are already available once the LSA preparation starts within the early LSA activities. LCC results can serve as one criterion to set up the LSA program plan and framework and it can support the selection of LSA candidate items and the identification of analysis activities per LSA candidate. Refer to [Fig 4](#).

During the LSA process, many data will be generated from the different analysis activities like eg, FMEA, MTA, SMA, Damage and special events analysis or SSA. These are important inputs necessary for updating the DADD of the LCC framework, particularly with regard to resources. At the same time, LCC analysis can also be used as one criterion decision making during activities such as influence on design, configuration assessment and LORA.



ICN-B6865-S3000L0073-002-01

Fig 4 Interactions between LCC and LSA

4.2

LCC aspects during early LSA activities

Before LSA work starts, LCC analysis has been provided for different reasons, such as:

- Affordability analysis
- Business case
- Evaluation of alternatives

Therefore, some data and assumptions have already been captured and used for that purpose. These data and assumptions can include, but not be limited to:

- Organizational and program data, such as support and maintenance organization, operation and the quantity of the Product. These data are reflected in an Operational Requirements Document (ORD) and in a Customer Requirements Document (CRD).
- Technical data, such as a Product breakdown structure, technical data, maintenance and support characteristics of the breakdown elements based on existing analysis results (eg, weight, reliability and testability information, duration of maintenance, periodicity of preventive maintenance, scrap rate)
- Economic data (eg, man hour costs for different activities like design, development, production, operation or maintenance, costs of other resources)

At this early stage, LCC models are usually based on parametric or analogy methods with its range of uncertainties. However, the LCC model developed allows the identification of cost drivers based on the breakdown elements and the corresponding support activities. This identification can be used as one decision criterion during the LSA preparation for candidate item and analysis selections, such as:

- LCC analysis can be used to support the candidate items selection from the Product breakdown structure that are cost drivers and have many uncertainties. These can potentially be part of the candidate item list.
- For each candidate item selected, LCC analysis can be used to identify the activity driver (eg, if corrective maintenance is the activity driver, it can be useful to carry out FMEA and MTA in depth)

4.3 LCC analysis during LSA process

During the complete LSA process, LCC analysis can be updated with the results of different analysis activities. It is vital to maintain a coherency between LSA data and the DADD used for the LCC analysis.

LCC analysis is also used as one criterion for decision for some activities such as influence on design, configuration assessment and LORA.

4.3.1 Influence on design

For influence on design, the following steps can help in the decision making process. However, the use of these steps needs to be combined with other criteria, such as safety and availability. The steps are:

- 1 Use LCC model to identify breakdown element drivers (high LCC)
- 2 Perform LSA to decrease the support cost part of the LCC for these drivers and identify LSA requirements
- 3 Analyze whether these requirements need design change
- 4 Analyze what design changes are needed (with design office), collect data and assumptions for design, development, production and disposal
- 5 Update LCC analysis taking into consideration new design, including LSA requirements
- 6 Compare with initial LCC to quantify benefits

4.3.2 Configuration assessment

When a design change is needed for any other reason than above, LCC analysis is used as one criterion for decision making during the configuration assessment. Certain steps can help in the decision making process. However, the use of these steps needs to be combined with other criteria such as safety and availability. The steps are:

- 1 Identify the data and assumptions in the change request
- 2 Perform LSA if the change concerns Candidate items and identify LSA requirements
- 3 Update LCC analysis with all the data above

4.3.3 LORA

For LORA, LCC analysis is used to evaluate alternatives in terms of levels of repair (from an economical perspective). The part of the LCC to be considered in the evaluation is maintenance, spares provisioning, training maintainers, support equipment and facilities, non-recurring and recurring costs.

4.4 Data exchange between LSA and LCC

LCC analysis outputs are used as one criterion in making decisions during the LSA process. Conversely, the LSA process generates data that are useful for LCC analysis. [Table 2](#) gives an overview of the elements exchanged from LSA to LCC. The UoFs, classes and data elements are defined in [Chap 19](#). The number of data elements can be tailored to meet the requirements of the project.

Table 2 List of potential LSA data elements to support LCC analysis

UoF	Classes
Product and Project	ContractedProductVariant
Product Usage Context	MaintenanceLevel ContractedProductVariantAtOperatingLocationType
Breakdown Structure	BreakdownElementUsageInBreakdown BreakdownElementStructure BreakdownElement BreakdownElementRevisionRelationship
Part	HardwarePartAsDesigned SoftwarePartAsDesigned PartAsDesignedPartsList
LSA Candidate	ProductServiceLife ScheduledMaintenanceInterval MaintenanceFreeOperatingPeriod DownTime MaintenanceManHoursPerOperatingHour MeanTimeBetweenUnscheduledRemoval MeanTimeToRepair DirectMaintenanceCost FailureRate ReplacementTime FailuresPerOperatingHour ShopProcessingTime
LSA FMEA	LSAFailureMode LSAFailureModeWithDistributionRating LSAFailureModeWithDistributionRatio
Special Event And Damage	SpecialEvent SpecialEventEffect
Task	Task
Task Resources	TaskPersonnelResource TaskFacilityResource TaskMaterialResource AdditionalTrainingRequirement
Task Usage	TaskFrequency

Chapter 15

Obsolescence analysis

Table of contents

	Page
1 General	1
1.1 Introduction	1
1.2 Objective	2
1.3 Scope	3
2 Obsolescence strategy	3
2.1 Reactive obsolescence management	3
2.2 Proactive obsolescence management	3
2.3 Combination of reactive and proactive OM	4
2.4 Obsolescence monitoring	4
2.5 Strategy tools	5
3 Develop an obsolescence management plan	5
4 Implementation of obsolescence solutions	6
4.1 Budget for obsolescence management costs	6
4.2 Linking obsolescence to the risk management activity	7
4.3 Obsolescence and life cycle costs	7
4.4 Ensuring that industry plans and manages obsolescence	7
5 Summary	8

List of tables

1 References	1
2 Options for reactive obsolescence management	3
3 Options for proactive obsolescence management	3

List of figures

1 Life cycle phases	2
---------------------------	---

References

Table 1 References

Chap No./Document No.	Title
EN 62402:2007	Obsolescence management - Application guide

1 General 1.1 Introduction

Obsolescence occurs due to the length of time it takes to develop and field a Product and then the subsequent long life cycles of Products (Refer to [Fig 1](#)). Obsolescence affects all Products and systems and is not limited to hardware and components, but includes test and support equipment, software, tools, processes, logistic products, standards, specifications and expertise.

Early phases	Design & development	Production & implementation	In-service	Disposal
Up to 5 years for the concept phase	Up to 5 years for the design & development phase	Up to 25 years for the production phase	30 or more years for inservice phase	
Concept and technology development	System development and demonstration	Production and deployment	Operations and support	Disposal

ICN-B6865-S3000L0093-002-01

Fig 1 Life cycle phases

Obsolescence occurs for a number of reasons:

- The life spans of the components that make up the Product are decreasing, especially the life cycle of electronic components (presently approximately 5 years or less per innovation cycles)
- Obsolescence occurs because the manufacturing base, subcontractors and vendors, are subject to market forces. Manufacturers can go out of business and essential parts or sub assemblies can become unavailable.
- The loss of design and technical knowhow can have a big impact on the supportability of long life cycle Products particularly if bespoke software and components (eg, ASICS) were used in the original design
- Increasing environmental legislation regarding the use a specific chemicals (eg, MEK) or materials (eg, beryllium, copper) has also increased the pace of obsolescence as it restricts the use of materials and it is recommended that this be considered from the earliest phases of a project.
- The costs of scarce materials, production facilities and support services (eg, software maintenance) can also make the support of a Product unaffordable.

The rate of technological innovation coupled with long utilization of Products and services mean that it is almost inevitable that obsolescence will have an impact at some time in the life cycle. Despite this inevitability, obsolescence can be managed and mitigated. The cost of mitigation however increases significantly as the system, Product or service moves closer to becoming obsolete. It is recommended, therefore that Obsolescence Management (OM) be undertaken as early as possible and as an integral part of the design, development, production and in-service support phases in order to minimize potential remedial expenditure and thereby the overall Life Cycle Cost (LCC).

1.2

Objective

The relationship between LSA and obsolescence is that both processes occur repetitively in the Product life cycle and obsolescence must be an integral part of the LSA process as it impacts supportability, operability and LCC and therefore must be closely managed until the disposal phase. The objective of this chapter is to provide a description of obsolescence analysis and the implementation of the associated activities. It offers guidance on obsolescence planning to mitigate the risk of obsolescence and minimize the LCC of long life cycle Products.

Note

The information in this chapter is intended as an introduction and overview of the issues related to OM. Authoritative guidance can be found in the European standard EN 62402:2007 Obsolescence management - Application guide.

1.3

Scope

This chapter is aimed at engineering, quality, logistic, supply chain and sustainment organizations. It provides guidance on:

- Defining an obsolescence strategy
- Implementation of obsolescence solutions
- Develop an OM plan
- Linking obsolescence to the risk management activity
- Obsolescence and LCC
- Ensuring OM costs are budgeted for
- Ensuring industry plans and manages obsolescence

2

Obsolescence strategy

There are two basic strategies for managing obsolescence, reactive and proactive and these are outlined below.

2.1

Reactive obsolescence management

Reactive OM implies that there is either no specific provision for the task of managing obsolescence or for the resolution of the obsolescence issues when they occur. Each occurrence is dealt with in isolation, unless they occur concurrently, and there is no link into the risk management activity. Resolution is funded from within the budget allocated to other activities or additional funds are sought. If a reactive approach is being taken, management options are listed in [Table 2](#):

Table 2 Options for reactive obsolescence management

Option	Description
Do nothing	This is a valid choice. Your decision to have a reactive strategy can be based on the extended life and very low consumption of the item. This obviates the need for additional procurement activity.
Re-design	Usually an expensive option but it can be unavoidable. Costs can be mitigated by limited design transparency.
Alternative part procurement	This activity usually requires a parts search, which can include the removal of serviceable parts from an unserviceable Product for use in the repair of another unserviceable Product to make serviceable, known as cannibalization.

2.2

Proactive obsolescence management

Proactive OM implies that there is specific resource provision and a plan generated for managing obsolescence issues. It is linked into the risk management activity and has a budget allocated. It also has a stakeholder group identified for addressing issues as they arise. Product monitoring creates time to review resolution options and justify decisions. It is recommended that the proactive strategy is considered initially for all programmers. If a proactive approach to OM is being taken your management options are extended by the options listed in [Table 3](#):

Table 3 Options for proactive obsolescence management

Option	Description
Do nothing	As for reactive obsolescence management

Option	Description
Life time or end of life buy	A valid approach which can offer the best value. However care must be taken to consider the possibility of equipment life extension, changes in consumption rates and costs of ownership (eg storage, handling, exercising).
Planned upgrades	These offer good opportunities to address multiple issues at once, but would usually need to be coupled to obsolescence monitoring activities whether the upgrade is requirement or obsolescence driven. Care would also need to be taken to ensure sufficient spares are available between upgrades and dates and budgets strictly controlled.
Obsolescence monitoring	A key component of any OM strategy which if implemented correctly will at least ensure that enough time to plan and consider the solution is available. Once the obsolescence strategy is established, the preferred solution can be selected from the options available.
Design out obsolescence	This approach needs to be implemented in the earliest phases of a project. The intention is to design the equipment so that a subassembly capability is not tied to a particular solution.

A combination of these two approaches can be applied, too. This can be particularly appropriate where very large numbers of components are concerned, some of which can be identified as low risk. This option reduces the management cost of the OM task and also focuses on the high-risk items. Though there is increased risk attached to this approach which must be considered against cost and impact considerations. Care must be taken if considering reactive OM in the early phases of a project as it cannot be possible to recover important information necessary to support proactive OM later in the project.

2.3

Combination of reactive and proactive OM

Projects sometimes determine that an OM strategy using a combination of reactive and proactive OM is the most suitable approach.

This can be particularly appropriate where very large numbers of components are concerned, some of which can be identified as low risk. This option reduces the management cost of the OM task and also focuses on the high-risk items. However, there is increased risk attached with this approach. Proactive OM can carry costs which can increase with the number of components considered and monitored. These costs must be weighed against the possible impact on availability and obsolescence rectification costs. The cost of OM is a driver in the choice of the strategy.

Care must be taken when considering using reactive OM in the early phases of a project as it can become impossible to recover important information necessary to support proactive OM later in the project.

2.4

Obsolescence monitoring

When implementing the obsolescence monitoring solution, there are various options available. The use of the Design Authority (DA) to manage obsolescence is an obvious choice. The DA is best placed to develop a list of parts, identify solutions and modify the design. This is particularly the case during the early phases of the project when the design can be influenced by Obsolescence analysis. An alternative solution, particularly in the in service phase, is the use

of a third party contractor with a dedicated OM portfolio of skills. There are many companies which offer an OM service. The depth of service can be tailored to customer requirements. The advantages of this are that they have specialized skill sets and experience in the implementation of OM. Any overhead can be spread over many other projects and therefore offer better value. They do not have a vested interest in identifying obsolescence issues unless they are also contracted to identify solutions. A further option would be to employ a combination of the DA and third party. The DA can be employed to do the bulk of the work whilst the third party can offer specialist skills and, because its independence, be used to verify DA obsolescence claims, as the DA can have a vested interest.

2.5

Strategy tools

The impact, probability and cost are significant discriminators in the decision making process of the OM strategy. The data supporting this process can initially be based solely on the judgment and experience of the team but this should improve as the project develops.

The process considers:

- The impact of an event on the capability of the system. This must be considered with the customer or user and includes consideration of any reliability, maintainability and availability aspects.
- The probability of the event occurring, which include possible environmental and safety legislation aspects.
- The cost of the occurrence of an obsolescence event. This includes the total Product life cycle cost addressing the design, production and implementation elements of the immediate system and any rework or support activity costs (cost of ownership).

These three elements are considered together to determine the classification of the obsolescence event. This classification can be used to determine the obsolescence strategy, the level of monitoring, whether further investigation is required and the periodicity of review.

3

Develop an obsolescence management plan

The first step in managing obsolescence is to assemble a comprehensive OM plan. This document will implement the strategy that will be applied during the life of the Product or project. It is recommended that this be developed at the earliest phase of the project and reviewed at least at the end of each phase for applicability. It often starts as quite a small document but will develop and be refined as the lessons and decisions made during the design and development phases are incorporated. As the project progresses, the plan will develop into a through life record of the strategies employed, options considered and decisions made. It can incorporate the contractor's OM plan, which will be provided in response to the Invitation to Tender and contract.

The OM plan must:

- achieve the optimum compromise between whole life costs, performance, availability and maintainability for the entire Product
- cover all materiel, including hardware, software, tools, test equipment, human resources and training
- be compatible with the customer's current support arrangements
- provide a clear basis on which obsolescence management requirements can be negotiated with suppliers and partners
- take into account the need for component or equipment requalification/re-certification following component or module substitution
- identify the linkages and interfaces into the risk and life cycle cost program activities

The scope and content of the OM plan are the responsibility of the plan and its contents are the responsibility of the OM project team and must reflect the scope, complexity and cost of the project. There are minimum requirements of the OM plan. It must:

- identify obsolescence issues in the Product
- identify which obsolescence standards will be used
- identify obsolescence performance metrics
- record the choice of strategy and provide detail of any plans and OM decisions
- for decisions made it must document the reasons for which options were considered, analyses carried out and the trade-off decisions made for later reference. It is recommended that subsidiary documentation contains a full record of the details
- record the details of the stakeholder community personnel with authority for decision making on trade-offs between cost and impact. OM requires input from the customer, OEM, subcontractor and, in some cases, suppliers. A team with the correct stakeholders must be assembled. The teams composition will vary with the Product and potentially the life cycle of the Product. The team must establish a good working relationship so that information and decisions can be quickly determined. The longer it takes to implement a solution reduces the chances of its success.
- establish a schedule and strategy to refresh Product components and allow capability upgrades including enabling future Product support. The strategy will depend on the Product and can be broken down into electrical/electronic parts, systems or subsystems and non-electrical/electronic parts systems or subsystems. This is also the case where the frequency of scanning the Product breakdown is set. The more often the Product breakdown is scanned the less likely that an obsolescence alert will become a serious issue. However, the more often the scan is performed, the more expensive the process will become.
- identify a tool set which will be used to predict the life cycle of electronic components, enable a review of bill of materials and predict when an electronic component will no longer be supported. The tool must have the ability to read all levels of a Product breakdown and the ability to predict when a component or assembly can become obsolete.
- effectively manage obsolescence issues and maintain a record of discontinuance notices received from suppliers. Discontinuance notices alert customers that production is concluding for a specific part. The notices usually contain part numbers, last order and shipment dates, and minimum order quantities. In OM, it is essential that discontinuance notices are collected proactively.
- establish a secure data repository for obsolescence data. Establishing a secure area where information can be exchanged and stored is essential. This protects the intellectual property of all parties. This data can be used again for similar Products in the same stage of their life cycle.

4

Implementation of obsolescence solutions

When there is an obsolescence occurrence the choice of solution is wholly dependent on the project and market circumstances at that time. Therefore, each occurrence must be treated as an individual event that, at a minimum, must consider:

- reclamation
- last time buy
- life time buy
- re-manufacture
- use of an interchangeable item
- adaptive solutions
- reverse engineering
- new design

4.1

Budget for obsolescence management costs

It is essential that budgetary provision is made for the OM task itself. OM carries its own resource cost and this must be included in the project budget. In addition costs must be included for the resolution of the obsolescence issues as they arise. These issues will vary as the project develops and must be reviewed regularly. This cost forecasting effort must be linked

to the project risk activity to ensure that a consolidated approach is taken and that items are not accounted for twice or not at all.

4.2

Linking obsolescence to the risk management activity

Managers must consider the operational risks over the life of the equipment associated with obsolescence:

- What would be the impact of the Product being unavailable due to lack of spares or of performance degradation due to substituted parts?
- What would be the likely cost of premature replacement or of other measures to circumvent obsolescence?
- What is the probability of obsolescence occurring (including considerations of technology advances and the introduction of new legislation)?

Obsolescence is a key contributor to any risk assessment. It is recommended that the strategy for addressing obsolescence in risk processes be established early on in the project and reflected in the appropriate management plans. Having good visibility of the obsolescence risks is the first essential step required to minimize negative effects. This will enable the project management to identify and implement the most appropriate action to deal with issues in the most cost-effective way. It is essential to acknowledge that it is always the Product user who is most affected during in-service phase in terms of readiness, supportability and life cycle costs. Obsolescence is always a risk but it can create opportunities when alternative parts offer improved performance.

4.3

Obsolescence and life cycle costs

Obsolescence can lead to major changes in design, lead to changes in support philosophy and impact support arrangements. Because obsolescence can impact all phases of a project any review of solution options must consider the implications for the life of the Product. Hence the need for OM to be linked to the LCC activities. A whole life approach must be considered when addressing cost issues which must be reflected in the appropriate management plans.

4.4

Ensuring that industry plans and manages obsolescence

Industry must plan for obsolescence from the beginning of the design process including “obsolescence prevention” as another objective of the supportability engineering and logistics activities, maintaining this objective in all the phases of the project. Industry must:

- produce an OM plan
- conduct obsolescence critical analysis in the project to identify the Product items with the highest risk of obsolescence and, in accordance with a predefined criteria, use similar methodologies for the safety hazard and mission critical analyses
- adopt a pragmatic approach by putting into practice, as a continuous effort with all levels of industry participants, timely and cost-effective obsolescence engineering practices, management tools, methods and practices to avoid the negative effects of obsolescence in the project
- prepare obsolescence risk mitigation strategies for incorporation in the Statement of Work (SOW) for all provisioning contracts during the Product life cycle
- detail OM data requirements clearly within the SOW and ensure rights of access are addressed during commercial negotiations
- determine the obsolescence status of parts before their procurement
- ensure that all stakeholders plan the most cost effective time scales for in service technology updates/upgrades or technology road maps and managing obsolescence in existing equipment to extend its service life in line with the logistics and necessary qualification/certification processes
- apply practices already established in other projects for obsolescence mitigation whenever it is deemed convenient and cost-effective

It is essential that the Invitation to Tender, for all phases of a project, includes the requirement for an OM plan. The format and contents of the plan can be the contractor's best practice but, at a minimum, must conform to the requirements of a recognized standard. Intellectual property rights must be considered with regard to OM and in particular to ensure access to data required is provided to support the OM strategy. The requirements of OM must also be considered in the context of an exit strategy, to ensure that all the data, necessary to manage obsolescence, is available in the event of a contract termination.

One commercial strategy is to pass responsibility for the management of obsolescence to the contractor completely. This hands-off solution can, in its simplest form, consist of a fixed price payment for the resolution of any future obsolescence arising. This is not an optimal technical solution but rather a more commercial or contractual approach, passing all liability for future obsolescence issues to industry during a specified time period. In some cases, this can be accompanied by a redistribution of payment plan by bringing forward some payments in order to cover potential obsolescence issues arising during a specified time period. However, there are drawbacks to this approach. Although "risk is being transferred to the contractor" for a fee, care must be taken to ensure that, at a minimum, there is limited visibility from the customer side so that he can see where action is being deferred or risk carried. These decisions can impact the customer on contract closure. Alternatively, it is recommended that provision be made to address obsolescence risk as the contract approaches completion, if this is achievable. In addition, during the fixed price negotiation, it can be difficult to justify or investigate costs unless the contractor proposal is supported by a detailed risk based obsolescence cost analysis.

5

Summary

There are many causes of obsolescence. The loss or impending loss of manufacturers or suppliers of critical items, raw materials, intellectual resources, exponentially rising costs of scarce parts and materials, rapid changes in technology, uneconomical production requirements, competition, environmental or safety legislation, all impact the sustainability of a Product during its life cycle. OM is an activity intended to minimize the impact of this potential loss of supply through the identification, quantification and resolution of obsolescence and aims to achieve optimum cost-effectiveness. OM focuses on the system-wide application of risk management and is applicable to the entire Product life cycle, becoming an integral part of the design, development, production and in-service support phases of the project.

Chapter 16

In-service LSA

Table of contents

	Page
In-service LSA.....	1
References.....	1
1 General	2
1.1 Introduction	2
1.2 Objective	2
1.3 Scope.....	3
2 Core principles.....	4
2.1 Overview	4
2.2 LSA triggered by data analysis in-service	4
2.3 In-service LSA for modification and change implementation	5
2.4 In-service configuration management	5
3 Analysis procedure and sub-process description	6
3.1 In-service performance monitoring process	6
3.2 In-service data feedback to design for Product update	6
3.3 In-service updates of technical and logistic analysis results	7
3.3.1 Reassessment and update of the LSA FMEA.....	7
3.3.2 Reassessment and update of damage and special event analysis	7
3.3.3 Reassessment and update of the LROA.....	7
3.3.4 Reassessment and update of the SMA	7
3.3.5 Reassessment and update of the LORA.....	8
3.3.6 Reassessment and update of the MTA	8
3.3.7 Reassessment and update of the SSA	9
3.3.8 Reassessment and update of LCC analysis	9
3.3.9 Reassessment and update of obsolescence analysis	9
3.3.10 Reassessment and update of disposal analysis	10
3.3.11 Product defect and warranty management	10

List of tables

1 References	1
-------------------------	---

List of figures

1 In-service ILS process	3
2 Overall process for monitoring in-service performance.....	6

References

Table 1 References

Chap No./Document No.	Title
Chap 5	Influence on design
Chap 7	Results of FMEA/FMECA in LSA

Chap No./Document No.	Title
<u>Chap 8</u>	Damage and special events analysis
<u>Chap 9</u>	Logistics related operations analysis
<u>Chap 10</u>	Scheduled maintenance analysis
<u>Chap 12</u>	Maintenance task analysis
<u>Chap 13</u>	Software support analysis
<u>Chap 15</u>	Obsolescence analysis
<u>Chap 17</u>	Disposal
S4000P	International specification for developing and continuously improving preventive maintenance
S5000F	International specification for operational and maintenance data feedback

1 General

1.1 Introduction

Using the LSA methodology during the early phases of the Product life cycle, the support solution will have been developed ensuring that it meets the capability and operational requirements, is optimized for through life cost, and is sustainable. This chapter provides a process to monitor, evaluate and adapt a support solution during the in-service period of Product life cycle

In-service LSA is part of the Product life cycle activities. After the Product enters into service, relevant in-service data must be collected and subject to an analysis process against defined support requirements. Such support requirements can change several times during Product life cycle and can also be the subject of upcoming problems or shortcomings. The LSA methodology must be updated accordingly to ensure a continuous improvement of the support solution for any Product.

During the in-service phase, a responsible ILS manager must identify and trigger the necessary LSA activities in support of the ILS process. These activities must ensure that the support solution:

- is delivered to satisfy any support contracts
- continues to meet the capability requirements
- is periodically reviewed to ensure it remains optimal, taking into account any changing capability, or operational and environmental requirements
- is routinely reviewed to ensure it meets all applicable legislative requirements

1.2 Objective

The objective of this chapter is to provide a methodology for defining the necessary in-service LSA activities. Prior to a Product's entry into service all resources required to support the Product must be defined. Reviews are required to monitor the technical applicability and the cost-effectiveness of the actual usage of support resources. In case of technical or economic insufficiencies based on in-service data, potential modifications of the support system must be identified, recommended and implemented. The positive effects of in-service LSA activities include, but are not limited to:

- enhancing Product availability
- harmonizing and improving required resources to support the Product

- improving the Product itself by incorporating modifications
- reducing the logistic footprint
- reducing the logistic support cost

Implementing LSA activities during the in-service phase is a win-win approach for both, customer and supplier, because:

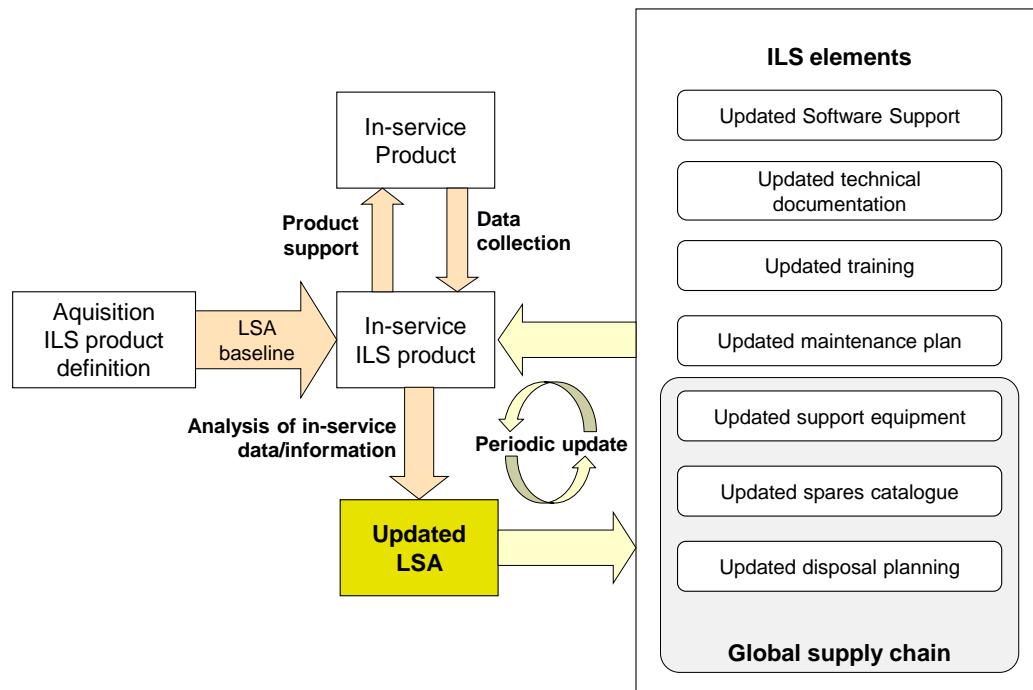
- the supplier will be better placed to know about the actual Product usage, and to anticipate support Product issues, enhancing their ability for effective support issues resolution
- the user will gain benefit from continuous improvement brought by the supplier on the Product supportability and on the support system efficiency

1.3 Scope

This chapter is applicable to the Products that enter into service and have already LSA performed prior to the Products' entry into service. For those Products that have not had LSA performed in accordance with S3000L, the requirements to conduct in-service LSA activities must be determined. The depth and extent of the LSA activities will depend on the scale of changes. These changes can be driven by:

- Product modification and mid-life updates, sometimes referred to as 'technology insertion'. This option involves predetermining points during the Product life at which the design of all or parts of the system are brought up to date and obsolete items replaced.
- Operational or environmental requirement changes or changes to the operational and environmental conditions must be assessed to identify the impact on the delivered support. The analysis can result in changes to the Product or the support solution.
- Legislation changes are to be reviewed, and if relevant, any impact on the delivered support solution must be assessed

The top level in-service ILS process is presented at [Fig.1](#).



ICN-B6865-S3000L0109-001-01

Fig 1 In-service ILS process

2 Core principles

2.1 Overview

During the in-service phase LSA aims at reviewing when necessary the Product support resources in order to ensure compliance with customers' technical, logistic and economical requirements. This review can be triggered by:

- Discrepancy between achieved and predicted results
- Modification or change to the Product required by the customer or to comply with external rules and legislation
- Search for areas of optimization taking into consideration the usage and support feedback as part of the overall process of support risk and opportunity management

As several LSA iterations can be conducted throughout the in-service phase, it is vital to put in place a Configuration Management (CM) process of the results of the LSA activities. This includes a thorough configuration control, accounting and audit. In addition, the in-service LSA relies on an operation and maintenance feedback system that needs to be in place at the Product entry into service.

2.2 LSA triggered by data analysis in-service

An analysis must be conducted of the available standard reporting systems to determine the amount and accuracy of supportability information that will be obtained on the new Product in its operational environment.

Any shortfalls in measuring achievement against the supportability goals that were established for the new Product must be identified. Supportability factors which were not tested during the procurement phases of the item's life cycle must be verified. It must be ensured to obtain required supportability data from the field which will not be obtained through standard reporting systems.

As an example, a Data Reporting and Corrective Action System (DRACAS) can be implemented. DRACAS is a documented closed loop system for reporting, collecting, recording, analyzing, categorizing, investigating and taking timely effective corrective action on all discrepancies and failures relating to design, manufacturing and test processes. DRACAS can be viewed as a component function in the greater scheme of in service ILSLSA data. Even if a dedicated DRACAS system is used and is relatively stand-alone, the data collected can become part of the overall data set used for ILS. Suitable opportunities for reporting can be integrated into the usual practice of maintenance. Facilities for reporting problems can be part of the following systems:

- Maintenance scheduling and maintenance recording systems
- Interactive electronic technical manuals

Electronic Health and Usage Monitoring Systems (HUMS) and Data Loggers (DL) can be encouraged wherever possible to remove the possibility of human induced errors.

Note

A DRACAS strategy can have an impact on the Product design and on DRACAS/HUMS/DL technical requirement specification and, finally, also on the selection process for an adequate Product including a proper support system.

Data are collected by the maintenance personnel discovering or rectifying the problem. Multiple reviews/approvals must be avoided and must be traceable to the reporting person. The detailed approach used for data collection must be as human friendly as possible. Where it is integrated into other systems, the data recording elements are usually aware of the context and avoid data re-entry. For example, when reporting a new problem encountered during a maintenance task, the option to report the problem must provide as much information as possible.

Additionally, the in-service data collection process will include but not be limited to the:

- type of data
- actual date and update rate
- exchange media
- confidentiality/intellectual property rights

As a future option, S5000F will provide a specification to identify suitable feedback data/information. A task team will be implemented with specialists from S3000L and S5000F to take care about data exchange issues. Main purpose will be to find a proper solution to develop proper information out of data which is collected within any data collection processes during the in-service phase.

The ILS manager will ensure that a trade-off analysis between cost, period of time that data is collected and number of operational units in which to collect data and statistical accuracy is conducted, to identify the best data collection plan. To support the provision of the capability through the in-service phase, reliability and maintainability is maintained by monitoring and recording:

- Product usage
- preventive and corrective maintenance activities
- consumption of spares and consumables
- periods of unavailability to monitor the reliability requirements

This enables the determination of the in-service achievement of reliability and maintainability requirements and the identification of shortfalls in the support solution. Analyzing these data will help understand the cause of shortfalls and, where appropriate, develop solutions and modifications for implementation.

2.3

In-service LSA for modification and change implementation

The Supportability Analysis (SA) activities that are undertaken during a Product modification or redesign process must be tailored to those that will return a cost effective benefit.

The SA activities that were specified eg in a Supportability Analysis Plan used for the procurement of the Product can be reused as a baseline for the identification of the required activities to be performed for modification or redesign and can assist in identifying and optimizing the support requirements.

The activities to be tailored out of the baseline will be those applicable to initial design that are primarily activities no longer possible. For example, basic design decisions having been made during the development process cannot be influenced by SA activities during in-service phase. However, in order to ensure that supportability aspects are addressed adequately, the applicability of the SA activities to the modification process must be identified in the modification instruction and harmonized with the CM system.

2.4

In-service configuration management

CM provides a mechanism for controlling Products' functional and physical characteristics throughout the acquisition life cycle, and enables an orderly transition from development thru production and entry into service. During the in-service phase consideration must be given to:

- configuration management during the LSA update process
- functional configuration change and physical configuration change identification
- configuration control during the LSA update process
- configuration verification audit during the LSA update process

Existing procedures will be adopted or new procedures will be implemented to apply CM as the control of Products' form, fit and functional characteristics, as described in the Products' technical documentation. CM provides a record of changes throughout the life cycle of the Product, and shows any dependencies between Products and their breakdowns. The record of changes against the baseline must be made available to both supplier and customer.

3

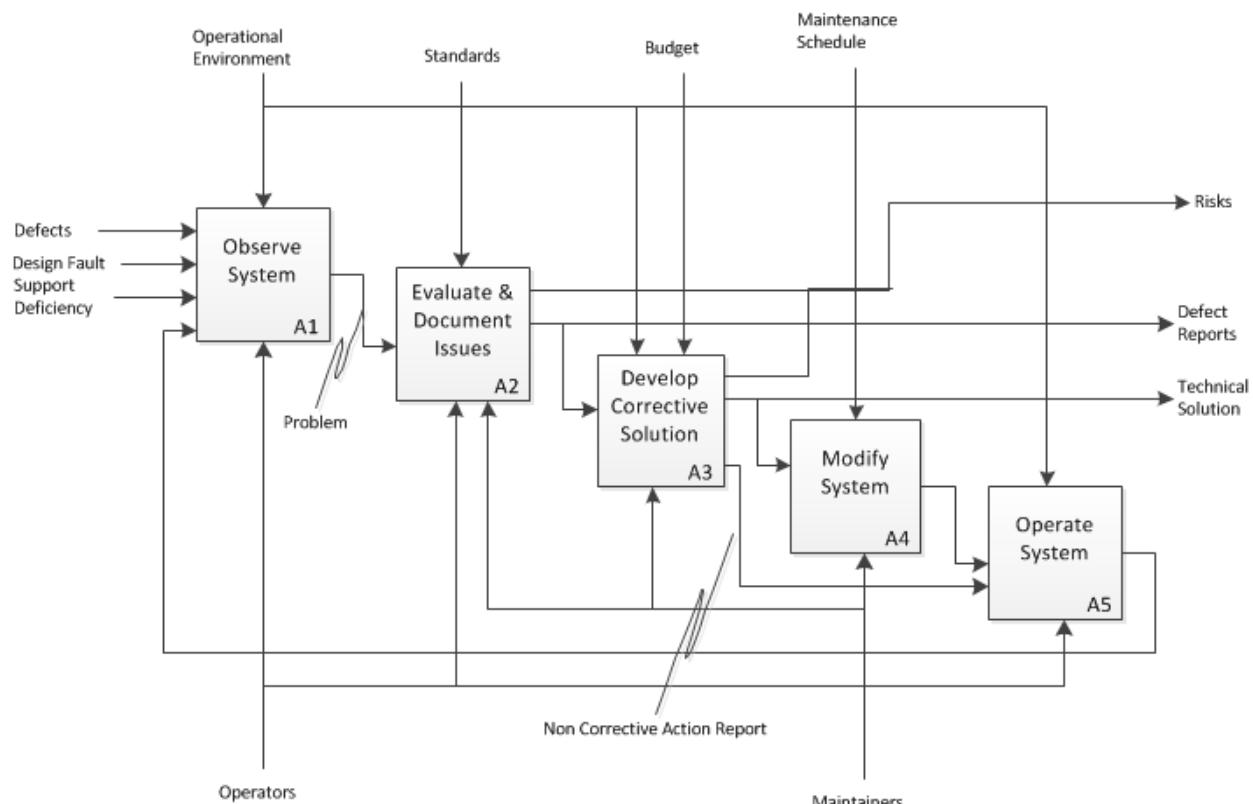
Analysis procedure and sub-process description

3.1 In-service performance monitoring process

[Fig 2](#) describes an overall process for monitoring in-service performance. Depending on the information collected and the customer's requirements, in-service LSA activities can include but not be limited to:

- results of FMEA/FMECA in the LSA, refer to [Chap 7](#)
 - results of damage and special events analysis, refer to [Chap 8](#)
 - results of Logistic Related Operations Analysis (LROA), refer to [Chap 9](#)
 - results of Scheduled Maintenance Analysis (SMA), refer to [Chap 10](#)
 - results of Maintenance Task Analysis (MTA), refer to [Chap 12](#)
 - results of Software Support Analysis (SSA), refer to [Chap 13](#)
 - results of obsolescence analysis , refer to [Chap 15](#)
 - results of disposal analysis, refer to [Chap 17](#)

In addition to ensuring the definition of optimized resources, the objective of "influence on design" is also a key factor in selecting the LSA activities, particularly in case of design changes. Refer to [Chap 5](#).



ICN-B6865-S3000L0094-001-01

Fig 2 Overall process for monitoring in-service performance

3.2

In-service data feedback to design for Product update

During in-service LSA, it is not possible to isolate the objective of influence on design from the update of any analysis results from for example, SMA, LORA, MTA or Damage and Special Event Analysis, because the actual status of the reliability data, maintainability data, testability data and others which can influence the design are clarified within the related activities in the support engineering activities reliability, maintainability, testability and safety.

3.3

In-service updates of technical and logistic analysis results

3.3.1

Reassessment and update of the LSA FMEA

The recommended analysis activities to determine a reassessment of the results of the FMEA/FMECA within the LSA are:

- To analyze logistic support products and their related in-service data which have the potential to influence the LSA FMEA result
- To analyze the in-service data, compare it with the original data in as designed LSA FMEA, or the baseline defined in a previous analysis and identify any differences
- If conflicts are identified between in-service data and original data, continue the analysis to judge whether the corrective tasks must be redefined
- Use the in-service data to verify the failure detection methods and detection rate, failure isolation methods and no failure found rate
- Redefine the requirement for troubleshooting for some items if necessary
- Update the corrective maintenance tasks list and the troubleshooting tasks list and transfer the revised tasks to MTA for further analysis

3.3.2

Reassessment and update of damage and special event analysis

The recommended analysis activities of the results of damage and special event analysis within LSA, include but are not limited to:

- Assessing actual usage data to determine if any special events have occurred that were not identified in the original analysis
- Assessing if new special events are likely to occur again and if yes, supplement them to the special event list and subject them to detailed analysis
- Using in-service data where possible, to assess if the damage sensitivity rating for new technologies is reasonable, and reanalyze if necessary
- Assessing whether all the damage factors and damage forms have been given adequate consideration
- Updating the corrective maintenance tasks list and troubleshooting tasks list and transfer the tasks to MTA for further analysis

3.3.3

Reassessment and update of the LROA

The recommended analysis activities of the results of the LROA within LSA, include but are not limited to:

- Using in-service data to assess whether all the logistic related operation tasks have been adequately analyzed or not
- Detailing any additional Logistic Related Operation Analysis activities required
- Assessing whether the ILS element of packing, handling, storage, and transport, any special or additional requirement must be added to existing tasks
- Updating the tasks list about logistic related operation and transfer the revised tasks to MTA for further analysis

3.3.4

Reassessment and update of the SMA

Effective and achievable maintenance schedules are essential to meeting the required availability and reliability, at an affordable through life cost. An initial scheduled maintenance plan is generated from an SMA, based on for example the principles of S4000P. These will need to be verified against actual performance of the Product in operational use. It is essential, therefore, that the scheduled maintenance plans remain a living document to reflect operating experience and changed usage.

The recommended analysis activities of the results of the SMA within LSA, include but are not limited to:

- Verifying the applicability and effectiveness of the original scheduled maintenance tasks

- Reviewing the records for all maintenance significant items at pre-set trigger points (eg numbers of events) to understand if there are any issues which needs addressing, and ensure that the relevant action is taken. For critical items the trigger point can be one event, for less critical items the number of events has to be judged against the amount of usage over a period of time.
- Reviewing maintenance reports for the items at pre-set trigger times (eg amount of use or calendar times as determined in the maintenance plan) to decide if the design of maintenance needs to be investigated. For example, if maintenance records show less "wear" than expected, consideration is given to extending the maintenance frequency, the actual decision is undertaken as a repeat of design of maintenance.
- Continuing to review more advanced maintenance approaches such as condition monitoring, to ensure that the threshold levels set remain appropriate
- Ensuring that all maintenance review activities balance the risk of failure against the uncertainty inherent in the maintenance records
- Ensuring that an analysis is conducted to assess the rationality of the task interval/threshold based on the in-service reliability data
- Using in-service data is used to assess the access method and removal routes identified for the tasks based on in-service use and routes modified where necessary
- Redefining the tasks type, interval and access method if needed, and transfer the revised or added tasks to MTA for further analysis

Note

The analysis activities described above are a selection of criteria for a reassessment of scheduled maintenance. For a detailed process for the optimization of scheduled maintenance refer to S4000P. This optimization process can be considered as a "test bench" for preventive/scheduled maintenance programs and is published as a part of S4000P.

3.3.5

Reassessment and update of the LORA

The recommended analysis activities of the results of the LORA within LSA, include but are not limited to:

- Judging whether the Economical LORA (ELORA) needs to be supplemented for the high cost items that have relative lower reliability
- Using the in-service data to determine through the cost and elapsed time of the LRU replacement and repair, whether the LRU level is still reasonable. If not, a possibility for the LRU being divided into several separated LRUs can be evaluated.
- Judging whether the decision about replace/repair level (including disposal) needs to be updated or not

3.3.6

Reassessment and update of the MTA

An LSA manager must regularly review the maintenance policy, as agreed with the customer/user to ensure that the assumptions are still valid. Factors to be considered include but are not limited to:

- The validity of the maintenance plan including the LORA and SMA results
- Identified problems from operator's side in relation with performance of any maintenance task
- Divergence of as maintained standards from the Product build standard
- Changes to legislation
- Changes to usage environment
- Quantities fielded versus distribution and quantity of spares
- The accuracy of original wastage estimates
- The Products' performance/availability falling below or exceeding original predictions

The recommended analysis activities of the results of an MTA within LSA, include but are not limited to:

- Collecting the revised tasks coming from FMEA/FMECA, damage and special events analysis, LROA, SMA and SSA and reanalyze the maintenance tasks for the new or revised tasks
- Updating the maintenance procedure and the related resources for the revised requirement related to the maintenance task itself
- Updating the facilities requirements required according to the revised task procedures.
- Updating the personnel and according training requirements

3.3.7 Reassessment and update of the SSA

The recommended analysis activities of the results of an SSA within LSA, include but are not limited to:

- Updating the following during the in-service phase:
 - Software failure mode and effect analysis
 - Software testability analysis
 - Software maintenance requirements
- Updating the software maintenance tasks and troubleshooting tasks list and transferring them to MTA for further analysis

3.3.8 Reassessment and update of LCC analysis

During in-service, the results of LSA activities provide technical and logistic information that are necessary for updating the Product LCC, particularly with regards to the maintenance and support costs. Furthermore, the results of LSA activities with regard to disposal are crucial for the economical assessment of the disposal options. When design changes are decided, the same process described in [Chap 14](#) applies for evaluating design influence.

3.3.9 Reassessment and update of obsolescence analysis

Platforms and Products are procured against a predicted or required life. This can be extended or shortened for any number of reasons. The activities undertaken must plan not only for the disposal, but also have a strategy to detect the onset of obsolescence, of the whole or part of the Product. The support manager will then, if necessary, instigate action to find a suitable replacement if required, or extend the life of Product, if possible.

The signs of the onset of obsolescence include, but are not limited to:

- The unavailability of spares and prohibitive costs for replacements
- The lack of the necessary skills or training to maintain the Product
- The Product no longer being used due to changes of role, or introduction of enhanced capability

The recommended analysis activities of the results of an obsolescence analysis within LSA, include but are not limited to:

- Capturing all in-service data that impacts items selected for obsolescence management
- Ensuring that the obsolescence management strategy is reevaluated for items using the in-service data and ensure its continuing viability, during the in-service phase
- Ensuring that a risk assessment for components identified for proactive obsolescence management is carried out. Where this risk is considered unacceptable, management activities must be undertaken to identify and assess the most appropriate actions to mitigate the obsolescence concerns.
- Ensuring that component monitoring is undertaken to identify potential high risk components, that need to be added to the obsolescence management list during the in-service phase

3.3.10 Reassessment and update of disposal analysis

3.3.10.1 Requirements for disposal

During the in-service phase, modifications to the support solution can result in the appearance of substances that will require special disposal. These substances can also arise because of initial overstocking due to incorrect LSA results, implementation of modifications to the Product and changes in equipment use.

The requirement for disposal during the in-service phase must be analyzed continuously to ensure that the disposal system is adequate for the Products' or Product components' needs and meets all necessary environmental and legislative requirements.

3.3.10.2 Update planning for removal of equipment from service

When a Product reaches the end of its life it is the start of an extended activity that will lead to the main equipment, associated support equipment, special to type maintenance tools, unique spares and technical documentation being made surplus to requirements. There are also likely to be implications for training and facilities.

The LSA activities will be revisited once a Product or a Product component has been identified for removal from service to :

- Ensure design changes and modifications are reduced to the minimum required to satisfy safety and legislative requirements
- Review the maintenance strategy and reduce future maintenance requirements to ensure minimum support to satisfy remaining training and operational requirements
- Assess whether the Product or Product components, including that in storage, can be subject to spares recovery. This activity must be closely linked to obsolescence management activities.
- Assess the requirement for demilitarization/declassification tasks for the Product or Product components prior to disposal

3.3.11 Product defect and warranty management

Virtually all Products, together with their support systems, will be delivered into an environment where a legal framework will be in operation that gives purchasers of goods or services a measure of redress if they are determined not fit for purpose in some way. There is usually a time limitation on the purchasers' right to seek redress for shortcomings and this is termed the warranty period.

The manufacture of equipment normally stipulates conditions on how a complex Product is to be used and maintained if the warranty is to be redeemable. The warranty requirements often require:

- Transport of the Product or Product components to and from the originator in defined packaging
- Maintenance of the Product or Product components according to a schedule determined by the supplier
- The use of consumables and spare parts that are specified by the supplier

The configuration management system during the in-service phase will need to identify items subject to warranty and the support solution for the items developed with the warranty constraints will need to be given for consideration in the support solution design process.

Under some circumstances the requirements to maintain rights to warranty benefits can be traded out to enable the development and operation of a more efficient support solution from the initial usage of the Product.

Revisiting the initial maintenance schedule for the Product or Product components and the corresponding spares\consumables procurement strategy can be required once a Product or Product components are outside its warranty period if the benefits of alternative suppliers and a less rigorous maintenance schedule can be achieved.

Chapter 17

Disposal

Table of contents

	Page
Disposal	1
References.....	2
1 General	3
1.1 Introduction	3
1.2 Purpose	3
1.3 Scope.....	3
2 Disposal tasks during Product life cycle phases	4
2.1 Concept definition phase	4
2.2 Development phase.....	5
2.3 In-service phase	5
2.4 Disposal phase	6
3 Disposal analysis	6
3.1 Disposal analysis activities	7
3.1.1 Define a disposal strategy	7
3.1.2 Integration of disposal requirements in design..	7
3.1.3 Disposal method analysis	8
3.1.4 Hazards	8
3.1.5 Configuration management	8
3.1.6 Monitor and control of sub-suppliers	9
3.1.7 Task Input.....	9
3.1.8 Task output.....	10
3.2 Disposal operation task identification	10
3.2.1 Disposal process breakdown analysis	10
3.2.2 Task identification	10
3.2.3 Disposal hazard analysis.....	11
3.2.4 Task input	12
3.2.5 Task output.....	12
3.3 Level of disposal analysis	12
3.4 Disposal task analysis	12
3.4.1 Task description.....	12
3.4.2 Task input	12
3.4.3 Task output.....	13
4 Product disposal file	13
4.1 Product disposal file content.....	14
4.2 Product disposal file compilation process	14
4.3 Product disposal file data elements.....	16
5 List of regulations.....	20
5.1 European Union directives	20
5.2 Transportation regulations.....	20
5.3 Support equipment and ground equipment	20
5.4 Navy equipment.....	21
5.5 Air equipment.....	21
5.6 Ammunition.....	21
5.7 Nuclear waste	21

List of tables

1	References	2
2	Residual product.....	8
3	Risk score example	11
4	Proposed PDF data elements	16
5	Environment aspect codes	18
6	Websites for additional information concerning disposal of nuclear waste.....	22

List of figures

1	LSA aspects of disposal analysis process	6
2	Disposal analysis activities	7
3	PDF compilation process.....	15
4	Example of a PDF (for a given subsystem of a Product)	19

References

Table 1 References

Chap No./Document No.	Title
Chap 19	Data model
Chap 22	Data element list
European Community Directive 2000/53/EC	Directive on End of Life Vehicle (ELV)
European Community Directive 2002/95/EC	Restriction of (the use of certain) Hazardous Substances (RoHS)
European Community Directive 2002/96/EC	Waste from Electrical and Electronic Equipment (WEEE)
European Community Directive 2006/121/EC	Registration, Evaluation and Authorization of Chemicals (REACH)
European Union Directive 67/548/EEC	Directive on the approximation of laws, regulations and administrative provisions relating to the classification, packaging and labeling of dangerous substances
IAEA RS-G-1.8	Environmental and Source Monitoring for Purposes of Radiation Protection Safety Guide
IAEA WS-G-2.7	Management of Waste from the Use of Radioactive Material in Medicine, Industry, Agriculture, Research and Education Safety Guide
IAEA INF CIRC/386	Code of Practice on the International Transboundary Movement of Radioactive Waste
NI 528	Green passport (Bureau Veritas)
OECD/NEA SSR-5	Disposal of Radioactive Waste (2011)
Resolution A.962(23)	International Maritime Organization (IMO) Guidelines on Ship Recycling

Chap No./Document No.	Title
Resolution A.980(24)	Amendments to the IMO Guidelines on Ship Recycling (Resolution A.962(23))
STANAG 4518 Edition 1 (2001)	Ammunitions disposal

1 General

1.1 Introduction

Disposal is both a business process and the name of a phase in a Product life cycle. This chapter describes the disposal business process.

Regarding disposal, it is no longer acceptable to dump worn out equipment into the sea or anywhere else that can allow it to cause harm to human health or the environment. Disposal needs to be performed in a controlled manner that does not impact human health and does not cause the contamination of land, water or air.

In an S3000L implementation it is considered that:

- disposal is an event that occurs only once during a Product life cycle. Once disposed, the Product no longer exists.
- the probability that disposal will actually occur is 100%, at some point and in some manner

This leads to the conclusion that disposal requirements must be implemented into the design from the very beginning of a development program. The general disposal principles to be applied are:

- to have as much material as possible recycled
- to minimize landfill waste resulting from the Product's lifecycle

Therefore, disposal must be considered as one of the major business processes to be implemented within the framework of an ILS/LSA program development strategy.

To dispose of a Product means to phase it out at the end of the Product's service life. Typically the disposal of Products happens during the last phase of their life cycle, however, disposal must be considered from the very early phases of every project that acquires a Product.

1.2 Purpose

This chapter provides guidance for developing a disposal process applicable for a Product (existing or under development), that can be demonstrated to be safe, efficient, environmentally acceptable and cost effective.

It also provides guidance to analyze required operations and disposal tasks for the Product to:

- Identify logistic support resource requirements for each task
- Identify new or critical logistic support resource requirements
- Identify transportability requirements
- Identify support requirements that exceed established goals, thresholds, or constraints
- Provide data to support participation in the development of design alternatives to reduce disposal costs and optimize logistic support resource requirements
- Provide guidelines on what must be done during each project phase and how to properly measure the effort
- Provide source data for the preparation of required ILS documents (eg disposal instructions, training program, manpower and personnel lists)

1.3 Scope

Disposal activity must consider the following issues:

- Destruction/neutralization of toxic substances such as carcinogenic, mutagenic or reprotoxic that harm humans or the environment for short or long periods of time (eg chemicals, radioactive substances)
- Allow a sustainable development by recycling materials (eg reuse of assemblies, raw material) or converting them into energy (eg burning with toxic gas processing)
- Demilitarization of defense Products to avoid weapons proliferation and use by terrorist groups

This applies to:

- the Product disposal at end of the service life
- the waste generated throughout the Product operation life cycle including, but not limited to:
 - Ozone depleting substances (CFC, HCFC, halon)
 - Global warming effect (sustainable development)
 - Health care waste and sanitary waste
 - Stores (eg paints, solvents)
 - Ballast water and sediments
 - Anti-fouling paints (eg biocides)
 - Oily bilge water (sludge)
 - Wastewater (black and grey water)
 - Galley food waste
 - Solid waste (eg glass, paper, plastics)
 - Exhaust emissions (eg NOx, SOx, COx)
 - Spares and consumables (eg batteries, nuclear sources, oils)
- All of the resources deployed and used to operate and support the Product (eg infrastructures, facilities, support equipment)

The disposal process impacts the following project phases:

- In-service phase
 - Take the necessary provisions to design safe operation and support tasks against dangerous substances contained in the Product
 - Disposal process of consumables and items removed as a result of performance of operation and support tasks (health and environment impact limitation)

This information must be available in the LSA database (as operation and support tasks) and in the Product's technical publications.

- Disposal phase
 - Dismantling a disassembling
 - Demilitarization
 - Dispose of Products (eg energetic recovery, material recovery, re-use, landfill waste)

This information is available in Product Breakdown Structure (PBS) and Product Disposal File (PDF).

2 2.1

Disposal tasks during Product life cycle phases

Concept definition phase

A Product disposal strategy is created along with a concept of operation and a support policy.

This strategy must make provisions to appropriately address the following:

- Sustainable design to limit the use of scarce resources, and the environmental impact of Product disposal
- Identification of health and environmental regulations that are applicable (refer to [Para 5](#))

- Continuous survey (regulation watch) of list of black (see REACH regulation, appendix 17) and grey (see REACH regulation, appendix 14) substances that will be respectively forbidden and limited in use and quantity
- Definition of the method for recording and tracking the dangerous substances, applicable in house and flowed down to suppliers
- Estimate of the Rough Order of Magnitude of Product disposal cost

2.2 Development phase

The disposal activity during development phase is twofold:

- **Design the disposal**

This includes all plans made during the Product design in a sustainable development approach, such as limiting the use of scarce resources, reducing the environmental impact of Product disposal (eg no toxic substances, identification of materials for easy recycling).

For this, a PDF is compiled (refer to [Para 4](#)).

Carcinogenic, mutagenic or reprotoxic substances used in the Product design are recorded in the Product breakdown as part of Product data and configuration management. Chemical substances are identified by their Chemical Abstract Substance (CAS) code (or equivalent), according to the REACH European regulation requirements.

As the Product design matures, the PDF information is updated.

- **Dispose of the design**

This includes all plans made during development to further prepare for Product disposal phase. It includes the following:

- Perform a Product dismantling task analysis, where tasks required to dispose of the Product are identified and characterized (eg tasks duration, required resources). This work can be considered as an extension of the Maintenance Task Analysis (MTA), using the same structured approach.
- Assess the impact of disposal on Product logistic support elements such as additional support equipment required to perform the Product disposal, describe the disposal tasks in the technical publications, perform training courses to teach operators how to dispose of the Product (including health and safety cautions)
- Estimate the cost of the Product disposal. This work can be a part of the Product global Life Cycle Cost (LCC) analysis work, using the same structured approach.

2.3 In-service phase

During this phase:

- the PDF is updated to take into account the following:
 - Design changes in the Product during its phase of operation and support
 - Evolution of applicable regulations (eg new banned substances, strengthening of health and safety)
- resources required for Product disposal are produced and delivered
- disposal demonstration can be performed on Product prototypes to validate:
 - Relevancy and completeness of information compiled in the PDF.
 - Dismantling and disposal operations (eg people skills, technical publications contents, support equipment adequacy)

Equipment to be replaced (eg batteries, oils) according to the maintenance plan must be disposed of in compliance to disposal requirements identified in the disposal analysis performed during the previous Product life cycle phases.

2.4 Disposal phase

During this phase:

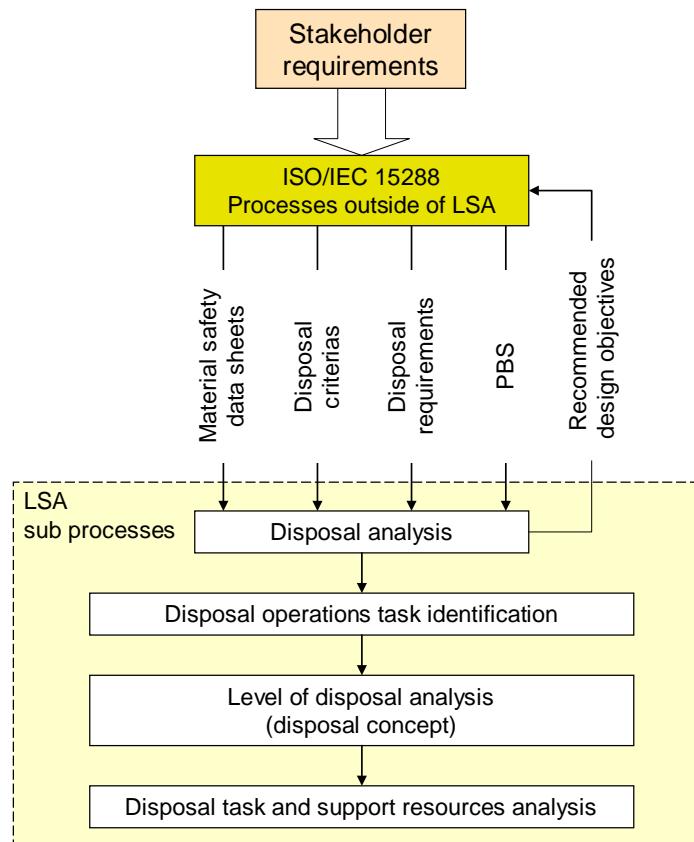
- The disposal strategy defined during the concept definition phase is implemented to dispose of the Product (eg dismantling, recycling, energetic valorization)
- The disposal operations are facilitated due to:
 - A Product design that is disposal oriented (design the disposal)
 - An available set of resources developed in order to perform the Product disposal (disposal of the design)
- Lessons learned are collected (eg cost, best practice, things to avoid) to be fed back to other projects

3 Disposal analysis

The LSA activities logic to determine the disposal process, disposal tasks, and support resources required, starts with the disposal analysis that needs to be tailored to the project (refer to [Fig 1](#)).

This analysis applies to both:

- Product components to be disposed of all along the Product life cycle, as a result of the Product maintenance plan implementation
- The entire Product, at the end of its operational life

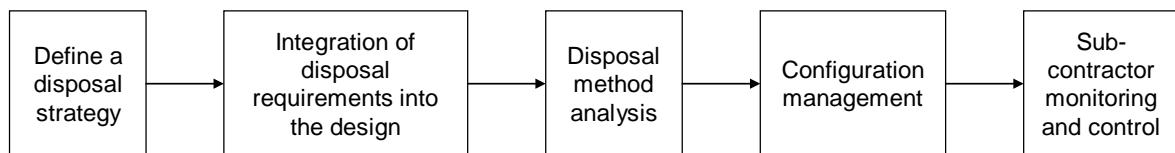


ICN-B6865-S3000L0089-001-01

Fig 1 LSA aspects of disposal analysis process

3.1 Disposal analysis activities

The LSA activities to be considered and tailored for the disposal at the end of a Product's life are summarized in an activities logic of the disposal analysis, as shown in [Fig 2](#):



ICN-B6865-S3000L0090-001-01

Fig 2 Disposal analysis activities

3.1.1 Define a disposal strategy

Define the disposal criteria for each item in the PBS and establish a design solution.

3.1.2 Integration of disposal requirements in design

Upon the development of a new Product, the disposal requirements must be balanced against those for performance, reliability, safety and life cycle cost.

The design principles must be a guide for design work and design considerations. Materials used in the design of the Product and its packaging must be selected to meet planned service life requirements.

3.1.2.1 Recommended materials

Materials to be used in the design of the Product and its packaging shall be selected to minimize:

- hazards to personnel or property during preparation for disposal
- hazards to personnel, property or the environment as a result of the processes used in disposal
- residual hazards from the materials which must be handled during disposal process

All materials must be fully specified, along with their associated hazards, in their primary state and during the stages of the disposal process. Material identification and marking processes must be defined. This information must be recorded in the PDF (refer to [Para 4](#)).

3.1.2.2 Accessibility and extraction of components and hazardous materials

Provide design recommendations with requirements for a minimum of non-standard processes, tools and technical expertise for access and extraction, and identify any non-standard processes or tools needed for access to components and hazardous materials, as well as any hazards associated with such processes.

3.1.2.3 Select components and materials to facilitate safe recovery and recycling

Identify the components and the base materials likely to be recovered from the Product and their potential for recycling. The assessment must consider the possibility of degradation and contamination of components and materials during their service life to ensure that recovery and/or recycling is viable.

3.1.2.4 Complete material list

List all materials used in the Product in a supplier material list and make it available to all personnel it may concern in the project. As a minimum, the list includes, but not limited to:

- Components
- Materials
- Compatibility aspects
- Residual products used by the Product

- Residual products at planned disposal
- Impact on the environment at planned disposal
- Applicable health and environmental regulations

3.1.3 Disposal method analysis

Disposal method analysis includes:

- 1 Identify potential alternative applications (re-use) for the Product and its components, including software. Demonstrate that they are technically achievable, environmentally acceptable and economically viable.
- 2 Identify components and material that must be recycled
- 3 Identify components and material that must be recovered
- 4 Identify components and material that will be land waste
- 5 For each alternative above, define the residual products. Refer to [Table 2](#).
- 6 Document the results in a disposal PBS
- 7 Disposal of sensitive information
When Products to be disposed of contain proprietary, confidential or classified software and data, the dismantling procedure includes:
 - 7.1 The transfer of the necessary historical information to a media or other storage device for proper preservation or historical archiving,
 - 7.2 The destruction of the data/software media or the erasure of the sensitive, proprietary, and classified information and software in an adequate manner so as to ensure that such information/software cannot be recovered by third parties, in accordance with the proper security regulations.

Table 2 Residual product

Residual products	Re-useable/recyclable [g]	Combustible [g]	Waste [g]	Accumulated weight [g]
Sum	20	2000	3000	5020

3.1.4 Hazards

Identify all hazards associated with the design and in particular:

- Provide an estimate of the likely levels of noise, flash, and toxic or corrosive fumes produced upon operation/firing/launching
- Identify any disposal activity hazards associated with the Product design that requires additional safety arrangements other than the existing disposal resources
- Identify any residual hazards from the disposal of the Product that requires additional safety arrangement other than the existing disposal resources
- Consider possible means of fail safe
- Identify any features that can pose an unusual hazard to Explosive Ordnance Disposal operatives, for example, anti-tampering/interference fuses or energetic materials with particularly high levels of sensitivity. This assessment must consider the effects of aging on the Product.

3.1.5 Configuration management

For any component/design change, assess its impact on the:

- Product life extension

- reduction of health and environment hazards
- and demonstrate that such changes are both technically achievable and economically viable.

3.1.6 Monitor and control of sub-suppliers

Product elements obtained from suppliers must meet the disposal requirements. This effort applies to equipment obtained from any supplier, regardless of tier. Review all sub-supplier contracts and perform evaluations of their disposal procedures.

If Government Furnished Equipment is used, it is treated as ordinary sub-supplier items. The same requirements shall be applied as for those suppliers.

During the subsequent phases of the development program, sub-supplier specifications shall incorporate aspects such as:

- Disposal requirements
- Design constraints and requirements
- Material Safety Data Sheets (MSDS)

3.1.7 Task Input

The inputs for the identification of the disposal analysis activities are:

- Disposal strategy
- Disposal requirements
- Disposal criteria
- Product Breakdown Structure (PBS)
- MSDS detailing:
 - The Product name used on the label, and chemical and common names of ingredients which have been determined to be health hazards, and which comprise 1% or greater of the composition, except carcinogens shall be listed if the concentrations are 0.1% or greater
 - The chemical and common names of all ingredients which have been determined to present a physical hazard when present in the mixture
 - Relevant physical and chemical characteristics of the hazardous chemical (such as vapor pressure, flash point)
 - Relevant physical hazards, including the potential for fire, explosion, and reactivity.
 - Relevant health hazards, including signs and symptoms of exposure, and any medical conditions generally recognized as being aggravated by exposure to the chemical
 - The primary routes of entry into the body
 - The Occupational Safety and Health Administration permissible exposure limit and American Conference of Industrial Hygienists threshold limit value. Additional applicable exposure limits may be listed.
 - Whether the hazardous chemical is listed in the National Toxicology Program Annual Report on Carcinogens (latest edition) or has been found to be a potential carcinogen in the International Agency for Research on Cancer Monographs (latest editions), or by Occupational Safety and Health Administration
 - Precautions for safe handling and use, including appropriate hygienic practices, protective measures during repair and maintenance of contaminated equipment, and procedures for clean-up of spills and leaks
 - Appropriate control measures, such as engineering controls, work practices, or personal protective equipment
 - Emergency and first aid procedures
 - The date of preparation of the MSDS or the last change to it
 - The name, address and telephone number of the chemical manufacturer, importer, employer or other responsible party preparing or distributing the MSDS, who can

provide additional information on the hazardous chemical and appropriate emergency procedures, if necessary

3.1.8 Task output

The outputs for the identification of the disposal analysis activities are:

- Scope of the disposal business process to be applied.
- A disposal PBS list, detailing which items can go for re-use, recycling, recovery or landfill waste.

3.2 Disposal operation task identification

3.2.1 Disposal process breakdown analysis

Define a representative set of activity sequences to identify all required services that correspond to anticipated disposal process of a Product and environments.

The task shall list the method recommended for disposal of the Product items. Disposal involves two aspects:

- Dismantling the Product
- Disposal of materials and contaminated items, including contaminated solvents

A breakdown of the recommended process must be updated as details of the design become available.

This requires the use of the following data elements:

- a disposal task identifier
- disposal recovery codes for the Source Maintenance & Recoverability (SMR) code, coding for the method by which the Product is intended to be disposed of (eg recycled, energy recovered, landfill waste)

3.2.2 Task identification

Identify and document the operations/tasks that must be performed for disposal of the Product and its components. These tasks must be identified to a level commensurate with design, including disassembly diagrams, step-by-step procedures, safety precautions and component and material tables, taking into account possible changes due to aging, and final destination of all Product components. Any special tools and equipment required and any limitations on locations must be identified.

A task inventory must be prepared for the Product and its components. This task inventory must identify all tasks that operators, maintainers, or support personnel must perform with regard to disposal of the equipment/components (hardware and software) based on the disposal process breakdown analysis, and scenarios/conditions. Tasks must be identified to a level of detail commensurate with design and the disposal scenario development. The task inventory must be organized in terms of a task classification which defines scenario/conditions, function, job and duty, task and subtasks and task resources. The task inventory must be composed of task descriptions, each of which consists of:

- An action verb which identifies what is to be accomplished in the task
- An object which identifies what is to be acted upon in the task

Task descriptions must be clear and concise. Hazardous materials, generation of waste, release of air and water pollutants and environmental impacts associated with each task must be identified. Where the same task appears in the duty of more than one job, and is therefore identified in a collective task for training purposes, it will be identified as such within the task inventory. All verbs must be unambiguously defined within the task classification.

3.2.3 Disposal hazard analysis

The hazard analysis of the chosen disposal process must identify the hazards related to the process and its products, and must quantify the associated risks. The results of the analysis can be presented using a matrix (refer to Table 3), where the risk scores are:

- 2 = Serious
- 1 = Moderate
- 0 = Harmless

Table 3 Risk score example

Risks	Type of risk	Weight	Hazards from planned disposal process	Total (Risk x Weight)
2	Safety	3	Detonation of warhead at dismantling of missile	6
1	Health	3	Toxic gases produced at burning of launch rocket motor propellant	3
1	Environmental	2	Environmentally polluting gases produced at burning of launch rocket motor propellant.	2

The hazard analyses of the Product must include a description of the Product with its configuration. The composition and mass of each material must be listed along with an aggregate mass of all materials in the item. The hazards associated with each material must be identified along with any special handling requirements. Both primary hazards and secondary hazards (those derived from material changes that occur as a result of the processes used in disposal) must be described in the analyses. All materials to be transported must be accompanied by safety data sheets.

Risks to be listed and considered are, for example:

- Explosive hazards
Risks associated with explosive materials in the Product
- Fire hazards
All flammable and oxidizing materials
- Caustic hazards
All caustic materials
- Ingestion, absorption, inhalation and adsorption
All materials that have a physiological effect on personnel or may damage property. The list must indicate the physical form of the material, the means of attack and the protection required.
- Trauma inducing
All non-explosive energy stores such as batteries, electrical and electromagnetic generating devices, capacitors, sources of static charge, springs under tension or compression, and any material that can transfer energy upon release, sufficient to cause trauma. The list must indicate likely energy output, likely effect and the protection required.
- Environmental hazard
Any potential environmental hazard not already covered. These must include potential damage to the atmosphere, soil and groundwater.

3.2.4 Task input

The inputs for a disposal operation task identification analysis are:

- Delivery identification of any data item required
- Identification of equipment hardware, software and firmware on which this task will be performed and the indenture levels to which this analysis will be carried
- Identification of the levels for disposal of hardware, software and firmware which will be analyzed during performance of this task to identify operations and tasks
- Any documentation requirements over and above the LSA database, such as functional flow diagrams or drawings
- Description of Product concepts under consideration
- Disposal method (disposal PBS)

3.2.5 Task output

The outputs for the disposal operation task identification are:

- A task inventory documented in the LSA database, or equivalent format approved by the project, identifying disposal operational tasks requirements, to include task descriptions on Product hardware software and firmware and to the indenture levels specified by the project
- Identification of any hazard risks involved in satisfying the disposal operational tasks requirements

3.3 Level of disposal analysis

The level of disposal will be driven by the hazardous substances location in the Product breakdown. The Product must be dismantled and disassembled to a level that can be disposed of using the same disposal method as defined in [Para 4](#).

3.4 Disposal task analysis

3.4.1 Task description

The detailed disposal task analysis based on the principles of MTA contains:

- Conduct a detailed analysis of each operation, maintenance and support task contained in the task inventory and determine the following:
 - Logistic support resources required (considering all ILS elements) to perform the task
 - Elapsed time and man-hours of the Product intended disposal facility/environment
 - Maintenance level assignment based on the established support plan
 - Environmental impact of the tasks, including use of hazardous materials, generation of hazardous waste and its disposal, and the release of air and water pollutants
- Identify new or critical logistic support resources required to perform each task, and the hazardous materials, hazardous waste and environmental impact requirements associated with those resources. New resources are those which require development in order to disassemble the new Product. These can include support and test equipment, facilities, new or special transportation products, new computer resources, and new test or inspection techniques. Critical resources are those which are not new but require special management attention due to schedule constraints, cost implications, or known scarcities. Unless otherwise required, document new and modified logistic support resources in the LSA database, or with equivalent documentation approved by the project, in order to provide a description and justification for the resource requirement.
- Conduct a transportability analysis on the Product and any sections thereof when disassembling is required for transport
- Document the results in the LSA database, or by an equivalent format approved by the project

3.4.2 Task input

The inputs for a disposal task analysis are:

- Identification of Product hardware and software on which this analysis will be performed
- Identification of indenture levels to which this analysis will be performed
- Identification of the levels of disposal operations that will be documented during performance of this task
- Known or projected logistic support resource shortages
- Any supplemental documentation requirements over and above the LSA database (eg transportability clearance diagrams)
- Delivery identification of any data item required
- Information available from the project relative to:
 - Existing and planned personnel skills, capabilities, and program of instruction
 - Lists of standard support and test equipment
 - Facilities available
 - Training devices available
 - Existing transportation products and capabilities
- Description of personnel capabilities (target audience) intended for disposal of the Product at each level of disposal
- Operations and disposal task requirements from task (disposal operations task identification)
- Recommended disposal plan for the Product from task (level of disposal analysis)
- Supportability and supportability related design goals and requirements

3.4.3

Task output

The outputs of the disposal task analysis are:

- Completed LSA database on Product hardware and software to the indenture level specified by the project, or equivalent format approved by the project
- Identification of new or critical logistic support resources required to operate and maintain the new Product
- Alternative design approaches where tasks fail to meet established goals and constraints for the new Product or where the opportunity exists to reduce disposal costs or optimize logistic support resource requirements
- Identification of management actions to minimize the risks associated with each new or critical logistic support resource requirement
- Validation of key information documented in the LSA database
- Output summaries and reports as specified by the project containing all pertinent data contained in the LSA database at the time of preparation
- An LSA database that is updated as better information becomes available and as applicable input data from other Product engineering programs is updated

4

Product disposal file

The PDF is intended to compile all data information regarding Product disposal. It must be updated throughout the Product lifecycle in order to reflect Product changes (configuration management), maintenance plan evolutions, and health and environmental regulations modifications.

This file must allow the Product project management to clarify the operational limits of the authorized substances over a given period, in order to anticipate the research of alternative solutions which are more suitable in terms of:

- Sustainable development
- Health and safety impact on humans and environment

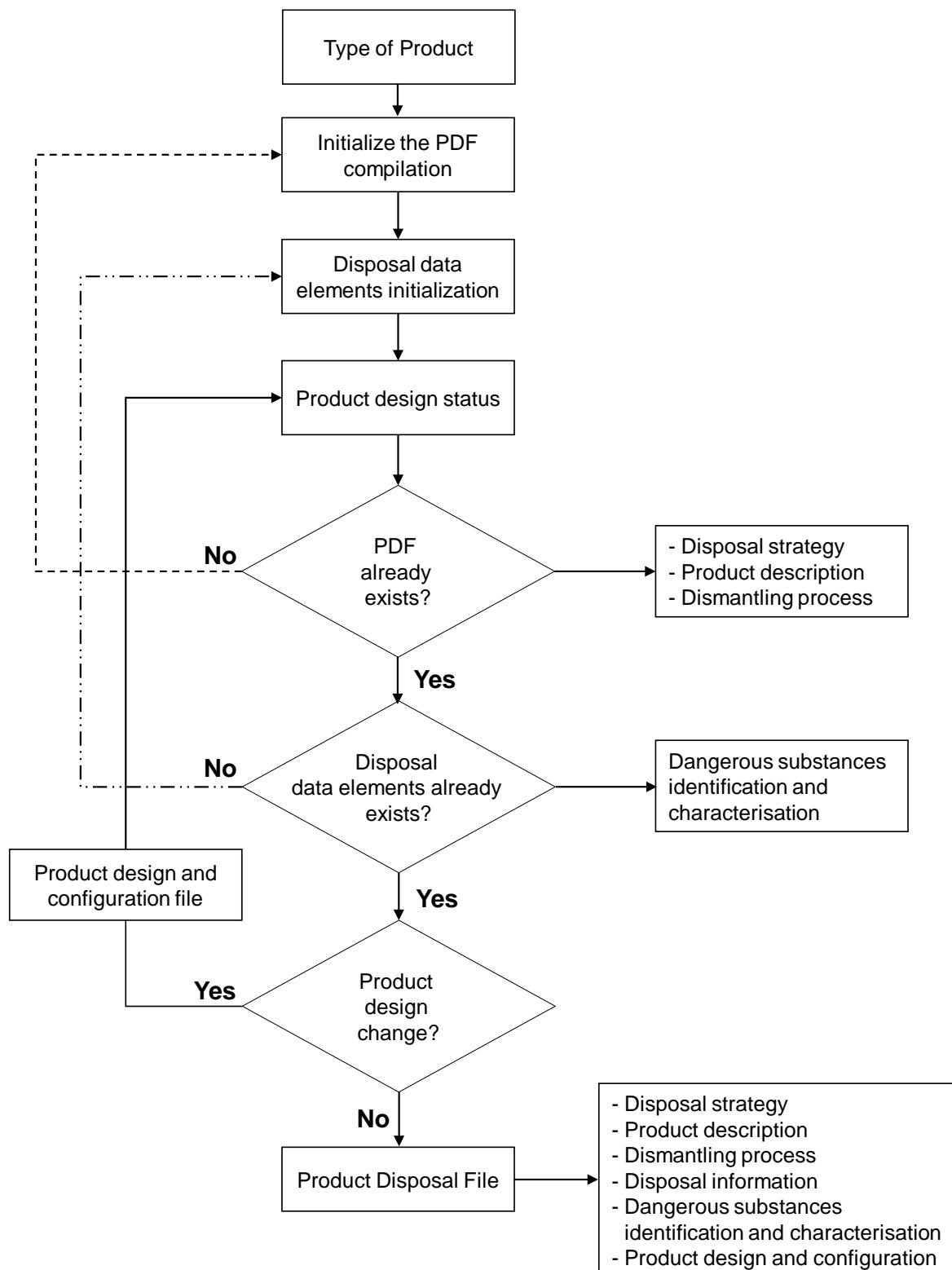
4.1 Product disposal file content

Whatever the Product, the PDF may compile information such as, but not limited to the following:

- Product breakdown and associated functional and physical description
- Related to the Product breakdown (with illustrations wherever possible), identification of components, products or ingredients (eg containing hydrocarbon, lubricating, paintings)
- For each dangerous substance identified in the Product, list of the risks for people and environment
- For each type of Product component, according to the type of technology (eg electronic, mechanic, hydraulic and pyrotechnic) and the contained substances, identify their possible processing at disposal.
 - those immediately reusable on another Product
 - those containing high value material (eg gems, precious metals)
- The methods of elimination that will make the difference between potentially harmful substances and harmless material
- Conditions for disassembly and the destination of the components (eg re-use, recycling, energetic valorization)
- The procedures of disassembly down to the required level of depth, such that all the materials (including in particular those harmful or dangerous) can benefit from the processing and their mode of elimination
- The rules, instructions, measurements or provisions to apply that guarantee safe practices at a sufficient level
- The collection of all safety data for all the harmful or dangerous materials composing the entire Product
- The detailed step by step procedures for dismantling and disposal
- A list of proposed organizations certified for proper recycling and elimination of the dangerous residues, according to the applicable regulation

4.2 Product disposal file compilation process

The PDF compilation process is shown in the flowchart in [Fig 3.](#)



ICN-B6865-S3000L0091-001-01

Fig 3 PDF compilation process

4.3

Product disposal file data elements

This paragraph presents a proposed list of data elements to be compiled in the PDF. This list is not restrictive and must be agreed to on a case by case basis (eg within LSA GC), depending upon the context of each specific project and contract.

Text in *italic* in the first column of [Table 4](#) refers to potential data element descriptions within a PDF (refer to [Fig.4](#)).

Text in *italic* in the second column of [Table 4](#) refers to S3000L data elements (if applicable) which can potentially cover the data requirement from a PDF (refer to [Chap 22](#)).

Table 4 Proposed PDF data elements

Header in PDF <i>(Meaning in PDF)</i>	Meaning <i>(corresponding S3000L data element, if available)</i>
Subsystem	Identifies the subunit to which the item belongs in the Product.
<i>Assembly identifier</i>	<i>breakdownElementIdentifier</i>
Item	Name of component according to drawing
<i>Part name</i>	<i>partName</i>
Article number	Refers to drawing No. for unequivocal identification or (for some components such as screws and adhesives) other article number.
<i>Part number</i>	<i>partIdentifier</i>
Material Denomination	Short description of material in component (if known)
<i>Material description</i>	<i>substanceDescription</i>
CAT	The main substances category:
<i>Material Category</i>	<i>substanceUsageCategory</i> <ul style="list-style-type: none"> – black (forbidden) – grey (authorized with limitation) – green (authorized with no limitation)
CAS	Chemical Abstract Substance identification number
<i>Material Code</i>	<i>substanceIdentifier</i>
Phrase of Risk	Phrases of risk (in accordance with REACh regulation Safety Data Sheet data)
<i>Type of Risk</i>	<i>substanceRiskDescription</i>
MSDS ref.	Reference of the associated Material Safety Data Sheet Document classified as 'Material Safety Data Sheet, assigned to the Material Class in the role of 'Reference'
<i>MSDS</i>	<i>Can be referred to as a document with the mean of the UoF Document in S3000L data model, refer to Chap 19.</i>
Service life	Product service life
<i>Service life</i>	<i>hardwarePartOperationalAuthorizedLife</i>
Mass (g)	Weight of component (in grams)
<i>Mass</i>	<i>quantityOfContainedSubstance</i>

Header in PDF <i>(Meaning in PDF)</i>	Meaning <i>(corresponding S3000L data element, if available)</i>
Reg.	Applicable environmental regulations
<i>Regulations</i>	<i>Can be referred to as a document with the mean of the UoF Document in S3000L data model, refer to Chap 19.</i>
SLA No.	Serial number of component
<i>Serial Number</i>	<i>not applicable</i>
Risk factor (SLA)	Judged risk factor (criticality) of component (1 to 4 scale)
<i>Risk Criticality</i>	<i>partHazardousClass in relation with substanceRiskFactor</i>
Critical property	The most important property for the function of the component
<i>Justification</i>	<i>containedSubstanceJustificationDescription</i>
Proceedings in connection with use	Describes how the component is to be disposed of during the in-service phase.
<i>Disposal concept in use</i>	<i>Documented in the Units of Functionality which cover all the data of a maintenance task</i>
Proceedings in connection with planned disposal	Describes how the component is to be disposed of during planned disposal.
<i>Disposal concept end of life</i>	<i>Documented in the Units of Functionality which cover all the data of a maintenance task</i>
Waste products in connection with use	States the waste products when the component is disposed of according to the procedure during use or after being used.
<i>Waste products in use</i>	<i>hardwarePartWasteProductsInUseDisposalDescription</i>
Waste products in connection with planned disposal	States the waste products when the component is disposed of according to the procedure for planned disposal.
<i>Waste products disposal</i>	<i>hardwarePartWasteProductsPlannedDisposalDescription</i>
Environmental aspects, use	Classification of environment aspects during use. Refer to codes in the "Environment aspect codes", refer to Table 5 .
<i>Environmental aspects in use</i>	<i>hardwarePartEnvironmentalAspectInUseClass</i>
Environmental aspects, planned disposal	Classification of environment aspects at end of life. Refer to codes in the "Environment aspect codes", refer to Table 5 .
<i>Environmental aspects end of life</i>	<i>hardwarePartEnvironmentalAspectPlannedDisposalClass</i>
Task	Dismantling tasks characterization and description, identifying required resources and associated safety cautions to be taken to preserve human beings and environment.
<i>Task requirement</i>	<i>Documented in the Units of Functionality which cover all the data of a maintenance task</i>

Header in PDF <i>(Meaning in PDF)</i>	Meaning <i>(corresponding S3000L data element, if available)</i>
Last updated SLA/Environ	Date of latest changes SLA respective environmental part
<i>Date</i>	<i>not applicable</i>

Table 5 Environment aspect codes

Code	Meaning
tox	Harmful to environment according to note g and h in MBT04072. (toxic, bioaccumulation and/or difficult to decompose)
acid	Acidification
ozon	Dangerous for ozone layer
gwp	Greenhouse effect
mw	Material waste
pos	Environmental aspects, positive
ene	Energy regaining by burning
mr	Material recycling

Item	Article number	Material denotation (specification)	C A T	C H S	P R I	MSDS ref.	Service life	Mass (g)	R E G	SLA No.	S L A	Critical Property (Important for service life qualification)	Proceedings in connection with		Waste products in connection with		Environmental aspects		T A S K	Last up-dated SLA/ Environ
													Use	Planned/ Disposal	Use	Planned/ Disposal	Use	Planned/ Disposal		
Outside tube	4113499	PC 10% glass fibre Black						58.0	1	4	Mechanical strength	Thrown	Burned/ recycling	Stable	COx/PC	mw	ene, gwp/hr	040327		
Plug	5229068	PC 10% glass fibre Black							1.1	2	Mechanical strength	Thrown	Burned/ recycling	Stable	COx/PC	mw	ene, gwp/hr	040327		
Support Stand	4113498	PC 10% glass fibre Black							26.7	4	Mechanical strength	Thrown	Burned/ recycling	Stable	COx/PC	mw	ene, gwp/hr	040327		
Slotted screw MCS M4x25	6436317	Stainless Steel ISO 1207							3.0	10	1	Mechanical strength	Thrown	Recycling	Steel	mw	mr	030130		
Connector cover	2799541	Aluminium EN 1706 AC42000-T6 Chromated Cr							5.0	368	2	Mechanical strength	Thrown	Recycling	Al	mw	mr	030301		
Adhesive	10363883	Lactile 406							0	3	4	Adhesion	Thrown	Burned	COx, NOx	mw	ene, gwp, acid, ozon	030130		

ICN-B6865-S3000L0092-001-01

Fig 4 Example of a PDF (for a given subsystem of a Product)

5 List of regulations

This list identifies international regulations for health, safety and environmental conditions.

The applicability status of the regulations must be defined on a case by case basis for each specific project and contract.

5.1 European Union directives

- The European Community Directive (2002/96/EC), Waste from Electrical and Electronic Equipment (WEEE) valid from 2006, may be used as a guideline stating target values for Recovery, Recycling and Re-use
- The European Community Directive (2006/121/EC), Registration, Evaluation and Authorization of Chemicals REACH, concerning the recording, the evaluation and the authorization of the chemical substances, as well as the restrictions applicable to these substances, instituting a European agency of the chemicals
- The European Community Directive (2002/95/EC) of the European Parliament and the Council of January 27, 2003 relating to the limitation of the use of certain dangerous substances in electric and electronic components

5.2 Transportation regulations

Transportation of Products recommended to be returned for safe disposal, must be approved by national/state-run Competent Authorities which issues transport certificates stating the components transportation classification based on the UN recommendation on the transport of Dangerous Goods ("orange book"). This international recommendation classifies dangerous goods by type of risk involved, UN classification code, and product category codes, UN Number.

The UN Number defines the packing group and packing instruction for the Product, maximum weight of the package and how the package shall be labeled for each mode of conveyance. Note that the packing itself need to have type approval issued by a Competent Authority.

The recommendations need to be tailored for transportation by road, railway, sea and air.

- Agreement on Dangerous Goods by Road (Europe)
- Regulations Concerning the International Transport of Dangerous Goods by Rail (European law)
- Dangerous Goods Regulations IATA - international recommendation for air transports
- International Maritime Dangerous Goods (United Nations) for sea transports

The Transport certificates are often national and normally have a period of validity of a maximum of 10 years. At the time for disposal, new certificates must be applied for at the Competent Authorities concerned.

5.3 Support equipment and ground equipment

The PDF will be based upon the 2000/53/EC directive from the European Parliament and the council relating to vehicles out of use.

This directive is valid only for the civil vehicles of a weight lower or equal to 3,5 ton. This directive will be used as a basis for the development of a future regulation which will include all the civil and military vehicles. This directive envisages certain measures and incentives so that the phase of dismantling is taken into account from the start of the design of the Product. It will particularly be based upon new standards, established according to scientific evolutions. The principal objective of this directive is the human, animal and environment protection, which will be done by better Vehicles out of use management, whose base will be the compilation of a disposal file which will allow providing the adequate solutions for the assumption of the waste responsibility or the use of less harmful replacement materials to the environment.

5.4 Navy equipment

The PDF must comply with the "green passport" requirements, whose principles are stated in the A.962(23) resolution from the International Maritime Organization (IMO) of December 5, 2003, supplemented by the A.980(24) resolutions and especially the A981(24) resolution of December 1, 2005, which is pressed on international texts into force such as the convention of Basle about trans-border transfer of waste.

The A.962(23) resolution is applicable to the maritime equipments of which one of the components is the "green passport", which is an instrument intended to facilitate the operations of dismantling and recycling of any ship that arrives at the end of its lifetime by the adoption of good practices all for the length of the life cycle of the ship (respectful of environment on the level of the dismantling site and making safe for the personnel carrying out this work), the objective being a progressive reduction of the use of any dangerous materials lasting the service life of the ship.

In order to have further information on the "green passport", refer to the NI 528 document published by Bureau Veritas.

The directives are intended to provide to the flag States, port States and re-user States, owners of ships, manufacturers of ships, suppliers of navy material and recycling installations, several indications on the "best practices", which take account of the ship recycling process throughout the ship life cycle.

They take account of certain codes and protocols relating to the environmental protection.

The list of the substances to be tracked is based on the directive 67/548/CEE of the European Parliament and the council, to which substances suggested by the states will be added intervening for the drafting of the convention.

5.5 Air equipment

No regulation or guidelines has yet been set up worldwide to dispose of aircrafts at the end of their service life.

5.6 Ammunition

The PDF will comply with the STANAG 4518 edition 1 (2001) "Ammunitions disposal".

The goal of this NATO standardization agreement is to provide a common base to the NATO members on the principles and requirements of safety for the design and the evaluation procedures in order to guarantee the safety of the operations of elimination of ammunitions.

5.7 Nuclear waste

Each state member of the International Atomic Energy Agency (IAEA) has its own standards to manage nuclear waste.

However, the IAEA released the three following documents (Safety Guides and information Circular) addressing nuclear wastes safety:

- Environmental and Source Monitoring for Purposes of Radiation Protection (RS-G-1.8)
- Management of Waste from the Use of Radioactive Materials in Medicine, Industry, Research, Agriculture and Education (WS-G-2.7)
- Code of Practice on the International Transboundary Movement of Radioactive Waste (INFCIRC/386)

Moreover, the Board of Governors approved a Safety Requirements publication, co-sponsored by the Organization for Economic Co-operation and Development/Nuclear Energy Agency (OECD/NEA), on geological disposal (SSR-5).

For further information refer to [Table 6](#) with a list of international websites.

Table 6 Websites for additional information concerning disposal of nuclear waste

Site address	Organization
International	
http://www.nea.fr/	Nuclear Energy Agency with 28 member states
http://www.iaea.org/	International Agency Energy Atomic with 144 member states
http://www.unscear.org/	United Nations Scientific Committee on the effects of atomic radiation

Chapter 18

Interrelations to other ASD specifications

Table of contents

Interrelations to other ASD specifications	1
References.....	1
1 General	2
1.1 Introduction	2
1.2 Objective	2
1.3 Scope.....	2
2 Benefits of using the ASD specifications suite	2
2.1 Conceptual background.....	2
2.2 Integrated concept of operation.....	4
3 Interrelation to S1000D.....	5
3.1 Purpose of S1000D	5
3.2 S3000L/S1000D	5
4 Interrelation to S2000M	5
4.1 Purpose of S2000M	5
4.2 S3000L/S2000M	6
5 Interrelation to S4000P	6
5.1 Purpose of S4000P	6
5.2 S3000L/S4000P.....	6
6 Interrelation to S5000F	7
6.1 Purpose of S5000F	7
6.2 S3000L/S5000F	7
7 Interrelation to SX000i	7
7.1 Purpose of SX000i.....	7
7.2 S3000L/SX000i.....	7

List of tables

1 References	1
-------------------------	---

List of figures

1 Acquisition logistics main business processes (1)	3
2 Acquisition logistics main business processes (2)	4

References

Table 1 References

Chap No./Document No.	Title
Chap 1	Introduction to the specification
Chap 10	Development of a scheduled maintenance program
Chap 20	Data exchange
DEX1A&D	Aerospace and Defense Product Breakdown for Support
DEX3A&D	Aerospace and Defense Task Set

Effectivity: All

S3000L-A-18-00-0000-00A-040A-A

Chap 18

Chap No./Document No.	Title
S1000D	International specification for technical publications using a common source database
S1003X	S1000D and S3000L interface specification
S2000M	International specification for material management
S4000P	International specification for developing and continuously improving preventive maintenance
S5000F	International specification for operational and maintenance data feedback
S6000T	International specification for training needs analysis
SX000i	International guide for the use of the S-Series Integrated Logistics Support (ILS) specifications
SX001D	Dictionary for the S-Series ILS Specifications
SX002D	Common data model for the S-Series ILS Specifications

1 General

1.1 Introduction

The S3000L specification should not be considered as a standalone entity. In the environment of logistics, there are existing ASD specifications which describe the development of technical documentation (S1000D) and define the materiel management processes and procedures to be used in support of a Product (S2000M). Also the new specification S4000P concerning Scheduled Maintenance Analysis (SMA) fits perfectly into the complete specification suite. The future specifications concerning the handling of in-service data and information (S5000F), performance of a Training Needs Analysis (S6000T) and a super ordinate specification concerning the Integrated Logistic Support Management (SX000i) will complete the specification suite and will cover the most important areas of Product supportability.

1.2 Objective

This chapter provides an overview of how the existing ASD specifications and the S3000L will be harmonized. It explains where the connecting points are and what benefits can be achieved by using the S3000L together with the other existing and well proven ASD specifications S1000D, S2000M and the new specifications S4000P, S5000F and SX000i.

1.3 Scope

This chapter is directed at logistics personnel who organize logistic analysis activities like an Integrated Logistics Support (ILS) manager or a Logistic Support Analysis (LSA) manager. The interdisciplinary character of the logistic supportability is also highlighted.

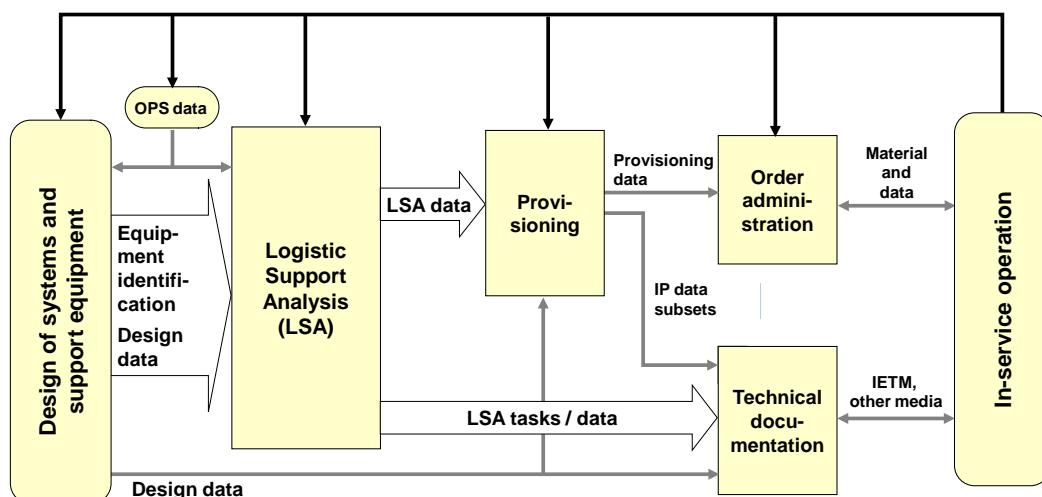
2 Benefits of using the ASD specifications suite

2.1 Conceptual background

The underlying basic concept for the development of a suite of ASD specifications in order to support the acquisition logistics main business processes was developed in early 1993. Refer to [Fig. 1](#). During an international workshop in Paris, with high-level industrial and government participation, the major domains of logistic support during an acquisition phase (generally for newly built systems and major retrofit campaigns) were identified and networked. These domains provide the framework for future specification development activities with the goal of covering the complete Product life cycle.

These domains were:

- Logistic support analysis
- Provisioning
- Order administration
- Technical documentation



ICN-B6865-S3000L0064-003-01

Fig 1 Acquisition logistics main business processes (1)

At the time of the ASD workshop in Paris, the technical documentation domain was already covered by S1000D and the provisioning and order administration domains were covered by S2000M.

No ASD specification existed, however, for the LSA at that time. In 2005, the ASD together with the Aerospace Industries Association of America (AIA) started the development of S3000L, International procedure specification for Logistics Support Analysis. Details thereof are provided in [Chap 1](#) and are therefore not repeated here.

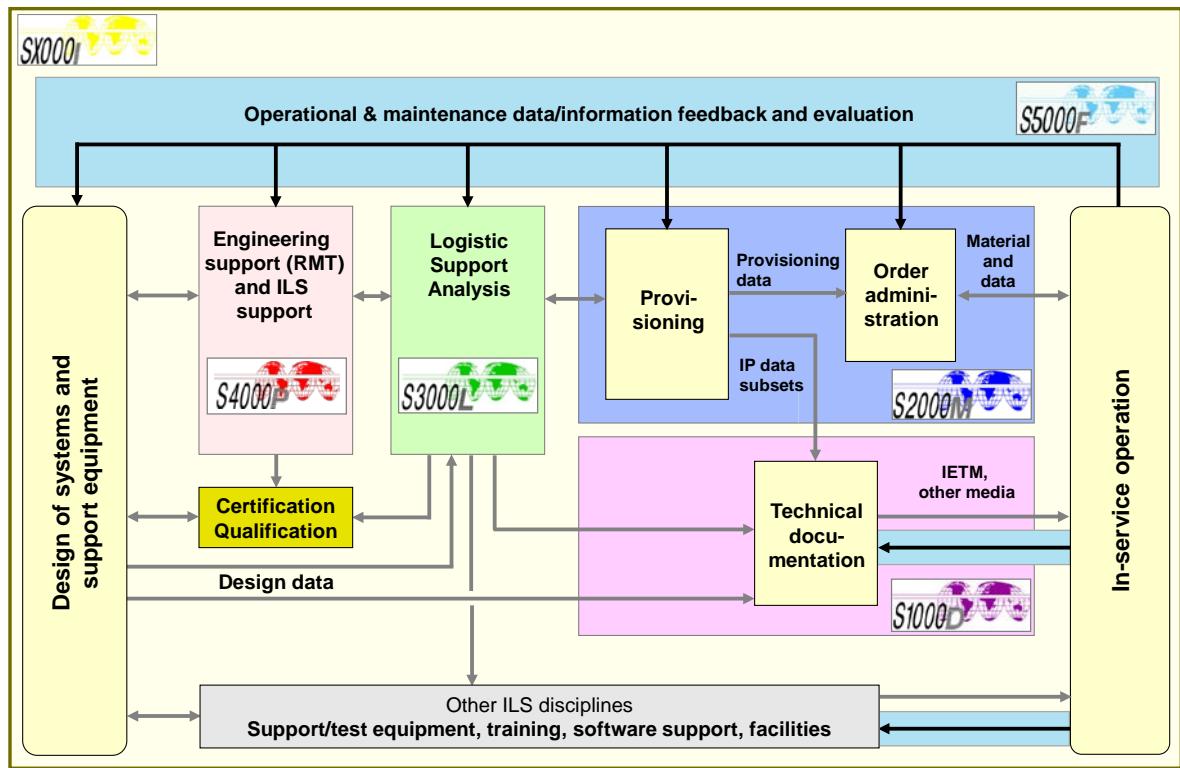
In the meantime it was obvious that additional specifications were needed to satisfy the requirements for the support of the full Product life cycle. The functionality concentrates on:

- Analysis of preventive Product maintenance requirements, covered by S4000P. This specification was published in May 2014.
- Maintenance and operational data and information feedback, covered by S5000F. This specification is still under development. A first draft was prepared for ASD internal commenting in April 2013, and a second draft also for external commenting was published in June 2014.
- Identification of required training for Product operation and maintenance, covered by S6000T. In 2014, this initiative was started to cover a further ILS discipline, the personnel training. S6000T will describe the process of a Training Needs Analysis (TNA).

An umbrella specification SX000i will be the guideline for the complete ILS process. The SX000i activity started as an ASD/AIA initiative in September 2011 and is under development by a team of specialists from operators, manufacturers and regulatory authorities.

The future network will comprise a total of seven specifications which will cover the major support functions of the Product life cycle, refer to [Fig 2](#) (S6000T not included in the figure). Additional future supplements which describe the data and information exchange processes (input data specifications, eg S3000X), an overall data dictionary SX001D and a common data

model SX002D (covering all data elements which are included in at least two different specifications) will improve and complete the ASD/AIA ILS Specification Suite.



ICN-B6865-S3000L0065-002-01

Fig 2 Acquisition logistics main business processes (2)

Note

For certification and qualification, also inputs from the ILS disciplines (eg, technical documentation, materiel support, support/test equipment) can be required.

Note

[Fig 2](#) is focused on the engineering support and ILS activities. However, in-service feedback can also be given to stakeholders outside of this frame, eg the customer or certification authorities.

2.2

Integrated concept of operation

The individual ASD specifications should not be considered as stand-alone entities.

In the environment of acquisition logistics they can be used as a coherent set (suite) of functional specifications with common data definitions and integrated data exchange. This integrated concept will ensure that all major functionalities of acquisition logistics are efficiently covered without overlap. The specifications are internationally accepted and are broadly used in the aerospace and defense business, but are not limited to these areas.

Of all the ASD specifications, S3000L plays a central role. It is the principal tool to:

- design the Products relevant to maintainability, reliability, testability and to optimize Life Cycle Cost (LCC)
- define all required resources to support the Product in its intended use during in-service operation
- maintain and update the basic Product support data and the Product maintenance concept during the entire service life

3 Interrelation to S1000D

3.1 Purpose of S1000D

S1000D was produced to establish an international specification for production and distribution of technical documentation and learning content. This specification can be applied to the documentation of any type of equipment including both military and civil Products.

Note

Since 2007, ASD, AIA and Air Transport Association of America (ATA) have jointly further developed, maintained and promoted S1000D in the international arena.

Note

The former Air Transport Association of America (ATA) is now represented by the trade organization Airlines for America (A4A).

3.2 S3000L/S1000D

The maintenance and operational tasks information developed during the LSA process is the baseline for the data modules to be produced in accordance with S1000D. The LSA data is also the input for the maintenance planning information.

The maintenance and operational tasks information includes as a minimum:

- Task description
- Preliminary requirements
 - Required conditions
 - Required resources (personnel, material, support equipment)
 - Safety conditions

This data must be used as a direct input to the corresponding procedure in the maintenance publications. Depending on the publication production environment, some of this data can be directly included in the procedure. During the update of the procedure, the author can be notified of any changes.

If a full set of maintenance planning data is developed in the LSA process this data can be transferred to the maintenance planning data modules.

Note

There is not always a one-to-one relation between a maintenance task originated in the LSA process and a procedural data module, since tasks can be grouped into one procedural data module or split into several reusable procedural data modules.

For S3000L, issue 1.0, the details of the data set to be exchanged are given in DEX1A&D and DEX3A&D and the specification S1003X, Issue 1.0 (S1000D, Issue 4.0 to S3000L, Issue 1.0 interchange specification). For S3000L, issue 1.1, data exchange principles are defined within [Chap 20](#).

Note

Interchange specifications, which were started by developing S1003X will be re-organized step by step by developing own data input specifications for each specification within the ASD/AIA ILS Specification Suite. For S3000L, the interface specification will be S3000X and will describe the data/information requirements of S3000L from the other ILS specifications, eg S1000D, S2000M or S4000P. Refer to [Para 2.1](#)

4 Interrelation to S2000M

4.1 Purpose of S2000M

S2000M originally defined the materiel management processes and procedures to be used in support of aircraft and other aerospace airborne and ground equipment supplied to military customers. S2000M has now been revised to include the business processes and data



applicable to any military Product. Although the S2000M was designed for military Product support, it can nevertheless be used for the support of any other complex Product.

The processes described within S2000M cover the interfaces between the contractor and the customer, which, when based upon contractual agreements, will provide the typical deliverables of the logistic materiel management:

- Provisioning
- NATO codification (as a special military feature)
- Procurement planning
- Order administration
- Invoicing
- Repair administration

4.2

S3000L/S2000M

During the S3000L LSA process, information will be generated that will determine the range and depth of the maintenance of the Product, as well as the required material resources during in-service operation. On the S2000M side, the appropriate functionality for the interface to S3000L is the provisioning functionality. Provisioning, according to chapter 1 of S2000M, is the process of selecting support items and spares necessary for the support of all categories of Products.

The compilation of provisioning data for those items identified as relevant for a customer's maintenance and support concept is the prime objective of that phase.

Provisioning data is compiled according to compilation rules specified in S2000M. These rules are normally confirmed or modified/specified during a Guidance Conference (GC). The GC takes place at the start of any project in which the S2000M procedures are required. During the GC, it will be specified in detail to what extent LSA data is used to support the provisioning process.

Note

It is recommended that data, which are common between S2000M and S3000L (simple example: the name of a part/equipment), are harmonized as much as possible. The leading discipline for technical values (eg a part identifier) must be determined during an ILS guidance conference.

5

Interrelation to S4000P

5.1

Purpose of S4000P

S4000P provides methodologies to develop preventive maintenance task requirements. This is the precondition for the determination of scheduled maintenance tasks with intervals which will be acceptable to operators, manufacturers and regulatory authorities (if applicable). The scheduled maintenance task and interval details will be developed by coordination with specialists from operators, manufacturers and the regulatory authority. Specifically, S4000P outlines the general organization and decision processes for determining preventive maintenance task requirements initially projected for the life of the Product and the continuous improving of preventive maintenance during the entire service life of a Product, which is covered by the In-Service Maintenance Optimization (ISMO) process.

5.2

S3000L/S4000P

LSA and SMA are very closely interconnected. Only the common view on unscheduled maintenance and scheduled or preventive maintenance respectively gives a complete impression of maintenance activities. The identification of the requirements for any scheduled task is the same as for the unscheduled tasks. For that reason, the documentation of the tasks in the LSA database can be done in almost the same way (only the justifying event differs). Also the packaging of the scheduled maintenance activities can be supported by the use of the LSA database. The scheduled tasks for each relevant equipment or subsystem can be documented in the LSA database as well as in the final relevant inspection and overhaul packages.

Rules for the documentation of scheduled maintenance packages within an LSA database must be established and harmonized with the customer during the LSA GC. It is strictly recommended that the validity of tasks is clarified within the LSA database.

The LSA tasks derived from the original SMA can serve as the documentation of SMA results, too. It should be possible for an S3000L application to link corresponding SMA documents. Also, the need for modification of task thresholds or intervals, or even the need for design changes, can be documented in the LSA as well as the original situation in the SMA tasks. Refer to [Chap 10](#).

6 Interrelation to S5000F

6.1 Purpose of S5000F

S5000F provides a means for the feedback of in-service data and information collected and prepared by the operator/customer to industry to support and improve maintenance and operations of a Product (refer to [Fig 2](#)).

6.2 S3000L/S5000F

In-service reality and the maintenance concept/tasks and the Product support requirements developed by an S3000L LSA process must be continuously compared to ensure the identification of required revaluation or adaptation. During operation, it becomes evident whether the predictions of the logistic analysis processes can bear comparison with the reality of the day-to-day experiences of the system operators. In-service data must be collected carefully and compared to the existing maintenance situation for permanent optimization of logistic support.

Specific data elements will be defined during the development of S5000F. At this time only the information clusters are addressed as they will be covered in S5000F:

- data for the analysis of defects, events and system/component health
- data for usage optimization
- data for integrated fleet management
- data relevant to consumption
- data relevant to engineering record cards
- data to support the management of performance based logistics contracts
- data to support LCC considerations
- data to support Product liability issues

7 Interrelation to SX000i

7.1 Purpose of SX000i

The SX000i activity started as an ASD/AIA initiative in September 2011 and is under development by a team of specialists from operators, manufacturers and regulatory authorities. The intention is to define the global ILS process and to identify the interfaces between the individual specifications of the ASD/AIA ILS Specification Suite. SX000i shall provide guidance about how to use the set of specifications when carrying out an ILS program.

7.2 S3000L/SX000i

LSA and ILS are related closely. The LSA process as described in this specification is the central tool to achieve the aim of an integrated development of the ILS products in close relation to the design of the Product to be supported. The strong interrelation between LSA as a central technical logistic process and ILS as a superior management process over the entire lifetime of complex and long living technical Products is described in SX000i.

Chapter 19

Data model

Table of contents

	Page
1 General	5
1.1 Introduction	5
1.2 Objective	5
1.3 Scope	5
1.4 Out of scope	5
2 Overview	6
2.1 Unified Modeling Language	6
2.1.1 UML Class model	6
2.1.2 Special guidance for reading the S3000L data model	15
2.2 Organization of the data model	16
2.3 S-Series Primitives (Data Types)	17
2.3.1 Overall description	17
2.3.2 Graphical representation	18
2.3.3 DateType	18
2.3.4 IdentifierType	19
2.3.5 DescriptorType	19
2.3.6 ClassificationType	19
2.3.7 PropertyType	20
2.4 Compound attributes	22
2.4.1 Overall description	22
2.4.2 Graphical representation	22
2.4.3 SerialNumberRange	22
2.4.4 DatedClassification	22
2.4.5 AuthorizedLife	22
3 Model overview	23
4 Functional areas	23
4.1 UoF Product and Project	23
4.1.1 Overall description	23
4.1.2 Graphical representation	24
4.1.3 UoF Product and Project - New class and interface definitions	24
4.2 UoF Product Usage Context	29
4.2.1 Overall description	29
4.2.2 Graphical representation	30
4.2.3 UoF Product Usage Context - New class and interface definitions	30
4.2.4 UoF Product Usage Context - Additions to referenced class and interface definitions	33
4.3 UoF Breakdown Structure	34
4.3.1 Overall description	34
4.3.2 Graphical representation	35
4.3.3 UoF Breakdown Structure - New class and interface definitions	35
4.3.4 UoF Breakdown Structure - Additions to referenced class and interface definitions	40
4.4 UoF Part Definition	40
4.4.1 Overall description	40
4.4.2 Graphical representation	41
4.4.3 UoF Part Definition - New class and interface definitions	41
4.5 UoF Breakdown Element Realization	47
4.5.1 Overall description	47
4.5.2 Graphical representation	48
4.5.3 UoF Breakdown Element Realization - New class and interface definitions	48

4.5.4	UoF Breakdown Element Realization - Additions to referenced class and interface definitions.....	51
4.6	UoF Breakdown Zone Element	52
4.6.1	Overall description	52
4.6.2	Graphical representation	52
4.6.3	UoF Breakdown Zone Element - New class and interface definitions	52
4.6.4	UoF Breakdown Zone Element - Additions to referenced class and interface definitions	53
4.7	UoF Breakdown Aggregated Element	54
4.7.1	Overall description	54
4.7.2	Graphical representation	54
4.7.3	UoF Breakdown Aggregated Element - New class and interface definitions	54
4.8	UoF Product Design Configuration.....	55
4.8.1	Overall description	55
4.8.2	Graphical representation	57
4.8.3	UoF Product Design Configuration - New class and interface definitions	57
4.8.4	UoF Product Design Configuration - Additions to referenced class and interface definitions.....	62
4.9	UoF LSA Candidate.....	63
4.9.1	Overall description	63
4.9.2	Graphical representation	64
4.9.3	UoF LSA Candidate - New class and interface definitions.....	65
4.9.4	UoF LSA Candidate - Additions to referenced class and interface definitions.....	77
4.10	UoF LSA Candidate Analysis Activity.....	77
4.10.1	Overall description	77
4.10.2	Graphical representation	78
4.10.3	UoF LSA Candidate Analysis Activity - New class and interface definitions	79
4.10.4	UoF LSA Candidate Analysis Activity - Additions to referenced class and interface definitions.....	87
4.11	UoF LSA FMEA	87
4.11.1	Overall description	87
4.11.2	Graphical representation	89
4.11.3	UoF LSA FMEA - New class and interface definitions	89
4.11.4	UoF LSA FMEA - Additions to referenced class and interface definitions	96
4.12	UoF Special Events and Damage	97
4.12.1	Overall description	97
4.12.2	Graphical representation	98
4.12.3	UoF Special Events and Damage - New class and interface definitions	98
4.12.4	UoF Special Events and Damage - Additions to referenced class and interface definitions.....	101
4.13	UoF LSA Candidate Task Requirement	102
4.13.1	Overall description	102
4.13.2	Graphical representation	103
4.13.3	UoF LSA Candidate Task Requirement - New class and interface definitions	103
4.13.4	UoF LSA Candidate Task Requirement - Additions to referenced class and interface definitions.....	106
4.14	UoF Task	107
4.14.1	Overall description	107
4.14.2	Graphical representation	108
4.14.3	UoF Task - New class and interface definitions	108
4.14.4	UoF Task - Additions to referenced class and interface definitions	119
4.15	UoF Task Resources	120
4.15.1	Overall description	120
4.15.2	Graphical representation	121
4.15.3	UoF Task Resources - New class and interface definitions	122
4.15.4	UoF Task Resources - Additions to referenced class and interface definitions	131
4.16	UoF Task Usage (Part 1).....	132

4.16.1	Overall description	132
4.16.2	Graphical representation	133
4.16.3	UoF Task Usage (Part 1) - New class and interface definitions	134
4.16.4	UoF Task Usage (Part 1) - Additions to referenced class and interface definitions....	144
4.17	UoF Task Usage (Part 2).....	145
4.17.1	Overall description	145
4.17.2	Graphical representation	146
4.17.3	UoF Task Usage (Part 2) - New class and interface definitions.....	146
4.17.4	UoF Task Usage (Part 2) - Additions to referenced class and interface definitions....	146
4.18	UoF Security Classification.....	146
4.18.1	Overall description	146
4.18.2	Graphical representation	147
4.18.3	UoF Security Classification - New class and interface definitions.....	147
4.18.4	UoF Security Classification - Additions to referenced class and interface definitions.	148
4.19	UoF Organization Assignment.....	149
4.19.1	Overall description	149
4.19.2	Graphical representation	149
4.19.3	UoF Organization Assignment - New class and interface definitions.....	149
4.19.4	UoF Organization Assignment - Additions to referenced class and interface definitions	151
4.20	UoF Document.....	154
4.20.1	Overall description	154
4.20.2	Graphical representation	154
4.20.3	UoF Document - New class and interface definitions.....	156
4.20.4	UoF Document - Additions to referenced class and interface definitions.....	162
4.21	UoF Remark	169
4.21.1	Overall description	169
4.21.2	Graphical representation	169
4.21.3	UoF Remark - New class and interface definitions	172
4.21.4	UoF Remark - Additions to referenced class and interface definitions	175
4.22	UoF Applicability Statement	184
4.22.1	Overall description	184
4.22.2	Graphical representation	185
4.22.3	UoF Applicability Statement - New class and interface definitions	187
4.22.4	UoF Applicability Statement - Additions to referenced class and interface definitions	197

List of tables

1	References	4
2	Association cardinalities used in S3000L	9

List of figures

1	Example of a class in UML	7
2	Example on relational representation of the Project class	7
3	Example of an attribute group in UML	8
4	Example of an association between classes in UML	9
5	Example on relational representation of the contractContext association	10
6	Example of an association with additional information in UML	10
7	Example on relational representation of an association with additional information....	10
8	Example of a directed association in UML	11
9	Example of specialization (inheritance) relationships between classes in UML	12
10	Example on relational representation for specializations	12

11	Example of a composition aggregation in UML.....	13
12	Example on relational representation of the composition aggregation	14
13	Example of Interface and Realize relationships in UML.....	15
14	Example on relational representation of interface definition	15
15	S3000L primitives (data types) - class model.....	18
16	S3000L Compound Attributes - class model	22
17	S3000L UoF overview	23
18	S3000L UoF Product and Project - class model	24
19	S3000L UoF Product Usage Context - class model.....	30
20	S3000L UoF Breakdown Structure - class model	35
21	S3000L UoF Part Definition- class model	41
22	S3000L UoF Breakdown Element Realization - class model.....	48
23	S3000L UoF Breakdown Zone Element - class model.....	52
24	S3000L UoF Breakdown Aggregated Element - class model	54
25	S3000L UoF Product Design Configuration - class model.....	57
26	S3000L UoF LSA Candidate - class model.....	64
27	S3000L UoF LSA Candidate Analysis Activity - class model	78
28	S3000L UoF LSA FMEA - class model	89
29	S3000L UoF Special Events and Damage - class model	98
30	S3000L UoF LSA Candidate Task Requirement - class model	103
31	S3000L UoF Task - class model	108
32	S3000L UoF Task Resources - class model	121
33	S3000L UoF Task Usage (Part 1) - class model.....	133
34	S3000L UoF Task Usage (Part 2) - class model.....	146
35	S3000L UoF Security Classification - class model.....	147
36	S3000L UoF Organization Assignment - class model.....	149
37	S3000L UoF Document - class model.....	155
38	S3000L UoF Document - document assignment items.....	156
39	S3000L UoF Remark - class model.....	170
40	S3000L UoF Remark - remark assignment items	171
41	S3000L UoF Applicability Statement - class model.....	186
42	S3000L UoF Applicability Statement - applicability assignment items.....	187

References

Table 1 References

Chap No./Document No.	Title
Chap 12	Maintenance task analysis
Chap 20	Data exchange
Chap 22	Data element list
GEIA-STD-0007	Logistics Product Data
ISO 10303:239 PLCS	Product Life Cycle Support
http://www.cax-if.org/documents/pdmug_release4_3.pdf	PDM Schema usage guide

www.oasis-open.org

Organization for the Advancement of Structured Information Standards

www.uml.org

Unified Modeling Language

www.plcs-resources.org

Product Life Cycle Support website

1 General

1.1 Introduction

This chapter defines a coherent data model for the data that can be exchanged between the Logistics Support Analysis (LSA) process as defined in S3000L, and related business processes.

The data model is described using the UML 2.0™ (Unified Modeling Language) class model diagrams (www.uml.org).

Each attribute defined in the S3000L data model is defined in [Chap 22](#), Data element list.

This data model is an extension of the Data Model and Exchange Working Group (DMEWG) Common Data Model issue 1.0 and is based upon information model defined in ISO 10303-239 Product Life Cycle Support (PLCS). This simplifies future implementation of PLCS Data EXchange specifications (DEXs) for actual data exchanges. Refer to [Chap 20](#).

The task related UoFs in S3000L were developed in cooperation with the S1000D Maintenance Data Task Team (MTDTT) in order to harmonize the S3000L data model with the task related schemas in S1000D.

1.2 Objective

The objective for this chapter is to define a coherent data model for the data that can be exchanged between the LSA process, and related business processes.

1.3 Scope

The following areas are within scope of the data model:

- Definition of the LSA program and the products that will be supported
- Support the early phases of the LSA program in terms of selecting the LSA candidate items and analysis activities that will be performed for each candidate item
- Support the LSA Failure Mode and Effect Analysis (FMEA), Special Events and Damage Analysis
- Document the results from the Maintenance and Operational Task Analysis (MTA/OTA) activities

1.4 Out of scope

The data model has been defined to support one LSA program at the time. Any implementation of the data model in a software package that will support multiple LSA programs, need to define how data can be reused between LSA programs as part of the software development.

The data model does not provide full support for all the identified candidate item analysis activities, but focuses on MTA/OTA, LSA FMEA Special Events and Damage Analysis. Results from other analysis activities, eg, Level Of Repair Analysis (LORA), can be referenced by summarized descriptions or by explicit document references.

The data model does not cover all data elements needed to support the LSA process, but just those data elements that will typically be exchanged between a contractor and his subcontractors, partners and customers.

2

Overview

2.1

Unified Modeling Language

The Unified Modeling Language™ (UML) is a widely used technique to model not only application structure, behavior, and architecture, but also business processes and data structures.

UML consists of a set of different modeling techniques of which this chapter only uses one, namely the UML class model.

2.1.1

UML Class model

Class models are the most widely used part of UML. Class models defines a static view of information (classes, attributes and relationships) needed to support the business processes.

This paragraph gives a short overview of the UML constructs used in the S3000L data model, in the style that is defined in the UML Writing Rules and Style Guide published by the ASD/AIA DMEWG.

Note

This chapter does not provide a complete description of UML class modeling techniques, but only those constructs that are relevant for the S3000L data model.

Each UML class model concept is also translated into a relational database example. These examples are provided for those readers who have an understanding of relational databases, but no previous knowledge of UML. Translations from UML to relational tables are only to be regarded as examples on how UML class model concepts can be represented using a relational database and must not be seen as the only solution for an implementation.

2.1.1.1

Class

The rectangle in a class diagram is called a classifier. The classifier gives the name of the class together with an enumeration of its attributes.

Note

Class names are written in UpperCamelCase, and attribute names are written in lowerCamelCase.

2.1.1.1.1

Attribute

Each attribute is presented with its attribute name, classification, data type and cardinality.

The classification of an attribute is shown within double angle brackets (<<...>>) above the attribute name.

Attributes that are part of the key (primary key) are classified as <<key>> attributes. Attributes that constitutes the key are always defined first in the attribute list for the Class.

Attributes that defines the characteristics of a given object (class instance) are classified as <<characteristic>>. Characteristics typically include measurable properties, classifications and descriptions.

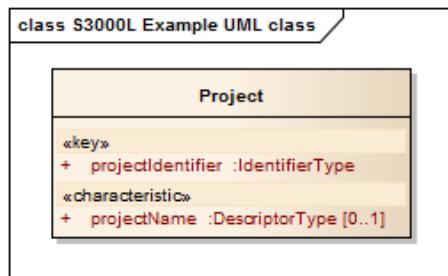
Metadata for a class instance provides information about one or more aspects of the class instance, such as a means of creation, author, time and date of creation. Metadata attributes are classified as <<metadata>>.

There are many cases where there is a need to provide additional information (metadata/characterization) for a given attribute value (eg, date when a classification was done, or the time when a property value was measured). These attributes are classified as <<characteristicMetadata>>. Characteristics metadata is always shown directly after the attribute to which it applies.

Data type and cardinality for an attribute is shown directly after the attribute name. Data types used in S3000L are described in detail in [Para 2.3](#). Cardinality for an attribute is defined in the same way as cardinality for an association as described in [Table 2](#). If an attribute has no explicit cardinality, it means that the attribute must have one value and one value only.

2.1.1.1.2 Class example

[Fig 1](#) shows an example of the class Project with its two attributes, projectIdentifier and projectName. Each attribute also shows its classification, data type and cardinality.



ICN-B6865-S3000L0200-002-00

Fig 1 Example of a class in UML

A class can have zero, one or many instances. Example on a project instance is the New Fighter Aircraft project.

A class in UML can be viewed as a table in a relational database, and an instance of that class can be seen as a row within that table, [Fig 2](#). The attributes of a class can be seen as the table columns.

PROJECT	
PROJECT_IDENTIFIER	PROJECT_NAME
P-123	New Fighter Aircraft

ICN-B6865-S3000L0201-002-00

Fig 2 Example on relational representation of the Project class

Note

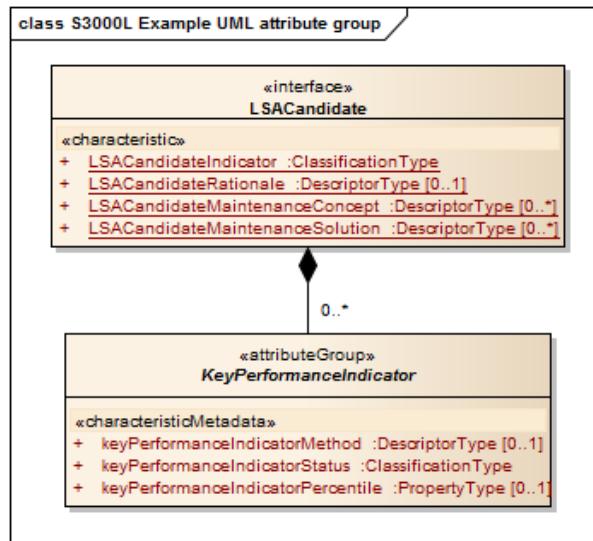
The relational table columns that constitutes its primary key is emphasized by underlining the column names.

A class can also have relationships to other classes in terms of association, composition, aggregation, generalization/specialization and realization.

2.1.1.1.3 Class attribute groups

A special case for defining attributes for a class is the use of attribute groups, [Fig 3](#). Attribute groups are typically used when the list of attributes defined for a class gets to extensive, and/or when there is some logic that is important to make explicit (eg, stakeholders such as 'Design' vs 'Commercial').

Attribute groups are illustrated using a class rectangle with the header <<attributeGroup>>. Note that attributes defined in an <<attributeGroup>> must be seen as any other attributes defined for the class it is associated with, and could have been defined within the parent class.



ICN-B6865-S3000L0263-001-00

Fig 3 Example of an attribute group in UML

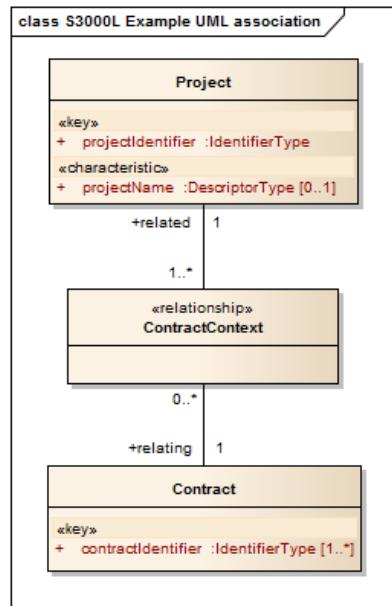
2.1.1.4 Abstract class

Classes which names are written in italic are defined as abstract classes in the data model. This means that this class cannot have any instances, but must always be instantiated by one of its specializations. Refer to [Para 2.1.1.5](#).

2.1.1.2 Association

Associations represent interdependencies between classes. Associations are often viewed as just another attribute where the attribute is modeled as a connector between the containing class and the class representing the attribute.

[Fig 4](#) shows that a contract can be associated with zero, one or many instances of project, and that a project must have at least one associated contract. The cardinality for the respective association is given at the end of the association.



ICN-B6865-S3000L0202-002-00

Fig 4 Example of an association between classes in UML

Note

One-to-many and many-to-many association relationships are defined using explicit `<<relationship>>` classes, instead of just using a simple association directly in between the related classes. The rationale for this is that many associations typically will have a set of characteristics that describes the association.

Table 2 Association cardinalities used in S3000L

Association	Explanation
1	One (and only one) instance of the associated class must be associated with each instance of the associating class (a mandatory association)
0..*	Zero, one or many instances of the associated class can be associated with each instance of the associating class (an optional association)
1..*	At least one instance of the associated class must be associated with each instance of the associating class (a mandatory association)

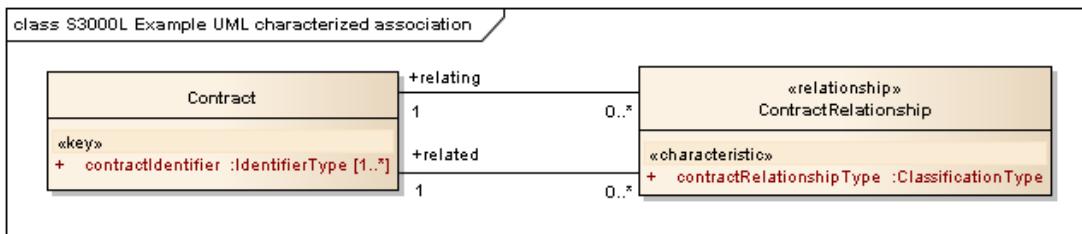
An association (including the `<<relationship>>` class) in UML can be viewed as a relationship table in a relational database where the columns in the table represent the primary keys from the respective associating and associated class. Table rows represent references to the associated instances, [Fig 5](#).

CONTRACT_CONTEXT	
<u>CONTRACT_IDENTIFIER</u>	<u>PROJECT_IDENTIFIER</u>
CCT-2008-011	P-123
CCT-2008-034	P-123

ICN-B6865-S3000L0203-002-00

Fig 5 Example on relational representation of the contractContext association

An example of an association (**<<relationship>>** class) that contains additional information (**<<characteristics>>**) about the association is the ContractRelationship **<<relationship>>** class. The ContractRelationship **<<relationship>>** class has an attribute contractRelationshipType which determines the type of association that is established between two contracts, [Fig 6](#).



ICN-B6865-S3000L0204-002-00

Fig 6 Example of an association with additional information in UML

Attributes that characterize a **<<relationship>>** class can be viewed as additional columns in a relationship table in a relational database, [Fig 7](#).

CONTRACT_RELATIONSHIP		
<u>CONTRACT_ID_1</u>	<u>CONTRACT_ID_2</u>	<u>REL_TYPE</u>
CCT-2008-011	CCT-2008-034	Subcontract

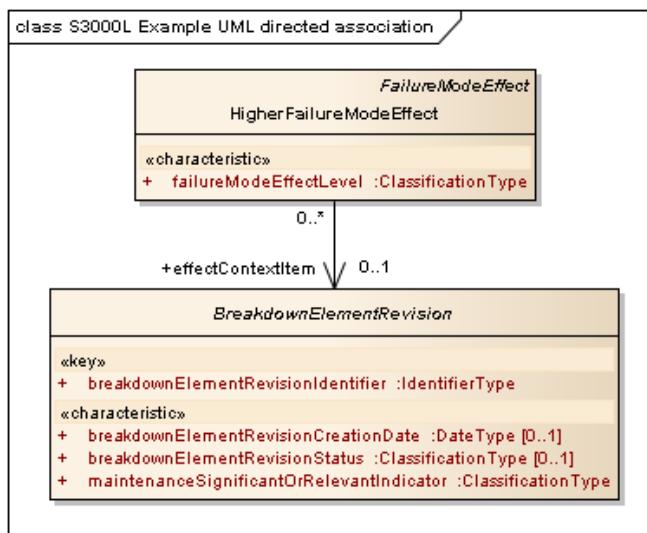
ICN-B6865-S3000L0205-002-00

Fig 7 Example on relational representation of an association with additional information

2.1.1.3 Directed association

A directed association has an open arrowhead at the target end of the association. In a directed association, two classes are related, but only one class knows that the relationship exists.

An example of a directed association, [Fig 8](#), is the possibility to associate a higher failure mode effect with a breakdown element that represents eg, the higher level function. However, there is no requirement for the breakdown element to be able to navigate to any associated higher failure mode effects.



ICR-B6865-S3000L0206-003-00

Fig 8 Example of a directed association in UML

Note

A directed association in UML can be viewed as a relationship table in a relational database, in the same way as described for an association.

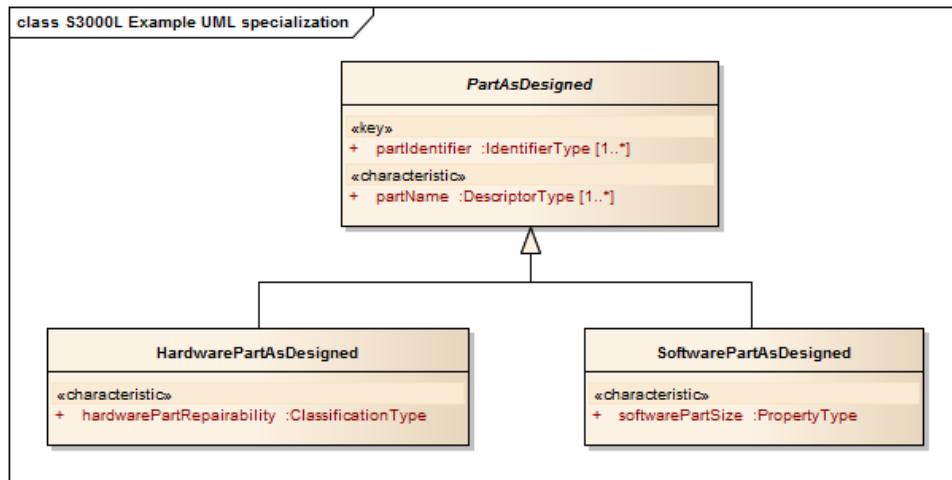
2.1.1.4 Generalization and specialization

Specialization defines an 'is-a' relationship. Specialization is modeled as a solid line connector with a hollow triangle at the end that points to the parent class (ie, the generalized class). Specialization is often referred to as inheritance.

The most important aspect of the generalization/specialization relationship is that the child class gets all the features of the parent class, and can then add features of its own, ie, a child class inherits all the attributes and relationships defined for its parent class.

Another important aspect of specialization is substitutability. Substitutability means that wherever a parent class is being used, it can be replaced by a child.

An example of the specialization/generalization association is PartAsDesigned and its specialization into HardwarePartAsDesigned and SoftwarePartAsDesigned, respectively. [Fig 9](#) shows an example where the PartAsDesigned class has two attributes, partIdentifier and partName. This means that both HardwarePartAsDesigned and SoftwarePartAsDesigned will have partIdentifier and partName as attributes, even though they are not explicitly enumerated in the attribute list for the respective class. The respective class only defines these additional attributes which are unique for the class.



ICN-B6865-S3000L0208-002-00

Fig 9 Example of specialization (inheritance) relationships between classes in UML

Note

A special characteristic that can be defined for the parent class is that it can be defined as an ‘abstract’ class. This means that the parent class itself cannot be instantiated, but must be substituted by any of its child classes. This is denoted in the model by making the class name (classifier) italic. *PartAsDesigned* in Fig 9 is defined as an abstract class.

Specialization can be viewed as multiple tables, Fig 10, with the same initial set of columns (incl. the primary key).

HARDWARE_PART		
PART_ID	PART_NAME	REPAIRABILITY
240-45-656654	Engine	Repairable

SOFTWARE_PART			
PART_ID	PART_NAME	SOFTWARE_SIZE_VALUE	SOFTWARE_SIZE_UOM
190-23-143244	Map	200	Megabyte

ICN-B6865-S3000L0209-003-00

Fig 10 Example on relational representation for specializations

Note

Both the softwarePartSize value and softwarePartSize Unit of Measure (UoM) are included in the PropertyType data type. Refer to [Para 2.3.7](#).

2.1.1.5

Aggregation

Aggregation defines whole and part relationships, ie, it indicates that one class is part of another class. In an aggregation relationship, the child class instances can outlive its parent class instance.

Note

An example of an aggregation relationship is car and its wheels, where the car represents the whole class and the wheels represents parts of the overall car. However, the wheel class instances can exist independently of the car class instance.

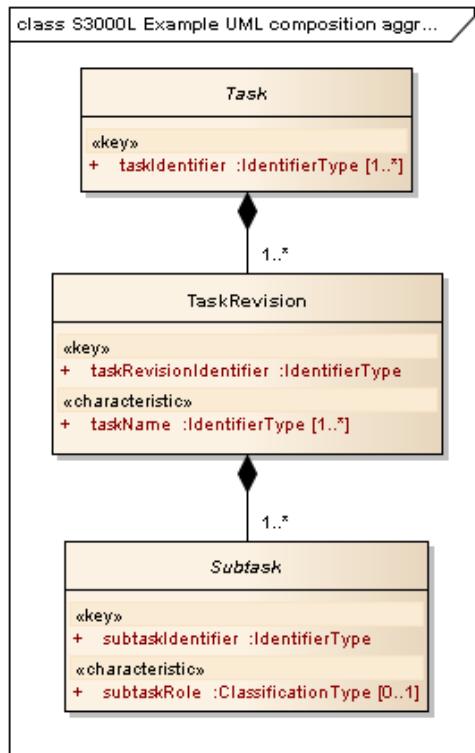
Note

Aggregation relationship is currently not used in the S3000L data model, but is described as background information for the composition aggregation defined in the next paragraph.

2.1.1.6 Composition aggregation

Composition aggregation is another form of an aggregation relationship, where the children are dependent upon the lifecycle for the parent class, ie, the child class instances cannot outlive its parent class instance.

An example of a composition aggregation, [Fig 11](#), is the possibility to define that a Task consists of one or many Subtasks (via its TaskRevision).



ICN-B6865-S3000L0210-003-00

Fig 11 Example of a composition aggregation in UML

Composition aggregation can be viewed as two relational tables, [Fig 12](#), where the relational table that represents the child class has a primary key which constitutes of both the primary key for the whole (composition) class, plus an additional key that distinguishes instances of the child class.

TASK		
TASK_ID	TASK_REV_ID	TASK_NAME
TSK-1112	1	Remove engine

SUBTASK			
TASK_ID	TASK_REV_ID	SUBTASK_ID	SUBTASK_ROLE
TSK-1112	1	STSK-1	Start-up
TSK-1112	1	STSK-2	Core

ICN-B6865-S3000L0211-002-00

Fig 12 Example on relational representation of the composition aggregation

2.1.1.7 Interface and the realize relationship

Interface is a way of modeling features that are common for a set of classes.

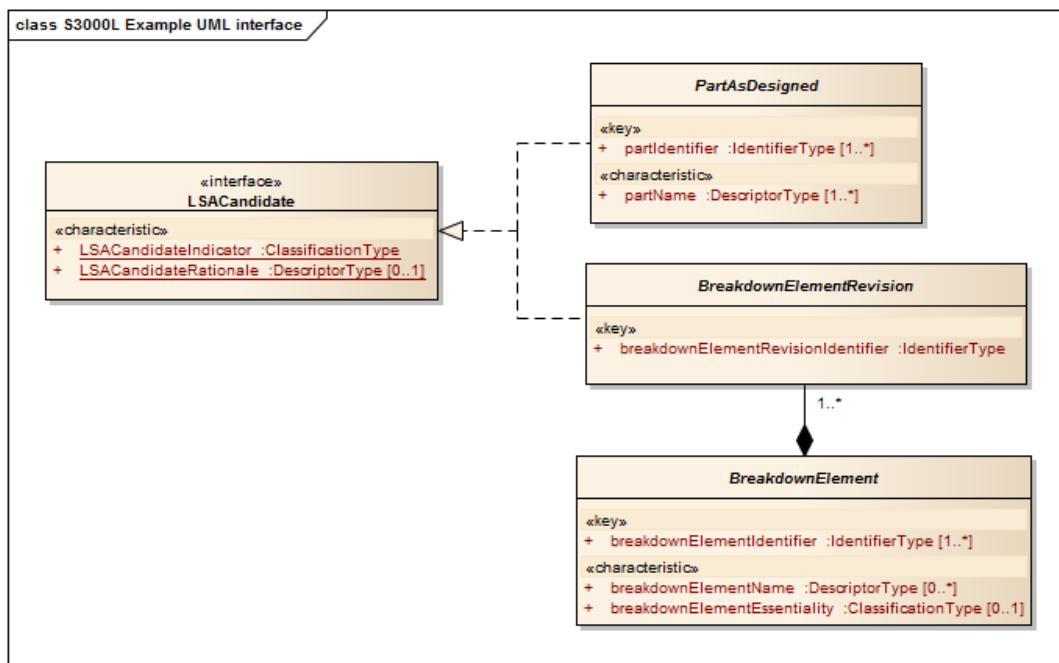
Any class that realizes (implements) an interface must support the features defined by the interface. These features include attributes, relationships and behavior.

An example of interface in the S3000L data model is the LSACandidate interface. This interface is realized by both BreakdownElementRevision and PartAsDesigned. This means that the attributes and relationships that are defined for the LSACandidate interface must be supported (implemented) by both PartAsDesigned and BreakdownElementRevision.

Note

Interfaces simplify the data model, eg, a similar relationship between one class and a set of other classes can be directed towards an interface which in turn is implemented by each class that can be part of that relationship.

The graphical representation, [Fig 13](#), for the realization connector is almost identical to the generalization connector, except that the connector is a dashed line instead of a solid line.



ICN-B6865-S3000L0212-003-00

Fig 13 Example of Interface and Realize relationships in UML

Interface definition can be viewed as adding columns to existing relational tables, ([Fig 14](#)) as well as adding relationship tables that will represent the interface associations.

BREAKDOWN_ELEMENT_REVISION			
BE_ID	BE_NAME	LSA_CAND_INDICATOR	LSA_CAND_RATIONALE
190-23-143244	Left engine	TRUE	Subject for replacement

PART			
PART_ID	PART_NAME	LSA_CAND_INDICATOR	LSA_CAND_RATIONALE
240-45-656654	Engine	TRUE	Subject for repair

ICN-B6865-S3000L0259-002-00

Fig 14 Example on relational representation of interface definition

2.1.2 Special guidance for reading the S3000L data model

2.1.2.1 Classes and attributes

Data types used in the S3000L class model are based on basic ISO 10303:239 Product Life Cycle Support constructs, and OASIS usage guidance on how to implement PLCS. Refer to [Para 2.3](#),

Each attribute used in the S3000L data model is defined in [Chap 22](#) Data element list.

2.1.2.2 Graphics

Graphical diagrams contain classes that are both filled and not filled (white). A filled class represents classes that are introduced and defined within the Unit of Functionality (UoF) under consideration. Classes without any fill (ie, white) are classes that are defined in other UoFs, and are always presented without any of its attributes.

2.1.2.3 Textual descriptions

The use of the terms "specific" and "instance of" in the textual descriptions means the same thing, ie, it refers to a specific instance of a class. For a description of what is meant by an instance, refer to [Para 2.1.1.1.2](#).

2.2 Organization of the data model

The S3000L data model is organized into a set of UoFs and divides the overall data model for S3000L into a set of smaller data models. Each UoF defines classes and attributes required to document a specific aspect of LSA.

Note

The set of UoFs does not follow the chapters of S3000L, but is structured in accordance with the order by which the data typically is created during the LSA process.

Each UoF contains:

- An overall description of the UoF
- A graphical representation of the class model for the UoF
- Definitions of new classes and interfaces identified within the UoF
- Additions to (extensions of) classes and interfaces defined within other UoFs

Each class defined in the data model has:

- A short description of its intent
- An enumeration of its attributes, including attribute cardinality
 - Where no cardinality is given, the attribute is mandatory and must only have one value
 - "zero or one", means that the attribute is optional but can only have one value if populated
 - "one or many", means that the attribute is mandatory and that it can have multiple values
 - "zero, one or many", means that the attribute is optional, but it can also have more than one value
- An enumeration of its associations
 - Associations described as "can" means that the association is optional
 - Associations described as "must" means that the association is mandatory
- An enumeration of implemented interfaces
 - Implemented interfaces mean that the class under consideration must support all the attributes and associations defined by the interface
- Special recommendations, such as
 - Special recommendations regarding the use of the attribute data types
 - Special recommendations regarding use of applicability

Note

Definitions for the attributes used in the data model are provided in [Chap 22](#) Data element list, and are not repeated in this chapter.

Note

Any reference to GEIA-STD-0007 is provided for informational purposes only, based on the fact that GEIA-STD-0007 (former MIL-STD-1388-2B) is widely known in the LSA community.

Note

Where no special recommendation is stated for an attribute of type `.PropertyType`, the default is to use the `ValueWithUnitPropertyType` class for the representation of its value.

2.3 S-Series Primitives (Data Types)

2.3.1 Overall description

Data types used in the S3000L, [Fig 15](#), data model are not the same as usually encountered in data models, eg, integer, real, string. The ASD/AIA DMEWG has defined a set of data types (primitives) that are closely related to the basic OASIS PLCS templates (www.oasis-open.org). These OASIS PLCS templates can be seen as small instantiation patterns, which ensure that each basic data element has an associated set of metadata.

For example, consider part identification (part number). In a traditional data model a part number would typically be represented as a string value. There is often, for example, no additional information about the organization that provided the part number.

In the OASIS PLCS template for assigning identifier, the string value representing the part number must always be defined together with the following additional metadata:

- Information about the type of part number that is being represented (eg, NATO Stock Number)
- Identification of the organization that determined ("owns") the part number
- Information on how the organization that owns the part number is identified (eg, NCAGE code)

The following <> primitives <> have been used throughout the S3000L data model:

- DateType
- IdentifierType
- DescriptorType
- ClassificationType
- PropertyType

Note

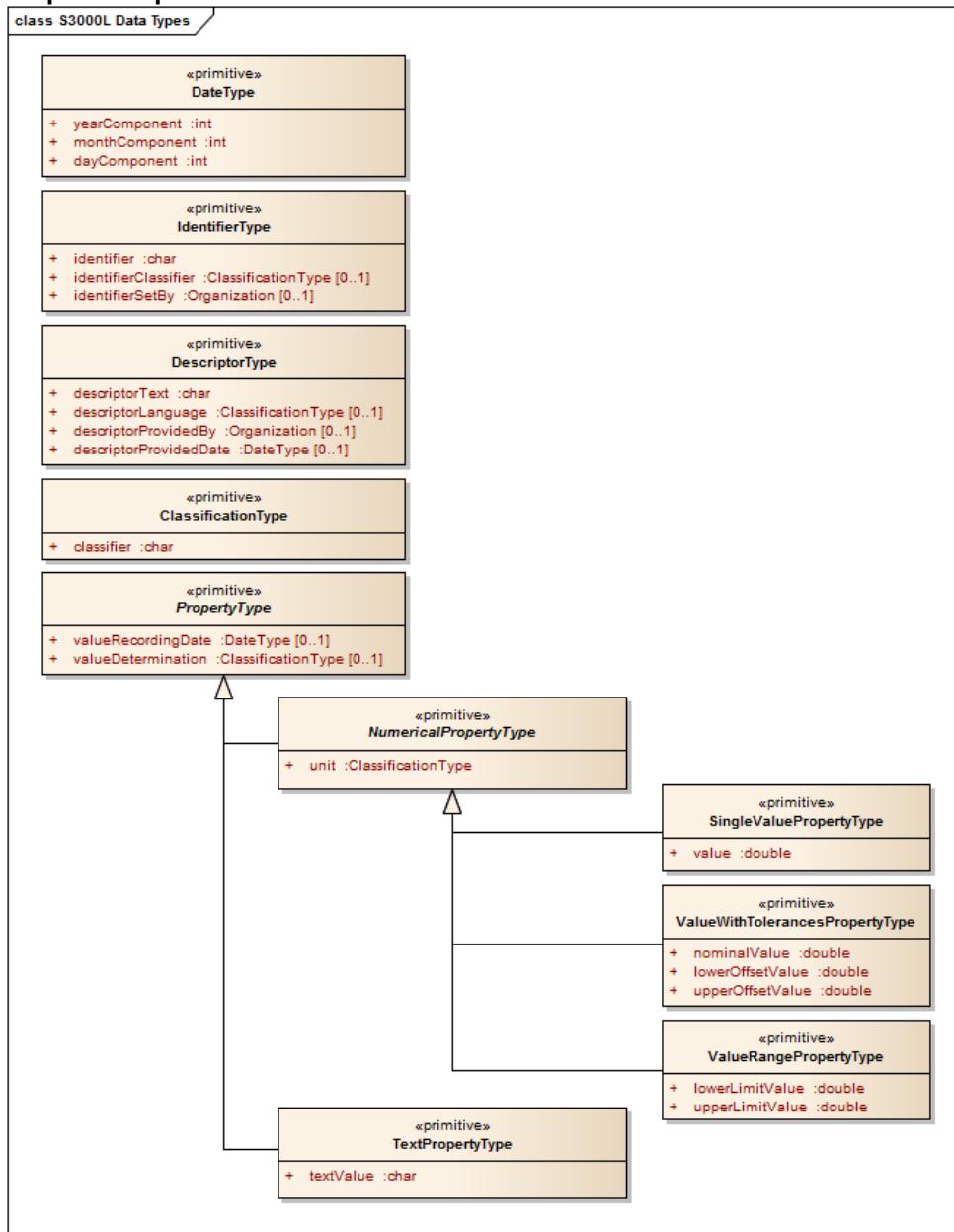
The Organization class is also used as a data type in some cases.

Note

There are also some exceptions in the S3000L data model where UML primitives have been used to define the attribute type for class attributes. UML primitives used are:

- boolean - true or false
- char - a sequence of characters, containing any character
- double - a signed double-precision floating-point number
- int - an element of the infinite set of integers (...,-3,-2,-1,0,1,2,3,...)

2.3.2 Graphical representation



ICN-B6865-S3000L0213-003-00

Fig 15 S3000L primitives (data types) - class model

2.3.3 **DateType**

The **DateType** <>primitive<> is used to represent calendar dates.

DateType attributes:

- `yearComponent`
- `monthComponent`
- `dayComponent`

Note

All **DateType** attributes are defined as integers. The rules for how to populate the respective value are defined in the data elements list.

Applicable to: All

S3000L-A-19-00-0000-00A-040A-A

Chap 19

Note

The DateType primitive does not include any predefined format, eg, 23/10/08 or 2008-10-23. Date format is avoided in order to simplify data exchange.

2.3.4**IdentifierType**

The IdentifierType <> primitive is used to represent any form of identification.

IdentifierType attributes:

- identifier
- identifierClassifier (zero or one)
- identifierSetBy (zero or one)

The identifier attribute contains the identifier string value as such (text string).

The identifierClassifier attribute determines the type of identifier that is provided.

The identifierSetBy identifies the organization that determined ("owns") the identifier.

Note

The identifierSetBy relates to an Organization as the data type. The Organization class is defined in UoF Project, and contains an organizationIdentifier (mandatory) and an organizationName (optional).

2.3.5**DescriptorType**

The DescriptorType <> primitive is used to represent any form of textual information (free form).

DescriptorType attributes.

- descriptor
- descriptorLanguage (zero or one)
- descriptorProvidedBy (zero or one)
- descriptorProvidedDate (zero or one)

The descriptor attribute contains text that provides further information about the object under consideration.

Note

The descriptor attribute must allow for text in an indented manner.

The descriptorProvidedBy attribute identifies the organization that provided the textual information. The descriptorProvidedDate identifies the date when the information was provided.

Note

The DescriptorType attributes descriptorLanguage, descriptorProvidedBy and descriptorProvidedDate enables eg, multiple descriptions to be provided for different purposes for the same object, and still be distinguished by organization and date.

2.3.6**ClassificationType**

The ClassificationType <> primitive is used for representing references to terms, which can be used for classification. The data model assumes that each term used has a definition in an external dictionary such as the ASD/AIA Dictionary, SX001D. This is also known as a Reference Data Library (RDL) in OASIS PLCS. For more information on the use of RDL refer to www.plcs-resources.org.

ClassificationType attributes:

- classifier

The classifier attribute contains the name of the term being used for classification.

Note

Classifications often appear as value lists (enumerations) in legacy applications.

Note

There is also a compound attribute type defined within S3000L for representing dated classifications (DatedClassification). This compound attribute type adds the possibility to define a date for when the classification was done.

2.3.7 PropertyType

2.3.7.1 PropertyType

The PropertyType <> primitive is used for representing property values and can also include additional information on when a property value was recorded and the method by which the value has been determined (eg, estimated, calculated).

.PropertyType attributes:

- valueRecordingDate (zero or one)
- valueDetermination (zero or one)

Note

The PropertyType class is an abstract class, ie, any of its subclasses must be used for the actual representation of a property value.

.PropertyType subclasses:

- Numerical.PropertyType (values with unit)
- Text.PropertyType (a property value described as a string, eg, "As-required").

Note

Where no explicit recommendations have been made for attributes that are of data type PropertyType in the S3000L data model, the default is that they use the SingleValue.PropertyType representation, ie, the property value is provided as a value together with its associated unit of measure.

2.3.7.2 Numerical.PropertyType

The Numerical.PropertyType <> primitive is used for representing numerical property values together with its associated unit of measure.

.PropertyType attributes:

- valueRecordingDate (inherited from the PropertyType <> primitive)
- valueDetermination (inherited from the PropertyType <> primitive)
- unit

Note

The Numerical.PropertyType class is an abstract class, ie, any of its subclasses must be used for the actual representation of a numerical property value.

.PropertyType subclasses:

- SingleValue.PropertyType (a property with a value and its UoM)
- ValueWithTolerances.PropertyType (a property with a value, its unit of measure, along with acceptable upper and lower offset values)
- ValueRange.PropertyType (a property with no fixed value, but defined by upper and lower limits along with a unit of measure)

2.3.7.3 SingleValue.PropertyType

The SingleValue.PropertyType <> primitive is used to represent a value with its unit of measure.

The SingleValue.PropertyType <> is a specialization of the Numerical.PropertyType <>.

Note

SingleValue.PropertyType is the default representation for all attributes that are of data type PropertyType.

SingleValue.PropertyType attributes:

- valueRecordingDate (inherited from the PropertyType <>)
- valueDetermination (inherited from the PropertyType <>)
- unit (inherited from the Numerical.PropertyType <>)
- value

2.3.7.4 ValueWithTolerances.PropertyType

The ValueWithTolerances.PropertyType <> is used to represent a value with its unit of measure, along with acceptable upper and lower offset values.

The ValueWithTolerances.PropertyType <> is a specialization of the Numerical.PropertyType <>.

ValueWithTolerances.PropertyType attributes:

- valueRecordingDate (inherited from the PropertyType <>)
- valueDetermination (inherited from the PropertyType <>)
- unit (inherited from the Numerical.PropertyType <>)
- nominalValue
- lowerOffsetValue
- upperOffsetValue

2.3.7.5 ValueRange.PropertyType

The ValueRange.PropertyType <> is used to represent a property with a value range in terms of upper and lower limits along with a unit of measure.

The ValueRange.PropertyType <> is a specialization of the Numerical.PropertyType <>.

ValueRange.PropertyType attributes:

- valueRecordingDate (inherited from the PropertyType <>)
- valueDetermination (inherited from the PropertyType <>)
- unit (inherited from the Numerical.PropertyType <>)
- lowerLimitValue
- upperLimitValue

2.3.7.6 TextProperty

The TextProperty <> is used to represent a property value described as a string value, eg, "As-required".

The TextProperty <> is a specialization of the PropertyType <>.

TextProperty attributes:

- valueRecordingDate (inherited from the PropertyType <>)
- valueDetermination (inherited from the PropertyType <>)
- textValue

2.4 Compound attributes

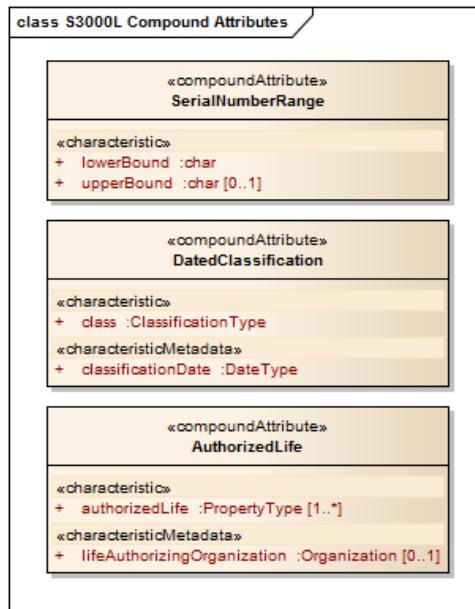
2.4.1 Overall description

Compound attributes defines often recurring patterns of attributes that are used together.

Compound attributes used in the S3000L data model, [Fig 16](#), are:

- SerialNumberRange
- DatedClassification
- AuthorizedLife

2.4.2 Graphical representation



ICN-B6865-S3000L0269-001-00

Fig 16 S3000L Compound Attributes - class model

2.4.3 SerialNumberRange

The SerialNumberRange <<compoundAttribute>> is used to identify a possibly open-ended interval of serialized items.

SerialNumberRange attributes:

- lowerBound
- upperBound (zero or one)

2.4.4 DatedClassification

The DatedClassification <<compoundAttribute>> is used to represent classifications where the date when the classification was done is as important as the classification itself.

DatedClassification attributes:

- class
- classificationDate

2.4.5 AuthorizedLife

The AuthorizedLife <<compoundAttribute>> is used to represent authorized life limits together with its authorizing organization.

AuthorizedLife attributes:

- authorizedLife (one or many)
- lifeAuthorizingOrganization (zero or one)

Note

Multiple values for authorizedLife mean ‘whichever is reached first’.

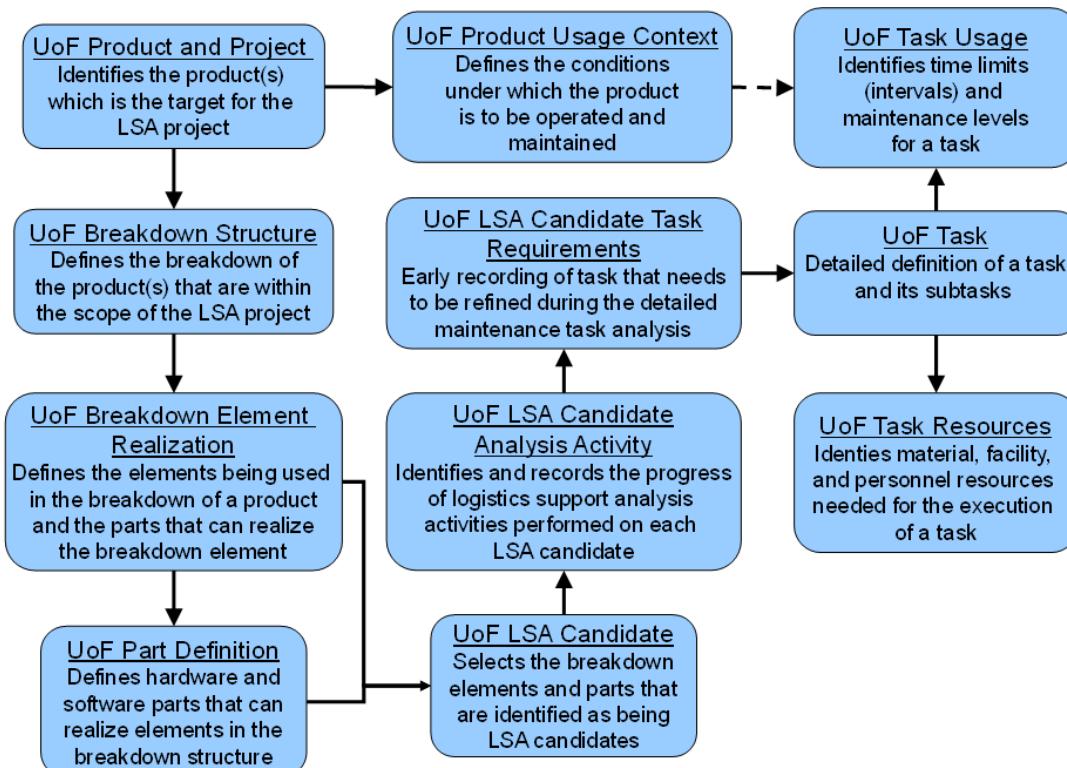
AuthorizedLife special recommendations:

- If an item has different authorized lives depending on eg, operational environments, these can be represented as a set of values for the authorizedLife attribute and be distinguished by the assignment of an ApplicabilityStatement to the respective instance of PropertyType that represents the individual value. Refer to [Para 4.22](#).

3 Model overview

The S3000L data model is organized into a set of UoFs, which splits the overall data model into a set of smaller data models. The purpose of this is to present small and coherent portions of the data model, and to gradually give the reader an understanding of the complete data model.

[Fig 17](#) provides an overview of the main UoFs, and how they are interrelated:



ICN-B6865-S3000L0214-003-00

Fig 17 S3000L UoF overview

4

Functional areas

4.1 4.1.1

UoF Product and Project

Overall description

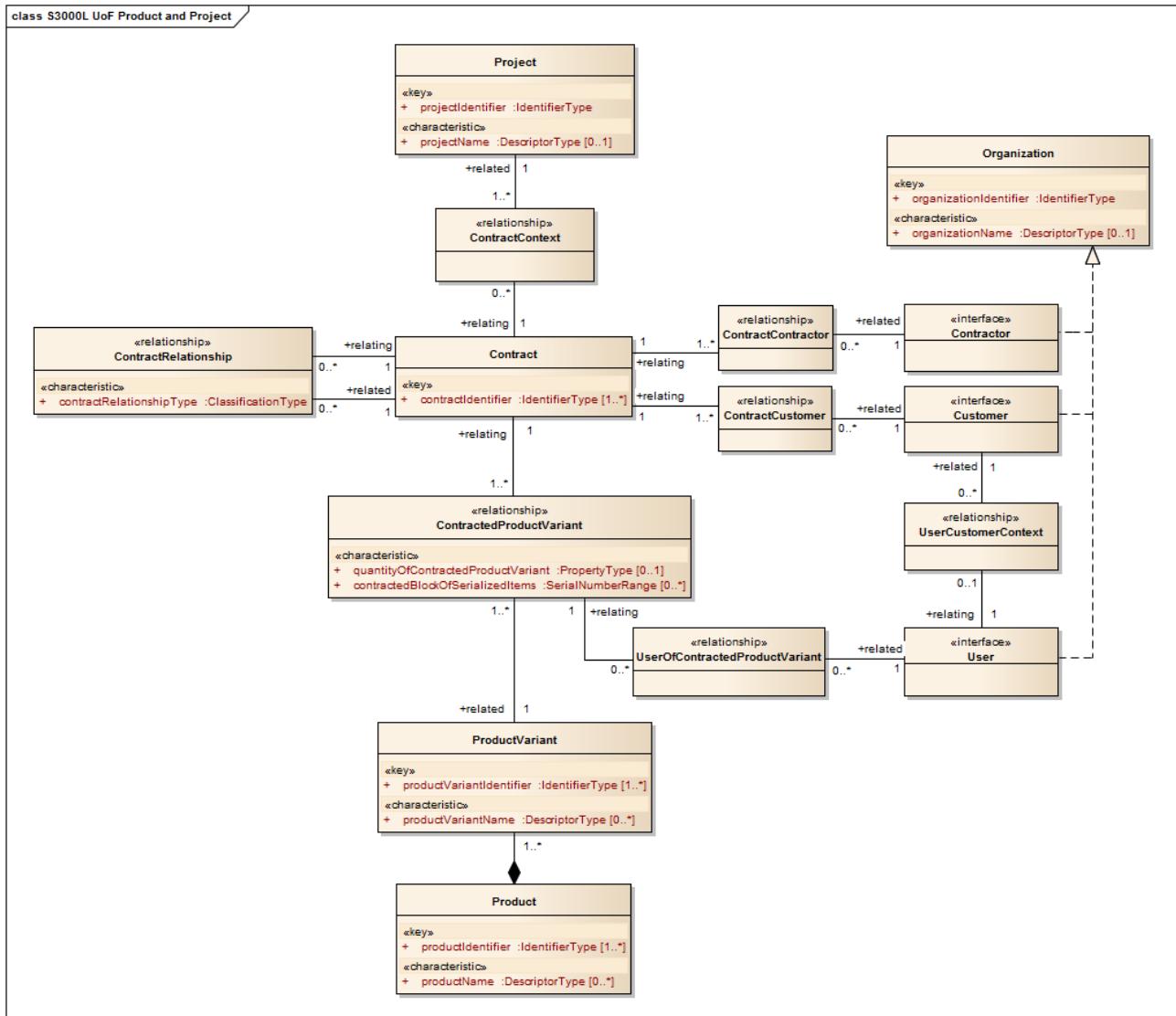
The first data elements defined during LSA are those required for defining the project (often referred to as the LSA program) and the Product in focus (often referred to as the end item).

A project can be related to one or many contracts, where contracts can be organized into a chain of associated contracts, eg, subcontracts, [Fig 17](#).

Each contract is associated with at least one customer and one contractor. Each contract includes one or many contracted product variants, where a contracted product variant relates to a specific variant of a generic product.

A contracted product variant can also include information on the quantity of contracted product variant and/or the serial number range, as known by the customer.

4.1.2 Graphical representation



ICN-B6865-S3000L0215-003-00

Fig 18 S3000L UoF Product and Project - class model

4.1.3 UoF Product and Project - New class and interface definitions

4.1.3.1 Project

The Project class identifies the overall LSA project (often also known as the LSA program).

Project attributes:

- projectIdentifier
- projectName (zero or one)

Project associations:

- Each instance of Project must be associated with at least one Contract (via the ContractContext <>relationship>> class)

4.1.3.2 ContractContext

The ContractContext class is a <>relationship>> class which defines the Project context for a Contract.

ContractContext associations:

- (relating) The Contract that has a related Project context
- (related) The Project that is the context for the relating Contract

Note

There is one instance of ContractContext per relevant combination of Contract and Project.

4.1.3.3 Contract

The Contract class identifies contracts that are relevant for the LSA program under consideration.

Contract attributes:

- contractIdentifier (one or many)

Contract associations:

- Each Contract can be associated with zero, one or many Projects (via the ContractContext <>relationship>> class)
- Each Contract can relate to zero, one or many other Contracts (via the ContractRelationship <>relationship>> class)
- Each Contract can be related to from zero, one or many other Contracts (via the ContractRelationship <>relationship>> class)
- Each instance of Contract must be associated with at least one contracted ProductVariant (via the ContractedProductVariant <>relationship>> class)
- Each instance of Contract must have at least one associated Customer (via the ContractCustomer <>relationship>> class)
- Each instance of Contract must have at least one associated Contractor (via the ContractContractor <>relationship>> class)

4.1.3.4 ContractRelationship

The ContractRelationship class is a <>relationship>> class, which defines relationships in between two related Contracts.

Note

An example of a contract relationship is subcontract.

ContractRelationship attributes:

- contractRelationshipType

ContractRelationship associations:

- (relating) The Contract that relates to another Contract
- (related) The Contract that is related to from the relating Contract

Note

There is one instance of ContractRelationship per relevant combination of relating Contract and related Contract.

4.1.3.5 Organization

The Organization class identifies organizations that can be referred to from the LSA program.

Organization attributes:

- organizationIdentifier
- organizationName (zero or one)

The Organization class implements the following interfaces, each representing a role that an organization can play in an LSA program:

- Contractor
- Customer
- User

Each of these interfaces has their own characteristics in addition to the characteristics that are generic for an Organization.

4.1.3.6 Contractor

The Contractor <> interface is implemented by classes that can play the role of a Contractor.

The class that implements the Contractor <> interface is:

- Organization

Classes that implement the Contractor <> interface must also implement the following association:

- An optional association with zero, one or many Contracts (via the ContractContractor <> relationship class).

4.1.3.7 ContractContractor

The ContractContractor <> relationship class defines relationships in between a Contract and the Contractor organization.

ContractContext associations:

- (relating) The Contract that has a defined Contractor
- (related) The Organization (via the Contractor <> interface) that is the contractor for the defined Contract

Note

There is one instance of ContractContractor per combination of Contract and Contractor.

4.1.3.8 Customer

The Customer <> interface is implemented by classes that can play the role of a Customer.

The class that implements the Customer <> interface is:

- Organization

Classes that implement the Customer <> interface must also implement the following associations:

- An optional association with zero, one or many Contracts (via the ContractCustomer <> relationship class)
- An optional association with zero, one or many instances of User (via the UserCustomerContext <> relationship class). This enables specific User organizations to be defined for a specific Customer.

4.1.3.9 ContractCustomer

The ContractCustomer <> relationship class defines relationships in between a Contract and the Customer organization.

ContractCustomer associations:

- (relating) The Contract that has a defined Customer
- (related) The Organization (via the Customer <> interface) that is the customer of the defined Contract

Note

There is one instance of ContractCustomer per combination of Contract and Customer.

4.1.3.10 User

The User <> interface is implemented by classes that can play the role of a user.

The class that implements the User <> interface is:

- Organization

Classes that implement the User <> interface must also implement the following associations:

- An optional association with zero, one or many ContractedProductVariants (via the UserOfContractedProductVariant <> relationship class)
- An optional association to an instance of Customer (via the UserCustomerContext <> relationship class), defining the Customer for which the User organization has been defined

4.1.3.11 UserCustomerContext

The UserCustomerContext <> relationship class defines relationships in between a User organization and the Customer organization for which the User organization has been defined.

UserCustomerContext associations:

- (relating) The Customer organization that has a defined User organization
- (related) The User organization that is defined for a specific Customer

Note

There is one instance of UserCustomerContext per combination of User and Customer organizations.

4.1.3.12 UserOfContractedProductVariant

The UserOfContractedProductVariant <> relationship class defines relationships in between a ContractedProductVariant and the Organization that is defined as being the User of the ContractedProductVariant under consideration.

UserOfContractedProductVariant associations:

- (relating) The ContractedProductVariant that has an associated User organization
- (related) The Organization (via the User <> interface) that is the user of the defined ContractedProductVariant

Note

There is one instance of UserOfContractedProductVariant per combination of ContractedProductVariant and User organization.

4.1.3.13 ContractedProductVariant

The ContractedProductVariant <> relationship class defines relationships in between a Contract and the ProductVariants that has been contracted. A contracted product variant can be restricted to one or many specific blocks of serialized items (using the

contractedBlockOfSerializedItems attribute), and/or to one or many specific user organizations (via the UserOfContractedProductVariant <>relationship> class).

ContractedProductVariantByQuantity attributes:

- quantityOfContractedProductVariant (zero or one)
- contractedBlockOfSerializedItems (zero, one or many).

ContractedProductVariant associations:

- (relating) The Contract for which the related ProductVariant has been contracted
- (related) The ProductVariant which has been contracted
- An optional association with zero, one or many User organizations (via the UserOfContractedProductVariant <>relationship> class)

4.1.3.14 Product

The Product class identifies a family of items sharing the same underlying design purpose. A Product must have at least one defined product variant.

Note

An example of generic product is F-15.

Product attributes:

- productIdentifier (one or many)
- productName (zero, one or many)

Note

An example of a type of productIdentifier is the GEIA-STD-0007 End Item Acronym Code.

Product associations:

- A Product must have at least one defined ProductVariant

4.1.3.15 ProductVariant

The ProductVariant class identifies a member of a Product family that is configured for a specific purpose and is offered to customers.

Note

A ProductVariant is also known as a Model in S1000D and S2000M.

Note

Examples of product variants are F-15A and F-15B

ProductVariant attributes:

- productVariantIdentifier (one or many)
- productVariantName (zero, one or many)

Note

An example of a type of productVariantIdentifier is the GEIA-STD-0007 System/End Item Usable On Code.

ProductVariant associations:

- Each ProductVariant must be of a defined Product
- A ProductVariant can be contracted by one or many Contracts (via the ContractedProductVariant <>relationship> class)

4.2 UoF Product Usage Context

4.2.1 Overall description

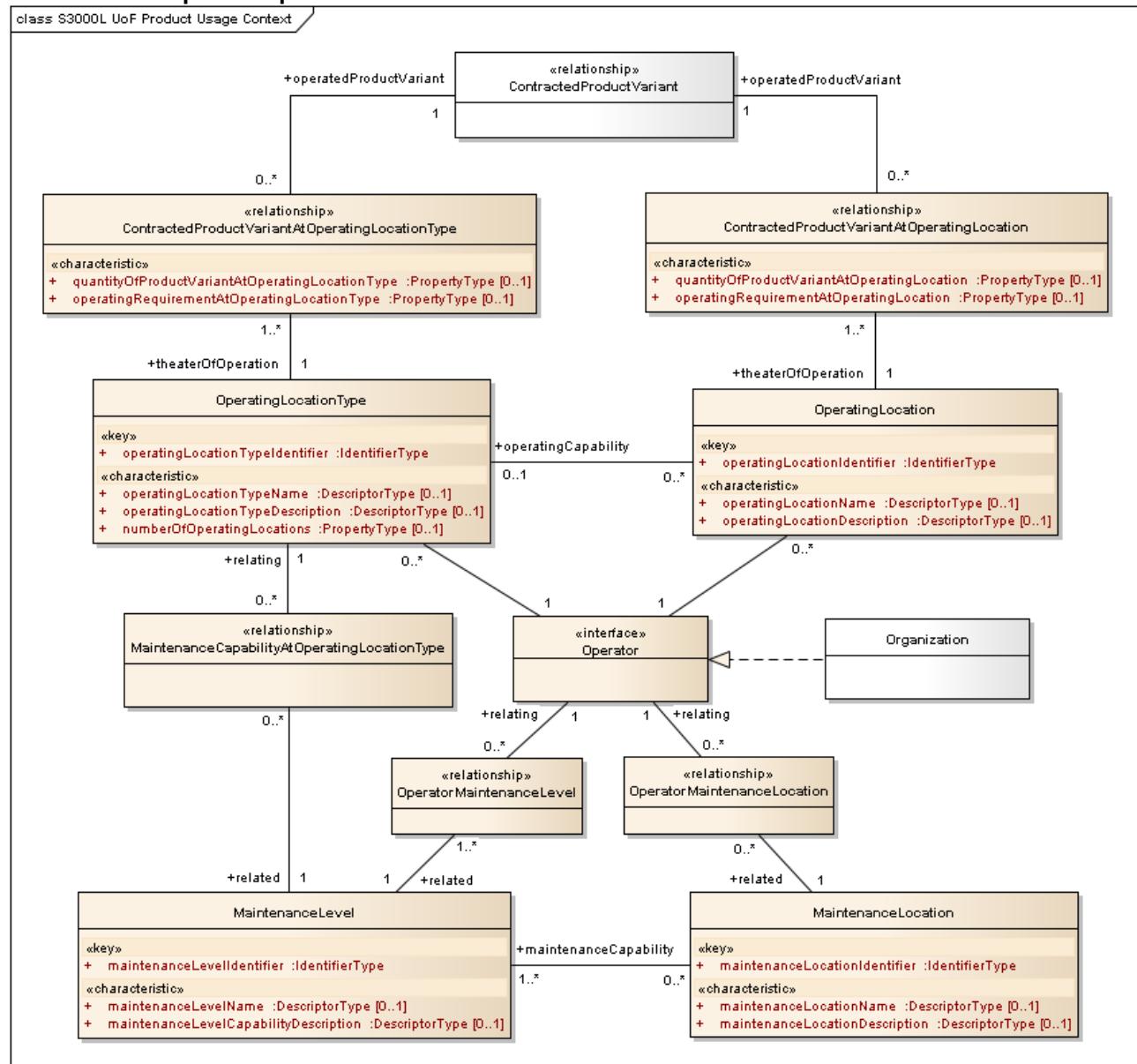
The Product Usage Context UoF, [Fig 19](#), defines the conditions under which the contracted product variant will be operated and maintained. This includes eg, maintenance levels and operating locations.

Note

MaintenanceLocation, OperatingLocationType and OperatingLocation classes are optional, and need not be defined within the LSA program.



4.2.2 Graphical representation



ICN-B6865-S3000L0216-002-00

Fig 19 S3000L UoF Product Usage Context - class model

4.2.3 UoF Product Usage Context - New class and interface definitions

4.2.3.1 Operator

The Operator <<interface>> must be implemented by classes that can be responsible for operating and maintaining the contracted product variant.

The class that implements the Operator <> interface is:

- #### – Organization

Classes that implement the Operator <> must also implement the following associations:

- An optional association with zero, one or many instances of OperatingLocationType
 - An optional association with zero, one or many instances of OperatingLocation

Applicable to: All

S3000L-A-19-00-0000-00A-040A-A

Chap 19

- An optional association with zero, one or many instances of MaintenanceLevel (via instances of the OperatorMaintenanceLevel <> relationship class)
- An optional association with zero, one or many instances of MaintenanceLocation (via instances of the OperatorMaintenanceLocation <> relationship class)

Note

Each OperatingLocationType and OperatingLocation is uniquely defined for a specific Operator.

4.2.3.2 MaintenanceLevel

Maintenance activities can be structured into maintenance levels according to operator maintenance policy, and organization of maintenance means.

MaintenanceLevel attributes:

- maintenanceLevelIdentifier
- maintenanceLevelName (zero or one)
- maintenanceLevelCapabilityDescription (zero or one)

MaintenanceLevel associations:

- A MaintenanceLevel can be defined for one or many Operators (via the OperatorMaintenanceLevel <> relationship class)
- A MaintenanceLevel can be located together with zero, one or many OperatingLocationTypes (defined via the MaintenanceCapabilityAtOperatingLocationType <> relationship class)
- Each MaintenanceLevel can be associated with zero, one or many MaintenanceLocations

4.2.3.3 MaintenanceLocation

The MaintenanceLocation class identifies actual locations where maintenance activities will be carried out.

MaintenanceLocation attributes:

- maintenanceLocationIdentifier
- maintenanceLocationName (zero or one)
- maintenanceLocationDescription (zero or one)

MaintenanceLocation associations:

- A MaintenanceLocation can be defined for one or many Operators (via the OperatorMaintenanceLocation <> relationship class)
- Each MaintenanceLocation can realize the capabilities of one or many defined maintenance levels.

4.2.3.4 OperatorMaintenanceLevel

The OperatorMaintenanceLevel <> relationship class defines relationships in between an Operator and the MaintenanceLevels which is part of the support solution for the contracted product variant.

OperatorMaintenanceLevel associations:

- (relating) The Operator that has a defined support solution which includes the related MaintenanceLevel
- (related) The MaintenanceLevel that is part of the Operators support solution.

Note

There is one instance of OperatorMaintenanceLevel per relevant combination of Operator and MaintenanceLevel.

4.2.3.5 OperatorMaintenanceLocation

The OperatorMaintenanceLocation <> relationship class defines relationships in between an Operator and actual MaintenanceLocations which is part of the support solution for the contracted product variant.

OperatorMaintenanceLocation associations:

- (relating) The Operator that has a defined support solution which includes the related MaintenanceLocation
- (related) The MaintenanceLocation that is part of the Operators support solution

Note

There is one instance of OperatorMaintenanceLocation per relevant combination of Operator and MaintenanceLocation.

4.2.3.6 OperatingLocationType

The OperatingLocationType class defines types of locations where the contracted product variant will be operated.

Note

The OperatingLocationType can be generic, ie, it does not need to define any specific characteristics, but just enables the recording of the number of contracted product variants and its operating requirements as defined within the ContractedProductVariantAtOperatingLocationType <> relationship class.

OperatingLocationType attributes:

- operatingLocationTypeIdentifier
- operatingLocationTypeName (zero or one)
- operatingLocationTypeDescription (zero or one)
- numberOfOperatingLocations (zero or one)

OperatingLocationType associations:

- An OperatingLocationType is uniquely defined for a specific Operator
- A defined OperatingLocationType must operate at least one ContractedProductVariant (via the ContractedProductVariantAtOperatingLocationType <> relationship class)
- An OperatingLocationType can be located together with zero, one or many MaintenanceLevels (via the MaintenanceCapabilityAtOperatingLocationType <> relationship class)
- An OperatingLocationType can be associated with zero, one or many specific OperatingLocations

4.2.3.7 OperatingLocation

The OperatingLocation class defines the exact location where the contracted product variant will be operated.

OperatingLocation attributes:

- operatingLocationIdentifier
- operatingLocationName (zero or one)
- operatingLocationDescription (zero or one)

OperatingLocation associations:

- An OperatingLocation is uniquely defined for a specific Operator
- A defined OperatingLocation must operate at least one ContractedProductVariant (via the ContractedProductVariantAtOperatingLocation <> relationship class)
- An OperatingLocation can be of a defined OperatingLocationType

- 4.2.3.8 **MaintenanceCapabilityAtOperatingLocationType**
The MaintenanceCapabilityAtOperatingLocationType <>relationship>> class defines relationships in between an OperatingLocationType and maintenance capabilities in terms of MaintenanceLevels that will be available at the operating location type.
- MaintenanceCapabilityAtOperatingLocationType associations:
- (relating) The OperatingLocationType that has a defined maintenance capability (MaintenanceLevel)
 - (related) The MaintenanceLevel capability that is available at the relating OperatingLocationType
- Note**
- There is one instance of MaintenanceCapabilityAtOperatingLocationType per relevant combination of OperatingLocationType and MaintenanceLevel.
- 4.2.3.9 **ContractedProductVariantAtOperatingLocationType**
The ContractedProductVariantAtOperatingLocationType <>relationship>> class adds information to the association in between a contracted product variant and its operating location type.
- ContractedProductVariantAtOperatingLocationType attributes:
- quantityOfProductVariantAtOperatingLocationType (zero or one)
 - operatingRequirementAtOperatingLocationType (zero or one)
- ContractedProductVariantAtOperatingLocationType associations:
- (theaterOfOperation) The OperatingLocationType where the relating ContractedProductVariant will be operated
 - (operatedProductVariant) The ContractedProductVariant that will be operated at the defined OperatingLocationType
- Note**
- There is one instance of ContractedProductVariantAtOperatingLocationType per relevant combination of ContractedProductVariant and OperatingLocationType.
- 4.2.3.10 **ContractedProductVariantAtOperatingLocation**
The ContractedProductVariantAtOperatingLocation <>relationship>> class adds information to the association in between a contracted product variant and its operating location.
- ContractedProductVariantAtOperatingLocation attributes:
- quantityOfProductVariantAtOperatingLocation (zero or one)
 - operatingRequirementAtOperatingLocation (zero or one)
- ContractedProductVariantAtOperatingLocation associations:
- (theaterOfOperation) The OperatingLocation where the relating ContractedProductVariant will be operated
 - (operatedProductVariant) The ContractedProductVariant that will be operated at the defined OperatingLocation
- 4.2.4 UoF Product Usage Context - Additions to referenced class and interface definitions**
- 4.2.4.1 **ContractedProductVariant**
The ContractedProductVariant <>relationship>> class defined in UoF Product and Project must also implement the following additional associations:
- An optional association with zero, one or many instances of OperatingLocationType where the ContractedProductVariant will be operated (via the ContractedProductVariantAtOperatingLocationType <>relationship>> class)

- An optional association with zero, one or many instances of OperatingLocation, where the ContractedProductVariant will be operated (via the ContractedProductVariantAtOperatingLocation <>relationship>> class).

4.2.4.2 Organization

The Organization class defined in UoF Product and Project must also implement the following additional <>interface>>:

- Operator

4.3 UoF Breakdown Structure

4.3.1 Overall description

The Breakdown Structure UoF, [Fig 20](#), can be used to define one or many types of breakdowns for a specific Product or for a ProductVariant. A breakdown can be eg, functional, physical, system, zonal, or hybrid (ie, a mixture of breakdown types).

Note

Existing breakdown structures are often hybrids consisting of both functional, physical and system elements.

Each breakdown can have one or many breakdown revisions (design iterations), where each breakdown revision references the breakdown element revisions that are included in the specific revision of the breakdown. These references are done using the BreakdownElementUsageInBreakdown class. References to the included breakdown elements (revisions) can then be organized hierarchically using the BreakdownElementStructure class.

Note

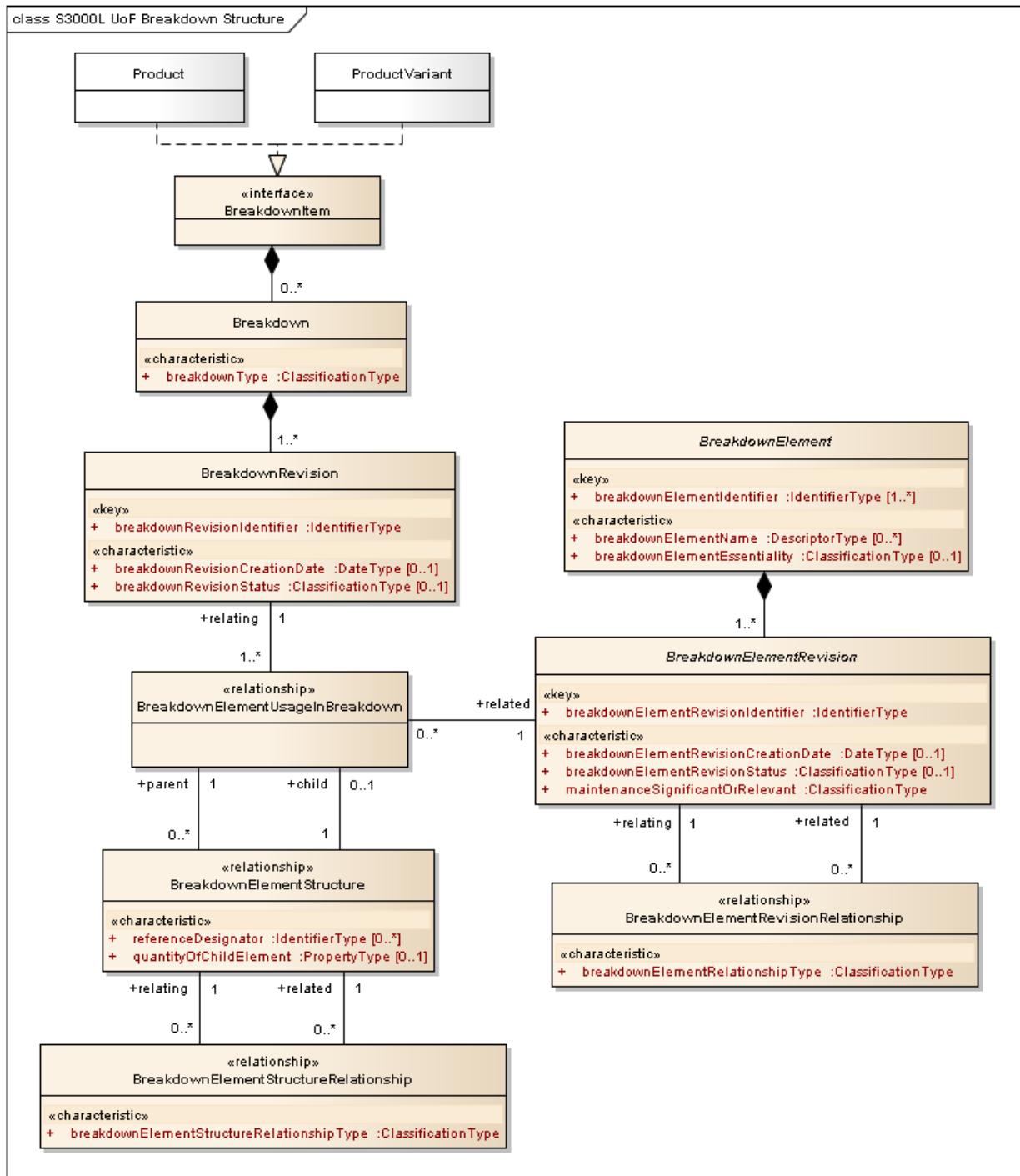
Having organized the breakdown element usages hierarchically by the use of BreakdownElementStructure, there needs to be one, and only one, instance of BreakdownElementUsageInBreakdown that has no parent element. This instance of BreakdownElementUsageInBreakdown is often referred to as the root node.

These breakdown principles allows for a defined breakdown element (revision) to be part of many breakdowns and revisions thereof. A breakdown element (revision) can be positioned differently in the respective breakdown and/or revision thereof. For example, a breakdown element (revision) can be used in both a functional and a physical breakdown.

Note

Each instance of BreakdownElementUsageInBreakdown and BreakdownElementStructure is uniquely defined for a specific breakdown revision.

4.3.2 Graphical representation



ICN-B6865-S3000L0217-003-00

Fig 20 S3000L UoF Breakdown Structure - class model

4.3.3 UoF Breakdown Structure - New class and interface definitions

4.3.3.1 Breakdown

The Breakdown class identifies a specific partitioning of a Product into a set of related elements so as to form parent-child views that comprise the product elements.

Applicable to: All

S3000L-A-19-00-0000-00A-040A-A

Chap 19

Examples of breakdowns are:

- Functional
- Physical
- System
- Zonal
- Hybrid, ie, a mixture of the functional, physical, system and zonal breakdowns

Breakdown attributes:

- breakdownType

Note

An example of a breakdown type is the GEIA-STD-0007 functional breakdown.

Breakdown associations:

- An association with the Product or ProductVariant for which the Breakdown is defined (via the BreakdownItem <>interface>>)
- An association with one or many BreakdownRevisions (design iterations) of the defined Breakdown

4.3.3.2 BreakdownItem

The BreakdownItem <>interface>> represents the common behavior of those items that can have one or many associated breakdown structures.

Classes that implement the BreakdownItem <>interface>> are:

- Product
- ProductVariant

Classes that implement the BreakdownItem <>interface>> must also implement the following association:

- An optional association with zero, one or many instances of Breakdown

Note

Breakdowns must be defined either on a Product level or on a ProductVariant level, but must never occur on both levels for one and the same Product.

4.3.3.3 BreakdownRevision

The BreakdownRevision class defines an explicit revision (design iteration) that is applied to a Breakdown.

Note:

BreakdownElementRevisions used in a breakdown are related to a BreakdownRevision and not directly to the Breakdown.

BreakdownRevision attributes:

- breakdownRevisionIdentifier
- breakdownRevisionCreationDate (zero or one)
- breakdownRevisionStatus (zero or one)

BreakdownRevision associations:

- A BreakdownRevision is always associated with one and only one Breakdown
- A BreakdownRevision is related to one or many BreakdownElementRevisions that are included in the breakdown revision under consideration (via the BreakdownElementUsageInBreakdown <>relationship>> class)

4.3.3.4 BreakdownElementUsageInBreakdown

The BreakdownElementUsageInBreakdown <> class is an association establishing the BreakdownElementRevisions (related) that belong to a BreakdownRevision (relating).

Note

Each instance of BreakdownElementUsageInBreakdown is uniquely defined for one combination of BreakdownElementRevision and BreakdownRevision. An instance of the BreakdownElementUsageInBreakdown class doesn't have any meaning by itself but only in the context of a BreakdownRevision.

BreakdownElementUsageInBreakdown associations:

- (relating) The BreakdownRevision in which the related BreakdownElementRevision is used
- (related) The BreakdownElementRevision that belong to the associated BreakdownRevision
- An optional parent association with zero, one or many children BreakdownElementUsageInBreakdown included in the same BreakdownRevision (via the BreakdownElementStructure <> class)
- An optional child association with zero or one parent BreakdownElementUsageInBreakdown included in the same BreakdownRevision (via the BreakdownElementStructure <> class)

Note

An instance of BreakdownElementUsageInBreakdown can be the parent of many instances of BreakdownElementUsageInBreakdown (ie, BreakdownElementRevisions) defined for the same BreakdownRevision.

Note

There is one instance of BreakdownElementStructure per parent-child relationship, and there is no explicit ordering in between the parent-child relationships defined for a specific BreakdownElementUsageInBreakdown (ie, ordering of child elements for a specific parent element).

4.3.3.5 BreakdownElementStructure

The BreakdownElementStructure <> class is an association that establishes a hierarchical structure between BreakdownElementRevisions (parent/child) that belong to the same BreakdownRevision.

Note

An explicit breakdown structure is accomplished by organizing the usages of the BreakdownElementRevisions, ie, by organizing the instances of BreakdownElementUsageInBreakdown instead of the organizing the BreakdownElementRevisions themselves.

Note

A breakdown structure can also be derived from the values given as breakdownElementIdentifiers. The breakdown structure is then implicit, rather than explicit and do not require the usage of the BreakdownElementStructure class. This approach can be used if the breakdownElementIdentifier follows the logic of eg, GEIA-STD-0007 Logistics Control Number (LCN) or S1000D Standard Numbering System (SNS). A discussion on the advantages/disadvantages of the respective approach is given in [Chap 4 Configuration Management](#).

BreakdownElementStructure attributes:

- referenceDesignator (zero, one or many)
- quantityOfChildElement (zero or one)

BreakdownElementStructure associations:

- (parent) The instance of BreakdownElementUsageInBreakdown that represents the BreakdownElementRevision which is defined as the parent element for the child element in the associated BreakdownRevision
- (child) The instance of BreakdownElementUsageInBreakdown that represents the BreakdownElementRevision which is defined as the child of the related parent element in the associated BreakdownRevision
- An optional association with zero, one or many instances of BreakdownElementStructure to which the BreakdownElementStructure under consideration has a defined dependency (via the BreakdownElementStructureRelationship <>relationship> class)
- An optional association with zero, one or many instances of BreakdownElementStructure which are dependent upon the BreakdownElementStructure under consideration (via the BreakdownElementStructureRelationship <>relationship> class)

Note

Both parent and child instances of BreakdownElementUsageInBreakdown must be defined for the same instance of BreakdownRevision.

4.3.3.6 BreakdownElementStructureRelationship

The BreakdownElementStructureRelationship <>relationship> class is an association where the usage of a BreakdownElementRevision (relating) is restricted or associated with the usage of another BreakdownElementRevision (related).

Note

Examples on information that can be represented using BreakdownElementStructureRelationship are:

- Substitute elements such as “real” vs “dummy”
- Element A cannot be installed at the same time as element B (“negative statement”)
- Element A must only be installed with element C (“positive statement”)

BreakdownElementStructureRelationship attributes:

- breakdownElementStructureRelationshipType

BreakdownElementStructureRelationship associations:

- (relating) The usage of a BreakdownElementRevision that defines a dependency to another BreakdownElementRevision used in the same BreakdownRevision
- (related) The usage of a BreakdownElementRevision upon which the relating BreakdownElementRevision usage is dependent

Note

Both instances of BreakdownElementStructure must relate to instances of BreakdownElementUsageInBreakdown that exist within the same BreakdownRevision.

4.3.3.7 BreakdownElement

The BreakdownElement class is used to represent items that are part of one or many Breakdowns and revisions thereof.

Note

BreakdownElement is defined as an abstract class, ie, this class will never be instantiated, but any of its subclasses will. Subclasses of BreakdownElement are:

- HardwareElement (defined in UoF Breakdown Element Realization). Refer to [Para 4.5](#).
- SoftwareElement (defined in UoF Breakdown Element Realization). Refer to [Para 4.5](#).
- ZoneElement (defined in UoF Breakdown Element Realization). Refer to [Para 4.5](#).
- AggregatedElement (defined in UoF Breakdown Aggregated Element). Refer to [Para 4.7](#).

BreakdownElement attributes:

- breakdownElementIdentifier (one or many)
- breakdownElementName (zero, one or many)
- breakdownElementEssentiality (zero or one)

BreakdownElement associations:

- An association with its BreakdownElementRevisions (design iterations)

4.3.3.8 BreakdownElementRevision

The BreakdownElementRevision class defines a revision (design iteration) that is applied to a BreakdownElement.

Note

New revisions of BreakdownElements are often defined during product design.

Note

BreakdownElementRevision is defined as an abstract class, ie, this class will never be instantiated but any of its subclasses will. Subclasses of BreakdownElementRevision are:

- HardwareElementRevision (defined in UoF Breakdown Element Realization). Refer to [Para 4.5](#).
- SoftwareElementRevision (defined in UoF Breakdown Element Realization). Refer to [Para 4.5](#).
- ZoneElementRevision (defined in UoF Breakdown Element Realization). Refer to [Para 4.5](#).
- AggregatedElementRevision (defined in UoF Breakdown Aggregated Element). Refer to [Para 4.7](#).
-

BreakdownElementRevision attributes:

- breakdownElementRevisionIdentifier
- breakdownElementRevisionCreationDate (zero or one)
- breakdownElementRevisionStatus (zero or one)
- maintenanceSignificantOrRelevant

BreakdownElementRevision associations:

- An association with the BreakdownElement of which the BreakdownElementRevision is a revision
- Associations with one or many instances of BreakdownElementUsageInBreakdown, ie, usages of the BreakdownElementRevision in BreakdownRevisions
- An instance of BreakdownElementRevision can relate to one or many other instances of BreakdownElementRevision via the BreakdownElementRevisionRelationship <> relationship class
- An instance of BreakdownElementRevision can be related to from one or many other instances of BreakdownElementRevision via the BreakdownElementRevisionRelationship <> relationship class

Note

A revision of a breakdown element can be used in one or many breakdowns via the BreakdownRevision and BreakdownElementUsageInBreakdown. This means that the same BreakdownElementRevision can be re-used in between different types of breakdowns for a single Product/ProductVariant, as well as be re-used in between Products/ProductVariants.

4.3.3.9 BreakdownElementRevisionRelationship

The BreakdownElementRevisionRelationship <> relationship class defines relationships in between two instances of BreakdownElementRevision.

Note

BreakdownElementRevisionRelationship can be used to represent the functional/physical breakdown element relationship as defined in GEIA-STD-0007.

Note

BreakdownElementRevisionRelationship can be used to represent, eg, use of software in hardware.

BreakdownElementRevisionRelationship attributes:

- breakdownElementRelationshipType

BreakdownElementRevisionRelationship associations:

- (relating) The instance of BreakdownElementRevision which relates to another instance of BreakdownElementRevision
- (related) The instance of BreakdownElementRevision that is being related to from another instance of BreakdownElementRevision

4.3.4 UoF Breakdown Structure - Additions to referenced class and interface definitions**4.3.4.1 Product**

The Product class defined in UoF Product and Project must also implement the following additional <>interface>>:

- BreakdownItem

4.3.4.2 ProductVariant

The ProductVariant class defined in UoF Product and Project must also implement the following additional <>interface>>:

- BreakdownItem

4.4 UoF Part Definition**4.4.1 Overall description**

The UoF Part Definition, [Fig 21](#), is used to represent hardware items as well as software.

PartAsDesigned in S3000L represents an abstract class, ie, an instance of a PartAsDesigned must be either a HardwarePartAsDesigned or a SoftwarePartAsDesigned.

PartAsDesigned can be used to realize breakdown elements in a breakdown of a product, as well as play the role of a resource in a maintenance task.

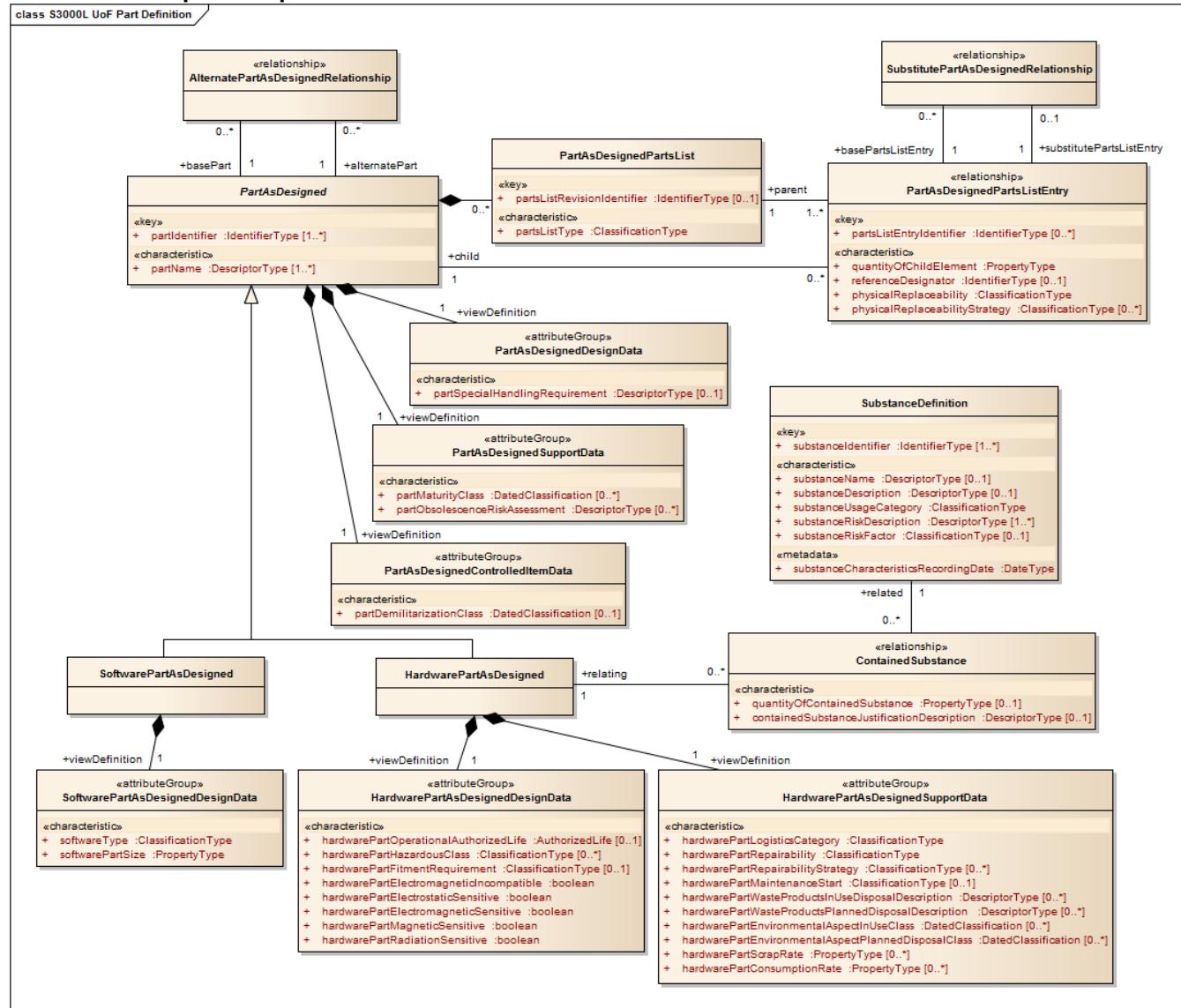
A part can have one or more alternate parts. The alternate part is interchangeable with the base part in all its usages.

PartAsDesigned can be organized into BOM using the PartAsDesignedPartsList class, ie, parts can consist of other parts.

A part used in a BOM can have one or many substitute parts. As opposed to the alternate part, which is context free, this substitute part is context dependent. The substitute part is only valid in the context of its usage as a component within the specified BOM.

Substances (often hazardous) are defined separately, and are then associated with the parts in which they are contained.

4.4.2 Graphical representation



ICN-B6865-S3000L0218-003-00

Fig 21 S3000L UoF Part Definition- class model

4.4.3 UoF Part Definition - New class and interface definitions

4.4.3.1 PartAsDesigned

PartAsDesigned is a discrete object that can come into existence as a consequence of a manufacturing process (*Source: ISO 10303:239 Product Life Cycle Support*).

A PartAsDesigned can be at any indenture level in a part structure, a PartAsDesigned can be eg, an assembly as well as an equipment, component or software package.

Note

PartAsDesigned is an abstract class, ie, an instantiation of PartAsDesigned must always be either a HardwarePartAsDesigned or a SoftwarePartAsDesigned.

PartAsDesigned attributes:

- partIdentifier (one or many)
- partName (one or many)

Applicable to: All

S3000L-A-19-00-0000-00A-040A-A
Chap 19

- PartAsDesignedDesignData (<<attributeGroup>>)
 - partSpecialHandlingRequirement (zero or one)
- PartAsDesignedSupportData (<<attributeGroup>>)
 - partMaturityClass (zero, one or many)
 - partObsolescenceRiskAssessment (zero, one or many)
- PartAsDesignedControlledItemData (<<attributeGroup>>)
 - partDemilitarizationClass (zero or one)

Note

Examples of different partIdentifiers are Original Equipment Manufacturer (OEM) part numbers and Supplier part numbers.

Note

PartAsDesignedDesignData (<<attributeGroup>>) is the definitional information which defines characteristics for the PartAsDesigned identified during the design activities.

Note

PartAsDesignedSupportData (<<attributeGroup>>) is the definitional information which defines characteristics for the PartAsDesigned identified during the supportability analysis activities.

Note

PartAsDesignedControlledItemData (<<attributeGroup>>) is the definitional information which defines characteristics for the PartAsDesigned identified during different control analysis activities.

Note

A PartAsDesigned can have more than one part maturity classification (partMaturityClass). This will most likely reflect changes in part maturity over time. If the classificationDate attribute included in the DatedClassification compound attribute is not enough to determine the validity of the respective partMaturityClass value, ApplicabilityStatement can be used to further determine the applicability of the respective value. Refer to [Para 4.22](#),

Note

A PartAsDesigned can have more than one partObsolescenceRiskAssessment description. This will most likely reflect changes in part obsolescence risk assessment over time. If the descriptorProvidedDate attribute included in the DescriptorType is not enough to determine the validity of the respective partObsolescenceRiskAssessment description, ApplicabilityStatement can be used to further determine the applicability of the respective description. Refer to [Para 4.22](#),

PartAsDesigned associations:

- A PartAsDesigned can have zero, one or many types of PartAsDesignedPartsList (BOM) and revisions thereof
- A PartAsDesigned can be included in zero, one or many other parts (as a child of one or many instances of the PartAsDesignedPartsListEntry <<relationship>> class)
- A PartAsDesigned can refer to zero, one or many alternate parts (via the basePart association with the AlternatePartAsDesignedRelationship class)
- A PartAsDesigned can be an alternate part for one or many other parts (defined via the alternatePart association with the AlternatePartAsDesignedRelationship class)

4.4.3.2 AlternatePartAsDesignedRelationship

The AlternatePartAsDesignedRelationship <<relationship>> class is used to define alternate parts. An alternate part is interchangeable with the base part in all its usages. The alternate part

is a context independent alternate that is form, fit, and function equivalent in any use (*Source: ISO 10303:239 Product Life Cycle Support*).

Note

AlternatePartAsDesignedRelationship defines one alternate PartAsDesigned at the time, and applies both to HardwarePartAsDesigned and SoftwarePartAsDesigned.

AlternatePartAsDesignedRelationship associations:

- (basePart) The PartAsDesigned for which an alternate is specified
- (alternatePart) The PartAsDesigned that can replace the base PartAsDesigned in all its usages

4.4.3.3 PartAsDesignedPartsList

The PartAsDesignedPartsList class represents a Bill-Of-Material for a given PartAsDesigned.

PartAsDesignedPartsList attributes:

- partsListRevisionIdentifier
- partsListType

PartAsDesignedPartsList associations:

- An association with the PartAsDesigned for which the PartAsDesignedPartsList is defined
- Associations with one or many instances of PartAsDesignedPartsListEntry, ie, references to one or many PartAsDesigned that are included in the PartAsDesignedPartsList

4.4.3.4 PartAsDesignedPartsListEntry

The PartAsDesignedPartsListEntry <>relationship>> class is being used to define the content in a PartAsDesignedPartsList.

Note

PartAsDesignedPartsListEntry defines one “row” at the time in a PartAsDesignedPartsList (Bill-of-Material), ie, there is one instance of PartAsDesignedPartsListEntry per contained part.

Note

The PartAsDesignedPartsListEntry applies both to HardwarePartAsDesigned and SoftwarePartAsDesigned, and a BOM can consist of both HardwarePartAsDesigned and SoftwarePartAsDesigned.

PartAsDesignedPartsListEntry attributes:

- partsListEntryIdentifier (zero, one or many)
- quantityOfChildElement
- referenceDesignator (zero, one or many)
- physicalReplaceability
- physicalReplaceabilityStrategy (zero, one or many)

Note

Physical replaceability and replaceability strategy in the context of a PartAsDesignedPartsListEntry are used to define if the child part can be replaced in its parent part and if so at what maintenance level.

PartAsDesignedPartsListEntry associations:

- (parent) An association with the PartAsDesignedPartsList for which the PartAsDesignedPartsListEntry is defined
- (child) An association with the PartAsDesigned that is included in the PartAsDesignedPartsList

- An association with zero, one or many instances PartAsDesignedPartsListEntry that specifies substitute PartAsDesigned for a specific PartAsDesignedPartsListEntry (defined via the SubstitutePartAsDesignedRelationship <>relationship>> class)
- An association with zero or one base PartAsDesignedPartsListEntry instances, for which the PartAsDesignedPartsListEntry under consideration can be a substitute (defined via the SubstitutePartAsDesignedRelationship <>relationship>> class)

PartAsDesignedPartsListEntry special recommendations:

- A PartAsDesigned included in another PartAsDesigned can have multiple replaceability strategies (ie, values of the attribute physicalReplaceabilityStrategy). Each physicalReplaceabilityStrategy value (ie instance of data type ClassificationType) must then have an associated ApplicabilityStatement, as defined in the UoF Applicability Statement, in order to define when the respective physicalReplaceabilityStrategy is applicable. The Applicability Statement will in most cases include Operator as one of the parameters to be evaluated.

4.4.3.5 SubstitutePartAsDesignedRelationship

The SubstitutePartAsDesignedRelationship <>relationship>> class is used to define substitute parts. As opposed to the alternate part, which is context free, substitute part is context dependent, ie, the SubstitutePartAsDesignedRelationship is a relationship that indicates that one PartAsDesignedPartsListEntry instance can be substituted by another instance of PartAsDesignedPartsListEntry, where both instances of PartAsDesignedPartsListEntry refers to the same parent PartAsDesignedPartsList (*Source: ISO 10303:239 Product Life Cycle Support*).

Note

SubstitutePartAsDesignedRelationship defines one substitute part at the time, and applies both to HardwarePartAsDesigned and SoftwarePartAsDesigned.

SubstitutePartAsDesignedRelationship associations:

- (basePartsListEntry) The PartAsDesignedPartsListEntry for which a substitute is specified
- (substitutePartsListEntry) The PartAsDesignedPartsListEntry which can replace the base PartAsDesignedPartsListEntry

4.4.3.6 HardwarePartAsDesigned

The HardwarePartAsDesigned class is a specialization of class PartAsDesigned, and represents parts that will be realized as physical items (hardware), including non-countable material.

HardwarePartAsDesigned attributes:

- partIdentifier (inherited from class PartAsDesigned)
- partName (inherited from class PartAsDesigned)
 - HardwarePartAsDesignedDesignData (<>attributeGroup>>)
- partSpecialHandlingRequirement (inherited from class PartAsDesigned PartAsDesignedDesignData <>attributeGroup>>)
- hardwarePartOperationalAuthorizedLife (zero or one)
- hardwarePartHazardousClass (zero, one or many)
- hardwarePartFitmentRequirement (zero or one)
- hardwarePartElectromagneticIncompatible
- hardwarePartElectrostaticSensitive
- hardwarePartElectromagneticSensitive
- hardwarePartMagneticSensitive
- hardwarePartRadiationSensitive

HardwarePartAsDesignedSupportData (<>attributeGroup>>)

- partMaturityClass (inherited from class PartAsDesigned PartAsDesignedSupportData <<attributeGroup>>)
- partObsolescenceRiskAssessment (inherited from class PartAsDesigned PartAsDesignedSupportData <<attributeGroup>>)
- hardwarePartLogisticsCategory
- hardwarePartRepairability
- hardwarePartRepairabilityStrategy (zero, one or many)
- hardwarePartMaintenanceStart (zero or one)
- hardwarePartWasteProductsInUseDisposalDescription (zero, one or many)
- hardwarePartWasteProductsPlannedDisposalDescription (zero, one or many)
- hardwarePartEnvironmentalAspectInUseClass (zero, one or many)
- hardwarePartEnvironmentalAspectPlannedDisposalClass (zero, one or many)
- hardwarePartScrapRate (zero or one)
- hardwarePartConsumptionRate (zero, one or many)
 HardwarePartAsDesignedControlledItemData (<<attributeGroup>>)
- partDemilitarizationClass (inherited from class PartAsDesigned PartAsDesignedControlledItemData <<attributeGroup>>)

Note

HardwarePartAsDesignedDesignData (<<attributeGroup>>) is the definitional information which defines characteristics for the HardwarePartAsDesigned identified during the design activities.

Note

HardwarePartAsDesignedSupportData (<<attributeGroup>>) is the definitional information which defines characteristics for the HardwarePartAsDesigned identified during the supportability analysis activities.

Note

HardwarePartAsDesignedControlledItemData (<<attributeGroup>>) is the definitional information which defines characteristics for the HardwarePartAsDesigned identified during different control analysis activities.

Note

A HardwarePartAsDesigned can have more than one hardwarePartWasteProductsInUseDisposalDescription and/or hardwarePartWasteProductsPlannedDisposalDescription. This will most likely reflect changes in part waste products disposal descriptions over time.

Note

A HardwarePartAsDesigned can have more than one hardwarePartEnvironmentalAspectInUseClass and/or hardwarePartEnvironmentalAspectPlannedDisposalClass classification. This will most likely reflect changes in hardware part environmental aspects over time.

HardwarePartAsDesigned associations:

- A HardwarePartAsDesigned can have zero, one or many types of PartAsDesignedPartsList and revisions thereof (inherited from class PartAsDesigned)
- A HardwarePartAsDesigned can be included in zero, one or many other parts, as a child of one or many instances of the PartAsDesignedPartsListEntry <<relationship>> class (inherited from class PartAsDesigned)
- A HardwarePartAsDesigned can refer to zero, one or many alternate HardwarePartAsDesigned (inherited from class PartAsDesigned)
- A HardwarePartAsDesigned can be an alternate part for one or many other HardwarePartAsDesigned (inherited from class PartAsDesigned)

- A HardwarePartAsDesigned can have zero, one or many associated substances, which are contained in the HardwarePartAsDesigned (via the ContainedSubstance <>relationship>> class)

HardwarePartAsDesigned special recommendations:

- A HardwarePartAsDesigned can have multiple reparability strategies (ie, values for hardwarePartRepairabilityStrategy). Each hardwarePartRepairabilityStrategy value (ie, instance of the ClassificationType data type) must then have an associated ApplicabilityStatement, as defined in the UoF Applicability Statement, in order to define when respective reparability strategy is applicable. The applicability statement will in most cases include Operator as one of the parameters to be evaluated.
- A HardwarePartAsDesigned can have multiple consumption rates (ie, values for hardwarePartConsumptionRate). Each hardwarePartConsumptionRate value (ie, instance of the PropertyType data type) must then have an associated ApplicabilityStatement, as defined in the UoF Applicability Statement, in order to define when respective consumption rate is applicable. The applicability statement will in most cases include HardwareElementPartRealization, ie, usage, as one of the parameters to be evaluated.

4.4.3.7 SubstanceDefinition

The SubstanceDefinition class deals with the identification of high concern physical matter that is contained in one or many HardwarePartAsDesigned.

SubstanceDefinition attributes:

- substanceIdentifier (one or many)
- substanceName (zero or one)
- substanceDescription (zero or one)
- substanceUsageCategory
- substanceRiskDescription (one or many)
- substanceRiskFactor (zero or one)
- substanceCharacteristicsRecordingDate

SubstanceDefinition associations:

- A SubstanceDefinition can be contained in zero, one or many HardwarePartAsDesigned (via the ContainedSubstance <>relationship>> class)

4.4.3.8 ContainedSubstance

The ContainedSubstance <>relationship>> class defines a relationship in between a SubstanceDefinition and a HardwarePartAsDesigned in which the substance is contained. It can eg, record the quantity of the contained substance, and a justification.

ContainedSubstance attributes:

- quantityOfContainedSubstance (zero or one)
- containedSubstanceJustificationDescription (zero or one)

Note

There is one instance of ContainedSubstance per relevant combination of SubstanceDefinition and HardwarePartAsDesigned.

4.4.3.9 SoftwarePartAsDesigned

SoftwarePartAsDesigned is a specialization of class PartAsDesigned, and represents parts that will be realized as executable software or as data files (eg, maps).

SoftwarePartAsDesigned attributes:

- partIdentifier (inherited from class Part)

- partName (inherited from class Part)
 - SoftwarePartAsDesignedDesignData (<<attributeGroup>>)
- partSpecialHandlingRequirement (inherited from class PartAsDesigned PartAsDesignedSupportData <<attributeGroup>>)
- softwareType
- softwarePartSize
 - SoftwarePartAsDesignedSupportData (<<attributeGroup>>)
- partMaturityClass (inherited from class PartAsDesigned PartAsDesignedSupportData <<attributeGroup>>)
- partObsolescenceRiskAssessment (inherited from class PartAsDesigned PartAsDesignedSupportData <<attributeGroup>>)
 - SoftwarePartAsDesignedControlledItemData (<<attributeGroup>>)
- partDemilitarizationClass (inherited from class PartAsDesigned PartAsControlledItemData <<attributeGroup>>)

Note

SoftwarePartAsDesignedDesignData (<<attributeGroup>>) is the definitional information which defines characteristics for the SoftwarePartAsDesigned identified during the design activities.

Note

SoftwarePartAsDesignedSupportData (<<attributeGroup>>) is the definitional information which defines characteristics for the SoftwarePartAsDesigned identified during the supportability analysis activities.

Note

SoftwarePartAsDesignedControlledItemData (<<attributeGroup>>) is the definitional information which defines characteristics for the SoftwarePartAsDesigned identified during different control analysis activities.

SoftwarePartAsDesigned associations:

- A SoftwarePartAsDesigned can have zero, one or many types of PartAsDesignedPartsList and revisions thereof (inherited from class PartAsDesigned)
- A SoftwarePartAsDesigned can be included in zero, one or many other parts, as a child of one or many instances of the PartAsDesignedPartsListEntry <<relationship>> class (inherited from class PartAsDesigned)
- A SoftwarePartAsDesigned can refer to zero, one or many alternate SoftwarePartAsDesigned (inherited from class PartAsDesigned)
- A SoftwarePartAsDesigned can be an alternate part for one or many other SoftwarePartAsDesigned (inherited from class PartAsDesigned)

4.5

4.5.1

UoF Breakdown Element Realization

Overall description

The Breakdown Element Realization UoF, [Fig 22](#), is used to:

- Define hardware and software breakdown elements along with their realizations in terms of hardware and software parts, respectively
- Distinguish hardware and software breakdown elements from other types of breakdown elements such as eg, systems and zones

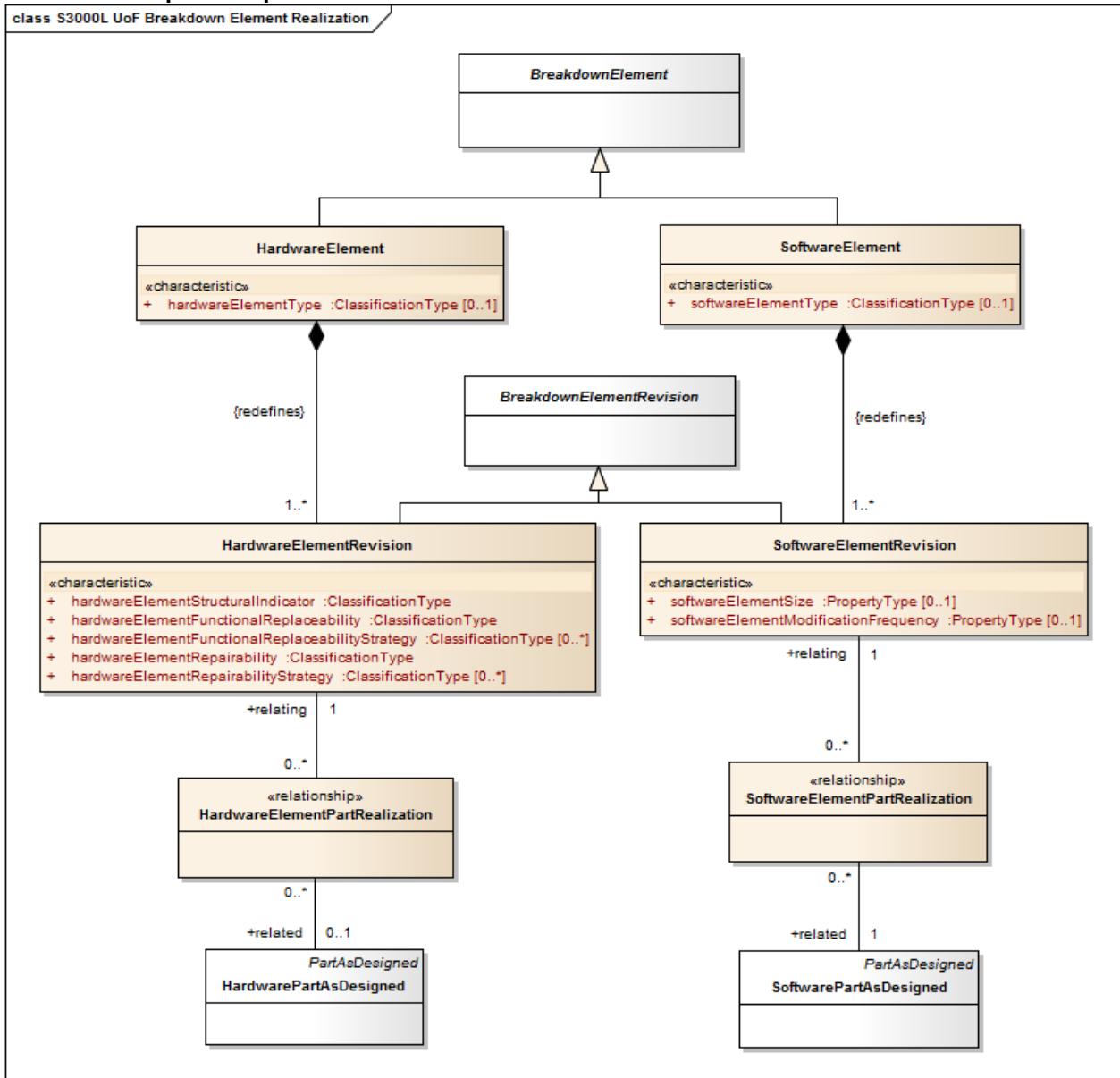
Note

HardwareElement and SoftwareElement are both specializations of the class BreakdownElement which means that, wherever the class BreakdownElement is being

used in the data model, HardwareElement or SoftwareElement can be used instead (the rule of substitutability).

HardwareElementRevision and SoftwareElementRevision are both specializations of the class BreakdownElementRevision which means that, wherever the class BreakdownElementRevision is being used in the data model, HardwareElementRevision or SoftwareElementRevision can be used instead (the rule of substitutability).

4.5.2 Graphical representation



ICN-B6865-S3000L0219-003-00

Fig 22 S3000L UoF Breakdown Element Realization - class model

4.5.3 UoF Breakdown Element Realization - New class and interface definitions

4.5.3.1 HardwareElement

The **HardwareElement** class is a specialization of the **BreakdownElement** class, and represents a specification that is realized as hardware parts (often referred to as equipment).

Applicable to: All

S3000L-A-19-00-0000-00A-040A-A

Chap 19

Note

Wherever the BreakdownElement class is used in the data model, HardwareElement can be used instead (the rule of substitutability).

HardwareElement attributes:

- breakdownElementIdentifier (inherited from class BreakdownElement)
- breakdownElementName (inherited from class BreakdownElement)
- breakdownElementEssentiality (inherited from class BreakdownElement)
- hardwareElementType (zero or one)

HardwareElement associations:

- An association with its HardwareElementRevisions (design iterations)

4.5.3.2 HardwareElementRevision

The HardwareElementRevision class is a specialization of the BreakdownElementRevision class, and represents a design iteration that is applied to a HardwareElement.

Note

Wherever the BreakdownElementRevision class is used in the data model, HardwareElementRevision can be used instead (the rule of substitutability).

HardwareElementRevision attributes:

- breakdownElementRevisionIdentifier (inherited from class BreakdownElementRevision)
- breakdownElementRevisionCreationDate (inherited from class BreakdownElementRevision)
- breakdownElementRevisionStatus (inherited from class BreakdownElementRevision)
- maintenanceSignificantOrRelevant (inherited from class BreakdownElementRevision)
- hardwareElementStructuralIndicator
- hardwareElementFunctionalReplaceability
- hardwareElementFunctionalReplaceabilityStrategy (zero, one or many)
- hardwareElementRepairability
- hardwareElementRepairabilityStrategy (zero, one or many)

HardwareElementRevision associations:

- An association with the HardwareElement of which the HardwareElementRevision is a revision
- Associations with one or many instances of BreakdownElementUsageInBreakdown, ie, usages of the HardwareElementRevision in BreakdownRevisions (inherited from class BreakdownElementRevision)
- An instance of HardwareElementRevision can relate to one or many other instances of BreakdownElementRevision via the BreakdownElementRevisionRelationship <> relationship class (inherited from class BreakdownElementRevision)
- An instance of HardwareElementRevision can be related to from one or many other instances of BreakdownElementRevision via the BreakdownElementRevisionRelationship <> relationship class (inherited from class BreakdownElementRevision)
- An optional association with zero, one or many HardwarePartAsDesigned that can be used to realize the defined HardwareElementRevision (via the HardwareElementPartRealization <> relationship class)

HardwareElementRevision special recommendations:

- A HardwareElementRevision can have multiple replaceability strategies (ie, values of the attribute hardwareElementFunctionalReplaceabilityStrategy). Each replaceability strategy value (ie, instance of ClassificationType) must then have an associated ApplicabilityStatement as defined in the UoF Applicability Statement, in order to define when the respective replaceability strategy is applicable.

- A HardwareElementRevision can have multiple reparability strategies (ie, values of the hardwareElementRepairabilityStrategy attribute). Each hardwareElementRepairabilityStrategy value (ie, instance of ClassificationType) must then have an associated ApplicabilityStatement as defined in UoF Applicability Statement, in order to define when respective hardwareElementRepairabilityStrategy is applicable.

4.5.3.3 **HardwareElementPartRealization**

The **HardwareElementPartRealization <>relationship>>** class defines an association between an instance of **HardwareElementRevision** and the **HardwarePartAsDesigned** which fulfills the hardware element specification.

HardwareElementPartRealization associations:

- (relating) The instance of **HardwareElementRevision** that has a related **HardwarePartAsDesigned** realization
- (related) The instance of **HardwarePartAsDesigned** that is the realization of the relating **HardwareElementRevision**

Note

There is one instance of **HardwareElementPartRealization** per combination of **HardwareElementRevision** and **HardwarePartAsDesigned**.

HardwareElementPartRealization special recommendations:

- Where an instance of **HardwareElementPartRealization** is dependent upon product variant or allowed product design configuration, it must be defined by the assignment of **ItemInProductVariant** or **ItemInAllowedProductConfiguration** to the instance of **HardwareElementPartRealization** under consideration. Refer to [Para 4.8](#).

4.5.3.4 **SoftwareElement**

The **SoftwareElement** class is a specialization of the **BreakdownElement** class, and represents a specification that is realized in terms of software.

Note

Wherever the **BreakdownElement** class is used in the data model, **SoftwareElement** can be used instead (the rule of substitutability).

SoftwareElement attributes:

- **breakdownElementIdentifier** (inherited from class **BreakdownElement**)
- **breakdownElementName** (inherited from class **BreakdownElement**)
- **breakdownElementEssentiality** (inherited from class **BreakdownElement**)
- **softwareElementType** (zero or one)

SoftwareElement associations:

- An association with its **SoftwareElementRevisions** (design iterations)

4.5.3.5 **SoftwareElementRevision**

The **SoftwareElementRevision** class is a specialization of the **BreakdownElementRevision** class, and represents a design iteration that is applied to a **SoftwareElement**.

Note

Wherever the **BreakdownElementRevision** class is used in the data model, **SoftwareElementRevision** can be used instead (the rule of substitutability).

SoftwareElementRevision attributes:

- **breakdownElementRevisionIdentifier** (inherited from class **BreakdownElementRevision**)
- **breakdownElementRevisionCreationDate** (inherited from class **BreakdownElementRevision**)

- breakdownElementRevisionStatus (inherited from class BreakdownElementRevision)
- maintenanceSignificantOrRelevant (inherited from class BreakdownElementRevision)
- softwareElementSize (zero or one)
- softwareElementModificationFrequency (zero or one)

SoftwareElementRevision associations:

- An association with the SoftwareElement of which the SoftwareElementRevision is a revision
- Associations with one or many instances of BreakdownElementUsageInBreakdown, ie, usages of the SoftwareElementRevision in BreakdownRevisions (inherited from class BreakdownElementRevision)
- An instance of SoftwareElementRevision can relate to one or many other instances of BreakdownElementRevision via the BreakdownElementRevisionRelationship <>relationship> class (inherited from class BreakdownElementRevision)
- An instance of SoftwareElementRevision can be related to from one or many other instances of BreakdownElementRevision via the BreakdownElementRevisionRelationship <>relationship> class (inherited from class BreakdownElementRevision)
- An optional association with zero, one or many SoftwarePartAsDesigned that can be used to realize the defined SoftwareElementRevision (via the SoftwareElementPartRealization <>relationship> class)

4.5.3.6 SoftwareElementPartRealization

The SoftwareElementPartRealization <>relationship> class defines an association between an instance of SoftwareElementRevision and the SoftwarePartAsDesigned which fulfills the software element specification.

SoftwareElementPartRealization associations:

- (relating) The instance of SoftwareElementRevision that has a related SoftwarePartAsDesigned realization
- (related) The instance of SoftwarePartAsDesigned that is the realization of the relating SoftwareElementRevision

Note

There is one instance of SoftwareElementPartRealization per combination of SoftwareElementRevision and SoftwarePartAsDesigned.

SoftwareElementPartRealization special recommendations:

- Where an instance of SoftwareElementPartRealization is dependent upon product variant or allowed product design configuration, it must be defined by the assignment of ItemInProductVariant or ItemInAllowedProductConfiguration to the instance of SoftwareElementPartRealization under consideration. Refer to [Para 4.8](#).

4.5.4 UoF Breakdown Element Realization - Additions to referenced class and interface definitions

4.5.4.1 HardwarePartAsDesigned

The HardwarePartAsDesigned class defined in UoF Part Definition, must also implement the following additional association:

- An optional association with zero, one or many instances of HardwareElementRevision, via the HardwareElementPartRealization <>relationship> class

4.5.4.2 SoftwarePartAsDesigned

The SoftwarePartAsDesigned class defined in UoF Part Definition, must also implement the following additional association:

- An optional association with zero, one or many instances of SoftwareElementRevision, via the SoftwareElementPartRealization <>relationship> class

4.6

4.6.1

UoF Breakdown Zone Element

Overall description

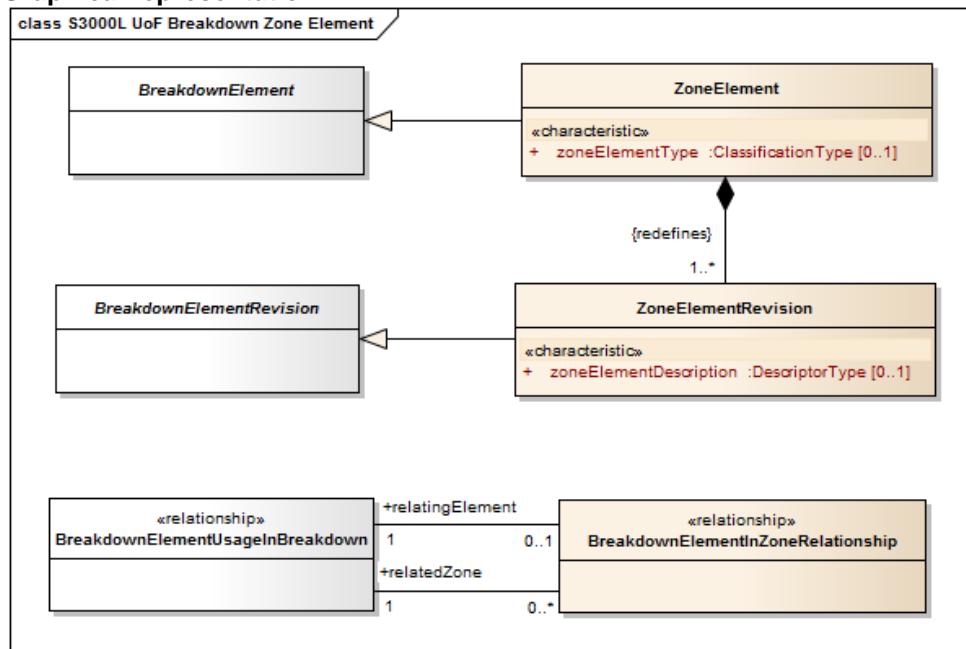
The Breakdown Zone Element UoF, [Fig 23](#), is used to distinguish between zonal breakdown elements and other types of breakdown elements (eg, hardware, software).

Note

Zone elements can be used to define eg, zones within a product, as well as generic work areas, eg, a mechanical workshop.

4.6.2

Graphical representation



ICN-B6865-S3000L0220-003-00

Fig 23 S3000L UoF Breakdown Zone Element - class model

4.6.3

4.6.3.1

UoF Breakdown Zone Element - New class and interface definitions

ZoneElement

The ZoneElement class is a specialization of the BreakdownElement class, and is used to represent a 3 dimensional space within a Product.

Note

Wherever the BreakdownElement class is used in the data model, ZoneElement can be used instead (the rule of substitutability).

ZoneElement attributes:

- breakdownElementIdentifier (inherited from class BreakdownElement)
- breakdownElementName (inherited from class BreakdownElement)
- breakdownElementEssentiality (inherited from class BreakdownElement)
- zoneElementType (zero or one)

ZoneElement associations:

- An association with its ZoneElementRevisions (design iterations)

4.6.3.2 ZoneElementRevision

The ZoneElementRevision class is a specialization of the BreakdownElementRevision class, and represents a design iteration that is applied to a ZoneElement.

Note

Wherever the BreakdownElementRevision class is used in the data model, ZoneElementRevision can be used instead (the rule of substitutability).

ZoneElementRevision attributes:

- breakdownElementRevisionIdentifier (inherited from class BreakdownElementRevision)
- breakdownElementRevisionCreationDate (inherited from class BreakdownElementRevision)
- breakdownElementRevisionStatus (inherited from class BreakdownElementRevision)
- maintenanceSignificantOrRelevant (inherited from class BreakdownElementRevision)
- zoneElementDescription (zero or one)

ZoneElementRevision associations:

- An association with the ZoneElement of which the ZoneElementRevision is a revision
- Associations with one or many instances of BreakdownElementUsageInBreakdown, ie, usages of the ZoneElementRevision in BreakdownRevisions (inherited from class BreakdownElementRevision)
- An instance of ZoneElementRevision can relate to one or many other instances of BreakdownElementRevision via the BreakdownElementRevisionRelationship <>relationship> class (inherited from class BreakdownElementRevision)
- An instance of ZoneElementRevision can be related to from one or many other instances of BreakdownElementRevision via the BreakdownElementRevisionRelationship <>relationship> class (inherited from class BreakdownElementRevision)

4.6.3.3 BreakdownElementInZoneRelationship

The BreakdownElementInZoneRelationship <>relationship> class defines an association between two instances of BreakdownElementUsageInBreakdown where the relatedZone instance must represent the usage of a zonal location (ZoneElementRevision).

BreakdownElementInZoneRelationship associations:

- (relatingElement) The instance of BreakdownElementUsageInBreakdown that has a zonal location
- (relatedZone) The BreakdownElementUsageInBreakdown that represents the zone where the relating breakdown element is located

Note

There is one instance of BreakdownElementInZoneRelationship per combination of BreakdownElementUsageInBreakdown and zonal location (ZoneElementRevision via the relatedZone association).

4.6.4 UoF Breakdown Zone Element - Additions to referenced class and interface definitions

4.6.4.1 BreakdownElementUsageInBreakdown

The BreakdownElementUsageInBreakdown class defined in UoF Breakdown Structure must also implement the following additional associations:

- An optional association with a ZoneElementRevision via the BreakdownElementInZoneRelationship and BreakdownElementUsageInBreakdown <>relationship> classes
- If the BreakdownElementUsageInBreakdown is a usage of ZoneElementRevision it can also have an optional association with zero, one or many BreakdownElementRevisions, via the BreakdownElementInZoneRelationship and BreakdownElementUsageInBreakdown <>relationship> classes

4.7 UoF Breakdown Aggregated Element

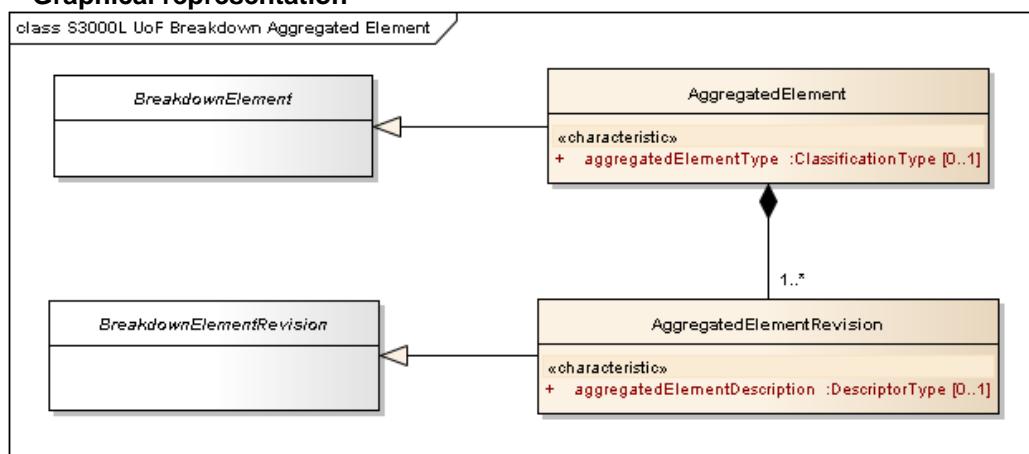
4.7.1 Overall description

The Breakdown Aggregated Element UoF, [Fig 24](#), is used to distinguish between aggregated breakdown elements such as systems, functions or slots and other types of breakdown elements such as hardware and software.

Note

Aggregated breakdown elements of type 'system' and 'function' can have a set of child breakdown elements. Refer to [Para 4.3](#). These child elements must be seen as a whole-parts relationship, ie, all children must exist in order for the parent system or function to be complete. However aggregated breakdown elements of type 'Slot' indicates that all child elements are alternative breakdown elements which can be used in the defined slot location.

4.7.2 Graphical representation



ICN-B6865-S3000L0221-002-00

Fig 24 S3000L UoF Breakdown Aggregated Element - class model

4.7.3 UoF Breakdown Aggregated Element - New class and interface definitions

4.7.3.1 AggregatedElement

The AggregatedElement class is a specialization of the BreakdownElement class, and identifies a collection of BreakdownElements that are grouped for an identified purpose, eg, a system, function or a slot.

Note

Wherever the BreakdownElement class is used in the data model, AggregatedElement can be used instead (the rule of substitutability).

AggregatedElement attributes:

- breakdownElementIdentifier (inherited from class BreakdownElement)
- breakdownElementName (inherited from class BreakdownElement)
- breakdownElementEssentiality (inherited from class BreakdownElement)
- aggregatedElementType

AggregatedElement associations:

- An association with its AggregatedElementRevisions (design iterations)

4.7.3.2 AggregatedElementRevision

The AggregatedElementRevision class is a specialization of the BreakdownElementRevision class, and represents a design iteration that is applied to an AggregatedElement.

Note

Wherever BreakdownElementRevision is used in the data model, AggregatedElementRevision can be used instead (the rule of substitutability).

AggregatedElementRevision attributes:

- breakdownElementRevisionIdentifier (inherited from class BreakdownElementRevision)
- breakdownElementRevisionCreationDate (inherited from class BreakdownElementRevision)
- breakdownElementRevisionStatus (inherited from class BreakdownElementRevision)
- maintenanceSignificantOrRelevant (inherited from class BreakdownElementRevision)
- aggregatedElementDescription (zero or one)

AggregatedElementRevision associations:

- An association with the AggregatedElement of which the AggregatedElementRevision is a revision
- Associations with one or many instances of BreakdownElementUsageInBreakdown, ie, usages of the AggregatedElementRevision in BreakdownRevisions (inherited from class BreakdownElementRevision)
- An instance of AggregatedElementRevision can relate to one or many other instances of BreakdownElementRevision via the BreakdownElementRevisionRelationship <>relationship> class (inherited from class BreakdownElementRevision)
- An instance of AggregatedElementRevision can be related to from one or many other instances of BreakdownElementRevision via the BreakdownElementRevisionRelationship <>relationship> class (inherited from class BreakdownElementRevision)

4.8 UoF Product Design Configuration

4.8.1 Overall description

The Product Design Configuration UoF, [Fig 25](#), enables definition of product variants (models) and allowed product configurations.

Note

Standards such as ISO 10303 often use the term effectivity to describe product design related restrictions, where the restriction defines the planned usage of components in the context of a particular product configuration.

A product variant identifies a member of a product family that is configured for a specific purpose and is offered to customers. A product variant can either have its own uniquely identified breakdown structures, or be filtered out from the product breakdown structures by ItemInProductVariant, also known as System/End Item Usable On Code in GEIA-STD-0007.

A product variant can be further refined into specific allowed product configurations, which defines permitted combinations of hardware and software parts which can or must be installed in specific locations (positions), and demonstrates that a product complies with applicable regulations.

An allowed product configuration can either be defined as:

- A part numbered item (AllowedProductConfigurationHardwarePartAsDesigned) and have its explicit BOM structure (PartAsDesignedPartsList)
- A product configuration where hardware and software parts are related to its allowed location

This UoF also allows for restricting the applicability of part numbered items to specific end item serial number ranges (ApplicableBlockOfSerializedEndItems).

Note

Information regarding ItemInProductVariant, AllowedProductConfiguration and ApplicableBlockOfSerializedEndItems, is often received from the design department as part of the product definition.

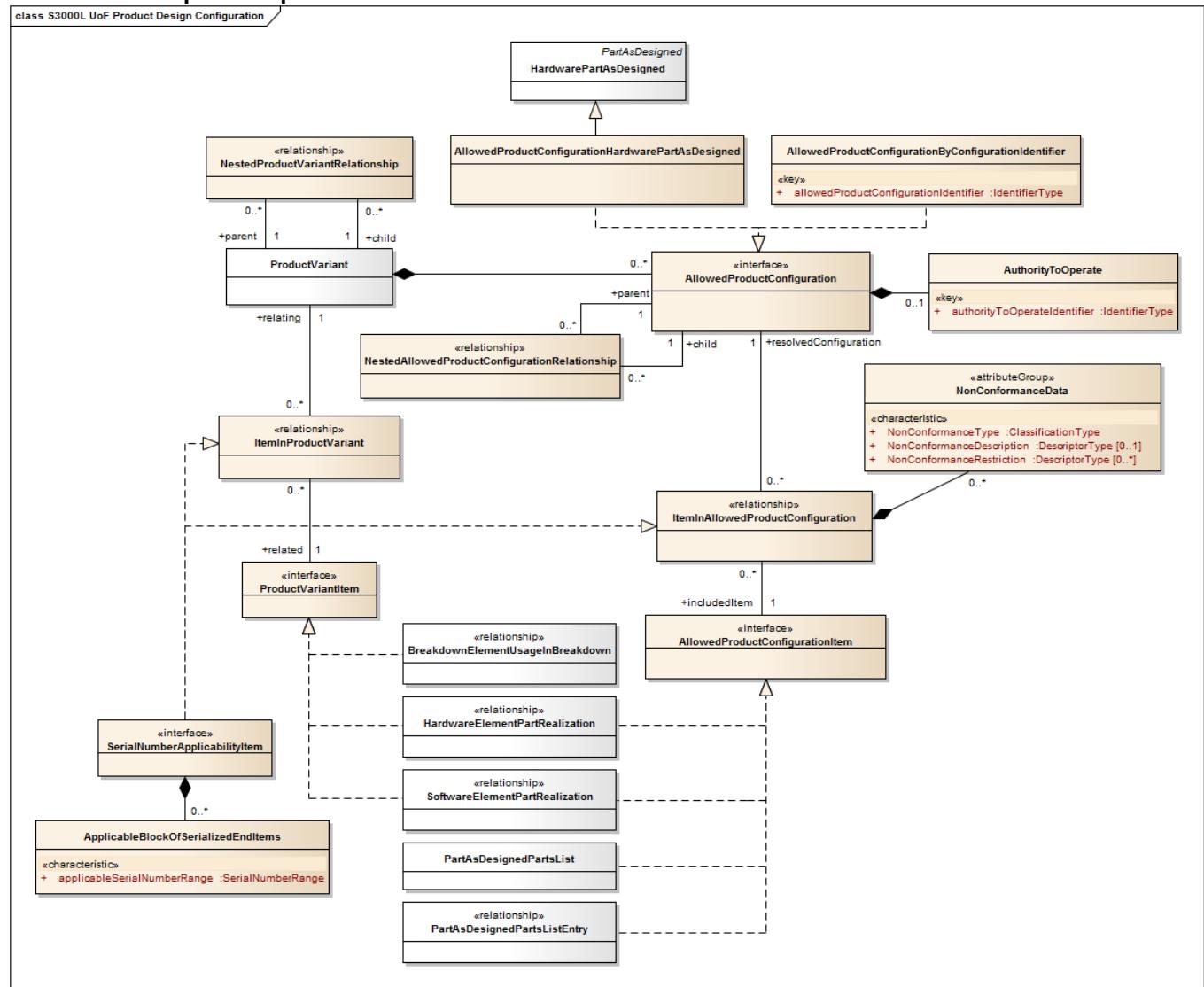
Note

ItemInProductVariant and AllowedProductConfiguration help the LSA analyst to filter out BreakdownElements and/or Parts that are within scope of a specific product variant, and/or allowed configuration thereof.

Note

ItemInProductVariant and AllowedProductConfiguration are used to define restrictions for a given product design (product definition), whereas ApplicabilityStatement is used to define usage related restrictions. Refer to [Para 4.22](#),

4.8.2 Graphical representation



ICN-B6865-S3000L0222-003-00

Fig 25 S3000L UoF Product Design Configuration - class model

4.8.3 UoF Product Design Configuration - New class and interface definitions

4.8.3.1 ItemInProductVariant

The ItemInProductVariant <<relationship>> class defines that the applicability of a breakdown element or specific realization thereof is limited to a specific product variant.

Note

ItemInProductVariant can be used for representing GEIA-STD-0007 System/End Item Usable On Codes.

Note

Instances of BreakdownElementUsageInBreakdown without any associated ItemInProductVariant means that the usage of the associated BreakdownElementRevision is valid for all ProductVariants.

Note

There must be one instance of ItemInProductVariant per permitted combination of ProductVariant and BreakdownElementUsageInBreakdown, HardwareElementPartRealization and SoftwareElementPartRealization, respectively.

ItemInProductVariant implement the following <><interface>>:

- SerialNumberApplicabilityItem

ItemInProductVariant associations:

- (related) The BreakdownElementUsageInBreakdown, HardwareElementPartRealization or SoftwareElementPartRealization which applicability is constrained to a specific ProductVariant
- (relating) The ProductVariant for which the breakdown element usage or part realization is applicable

4.8.3.2 ProductVariantItem

The ProductVariantItem <><interface>> is implemented by classes which applicability can be restricted to specific ProductVariants.

Classes that implements the ProductVariantItem <><interface>> are:

- BreakdownElementUsageInBreakdown
- HardwareElementPartRealization
- SoftwareElementPartRealization

Classes that implement the ProductVariantItem <><interface>> must also implement the following association:

- An optional association with zero, one or many instances of ProductVariant (via the ItemInProductVariant <><relationship>> class)

Note

An instance of a class that implements the ProductVariantItem <><interface>> can be associated with more than one ProductVariant, ie, an instance representing the ItemInProductVariant can be applicable to more than one ProductVariant.

4.8.3.3 NestedProductVariantRelationship

The NestedProductVariantRelationship <><relationship>> class defines an association between a parent ProductVariant and a ProductVariant that is defined to be included as part of the parent ProductVariant.

NestedProductVariantRelationship associations:

- (parent) The ProductVariant that includes the child ProductVariant
- (child) The ProductVariant which is part of a parent ProductVariant

Note

There is one instance of NestedProductVariantRelationship per parent-child relationship.

4.8.3.4 AllowedProductConfiguration

The AllowedProductConfiguration <><interface>> is implemented by classes that define allowed configurations for a defined ProductVariant. AllowedProductConfiguration defines permitted combinations of hardware and software parts which can or must be installed in specific locations (positions) of an end item, and demonstrates that the product complies with applicable regulations.

Classes that implement the AllowedProductConfiguration <><interface>> are:

- AllowedProductConfigurationHardwarePartAsDesigned (a specialization of HardwarePartAsDesigned)
- AllowedProductConfigurationByConfigurationIdentifier

Classes that implement the AllowedProductConfiguration <> must also implement the following associations:

- An association with the ProductVariant for which the AllowedProductConfiguration is defined
- An optional association with an AuthorityToOperate, issued by a regulating body
- An instance of AllowedProductConfiguration can relate to zero, one or many other instances of AllowedProductConfiguration that are allowed to be included as part of the AllowedProductConfiguration under consideration (via the NestedAllowedProductConfigurationRelationship <> class)
- An instance of AllowedProductConfiguration can be related to from zero, one or many other instances of AllowedProductConfiguration in which the AllowedProductConfiguration under consideration is allowed to be a part (via the NestedAllowedProductConfigurationRelationship <> class)
- An association with the hardware and software parts that are allowed to be installed on end items that complies with the defined AllowedProductConfiguration (via the ItemInAllowedProductConfiguration <> class)

4.8.3.5 NestedAllowedProductConfigurationRelationship

The NestedAllowedProductConfigurationRelationship <> class defines an association between a parent AllowedProductConfiguration and an AllowedProductConfiguration that is allowed to be included as part of the parent AllowedProductConfiguration.

NestedAllowedProductConfigurationRelationship associations:

- (parent) The AllowedProductConfiguration that includes the child AllowedProductConfiguration
- (child) The AllowedProductConfiguration which is part of a parent AllowedProductConfiguration

Note

There is one instance of NestedAllowedProductConfigurationRelationship per parent-child relationship.

4.8.3.6 AuthorityToOperate

AuthorityToOperate signifies the safety aspects of a product to be manufactured, and is issued by a regulating body, and once issued, the design cannot be changed without re-authorization.

AuthorityToOperate attributes:

- authorityToOperateIdentifier

AuthorityToOperate associations:

- An association with the AllowedProductConfiguration for which the AuthorityToOperate is issued

4.8.3.7 AllowedProductConfigurationHardwarePartAsDesigned

The AllowedProductConfigurationHardwarePartAsDesigned class is a specialization of class HardwarePartAsDesigned, and represents hardware parts that are issued as authorized allowed product configurations.

AllowedProductConfigurationHardwarePartAsDesigned implement the following <>:

- AllowedProductConfiguration

AllowedProductConfigurationHardwarePartAsDesigned attributes:

- partIdentifier (inherited from class PartAsDesigned)
- partName (inherited from class Part)
 - HardwarePartAsDesignedDesignData (<<attributeGroup>>)
- partSpecialHandlingRequirement (inherited from class PartAsDesigned)
- hardwarePartOperationalAuthorizedLife (inherited from class HardwarePartAsDesigned)
- hardwarePartHazardousClass (inherited from class HardwarePartAsDesigned)
- hardwarePartFitmentRequirement (inherited from class HardwarePartAsDesigned)
- hardwarePartElectromagneticIncompatible (inherited from class HardwarePartAsDesigned)
- hardwarePartElectrostaticSensitive (inherited from class HardwarePartAsDesigned)
- hardwarePartElectromagneticSensitive (inherited from class HardwarePartAsDesigned)
- hardwarePartMagneticSensitive (inherited from class HardwarePartAsDesigned)
- hardwarePartRadiationSensitive (inherited from class HardwarePartAsDesigned)
 - HardwarePartAsDesignedSupportData (<<attributeGroup>>)
- partMaturityClass (inherited from class PartAsDesigned)
- partObsolescenceRiskAssessment (inherited from class PartAsDesigned)
- hardwarePartLogisticsCategory (inherited from class HardwarePartAsDesigned)
- hardwarePartRepairability (inherited from class HardwarePartAsDesigned)
- hardwarePartRepairabilityStrategy (inherited from class HardwarePartAsDesigned)
- hardwarePartMaintenanceStart (inherited from class HardwarePartAsDesigned)
- hardwarePartWasteProductsInUseDisposalDescription (inherited from class HardwarePartAsDesigned)
- hardwarePartWasteProductsPlannedDisposalDescription (inherited from class HardwarePartAsDesigned)
- hardwarePartEnvironmentalAspectInUseClass (inherited from class HardwarePartAsDesigned)
- hardwarePartEnvironmentalAspectPlannedDisposalClass (inherited from class HardwarePartAsDesigned)
- hardwarePartScrapRate (inherited from class HardwarePartAsDesigned)
- hardwarePartConsumptionRate (inherited from class HardwarePartAsDesigned)
 - HardwarePartAsDesignedControlledItemData (<<attributeGroup>>)
- partDemilitarizationClass (inherited from class PartAsDesigned)

AllowedProductConfigurationHardwarePartAsDesigned associations:

- An AllowedProductConfigurationHardwarePartAsDesigned can have zero, one or many revisions of PartAsDesignedPartsList (inherited from class PartAsDesigned)
- An AllowedProductConfigurationHardwarePartAsDesigned can be included in zero, one or many other parts as a child of one or many instances of the PartAsDesignedPartsListEntry <<relationship>> class (inherited from class PartAsDesigned)
- An AllowedProductConfigurationHardwarePartAsDesigned can refer to zero, one or many alternate AllowedProductConfigurationHardwarePartAsDesigned (inherited from class PartAsDesigned)
- An AllowedProductConfigurationHardwarePartAsDesigned can be an alternate part for one or many other AllowedProductConfigurationHardwarePartAsDesigned (inherited from class PartAsDesigned)
- An AllowedProductConfigurationHardwarePartAsDesigned can have zero, one or many associated substances, which are contained in the HardwarePartAsDesigned (via the ContainedSubstance <<relationship>> class)

- 4.8.3.8 AllowedProductConfigurationByConfigurationIdentifier
The AllowedProductConfigurationByConfigurationIdentifier class represents an AllowedProductConfiguration that is identified and managed by other means than a part identifier.
- AllowedProductConfigurationHardwarePartAsDesigned implement the following <<interface>>:
- AllowedProductConfiguration
- AllowedProductConfigurationHardwarePartAsDesigned attribute:
- allowedProductConfigurationIdentifier
- 4.8.3.9 ItemInAllowedProductConfiguration
The ItemInAllowedProductConfiguration <<relationship>> class defines an association between a resolved configuration and a definition of the parts that is allowed to be installed on serialized Products that confirms with the resolved AllowedProductConfiguration.
- ItemInAllowedProductConfiguration implement the following <<interface>>:
- SerialNumberApplicabilityItem
- ItemInAllowedProductConfiguration associations:
- (resolvedConfiguration) The AllowedProductConfiguration in which the includedItem is applicable
 - (includedItem) The item which applicability is restricted to a specific AllowedProductConfiguration
 - An optional association with one or many instances of NonConformanceData <<attributeGroup>> that documents if a specific related part does not fully comply with the requirements of its usage in the resolved AllowedProductConfiguration
- Note**
- There is one instance of ItemInAllowedProductConfiguration per relevant combination of resolvedConfiguration and includedItem.
- 4.8.3.10 NonConformanceData
The NonConformanceData <<attributeGroup>> defines attributes that must be provided if a specific part does not fully comply with the requirements of its usage in the resolved AllowedProductConfiguration.
- NonConformanceData attributes:
- NonConformanceType
 - NonConformanceDescription (zero or one)
 - NonConformanceRestriction (zero, one or many)
- NonConformanceData association:
- An association with the ItemInAllowedProductConfiguration which is non-conformant
- 4.8.3.11 AllowedProductConfigurationItem
The AllowedProductConfigurationItem <<interface>> is implemented by classes which applicability can be the restricted to specific AllowedProductConfigurations.
- Classes that implements the AllowedProductConfigurationItem <<interface>> are:
- HardwareElementPartRealization
 - SoftwareElementPartRealization
 - PartAsDesignedPartsList
 - PartAsDesignedPartsListEntry

Classes that implement the AllowedProductConfigurationItem <<interface>> must also implement the following association:

- An optional association with zero, one or many instances of classes that implement the AllowedProductConfiguration <<interface>> (via the ItemInAllowedProductConfiguration <<relationship>> class)

Note

An instance of a class that implements the AllowedProductConfigurationItem <<interface>> can be associated with more than one AllowedProductConfiguration, ie, an instance representing the includedItem can be applicable to more than one AllowedProductConfiguration.

4.8.3.12 SerialNumberApplicabilityItem

The SerialNumberApplicabilityItem <<interface>> is implemented by classes that can have a limited applicability to ranges of serialized end items.

Classes that implement the SerialNumberApplicabilityItem <<interface>> are:

- ItemInProductVariant
- ItemInAllowedProductConfiguration

Classes that implement the SerialNumberApplicabilityItem <<interface>> must also implement the following association:

- An optional association with zero, one or many instances of the ApplicableBlockOfSerializedEndItems <<attributeGroup>>, identifying ranges serialized end items

4.8.3.13 ApplicableBlockOfSerializedEndItems

The ApplicableBlockOfSerializedEndItems <<attributeGroup>> defines a possibly open-ended serial number range.

ApplicableBlockOfSerializedEndItems attributes:

- applicableSerialNumberRange

ApplicableBlockOfSerializedEndItems association:

- An association with the item for which the serial number range is defined (via the SerialNumberApplicabilityItem <<interface>>)

4.8.4 UoF Product Design Configuration - Additions to referenced class and interface definitions

4.8.4.1 ProductVariant

The ProductVariant class defined in UoF Product and Project must also implement the following additional associations:

- An instance of ProductVariant can relate to zero, one or many other instances of ProductVariant which are included as part of the ProductVariant under consideration (via the NestedProductVariantRelationship <<relationship>> class)
- An instance of ProductVariant can be related to from zero, one or many other instances of ProductVariant in which the ProductVariant under consideration is a part (via the NestedProductVariantRelationship <<relationship>> class)
- An optional association with zero, one or many instances of items which are applicable to the ProductVariant under consideration (via the ItemInProductVariant <<relationship>> class)
- An optional association with zero, one or many instances of classes that implement the AllowedProductConfiguration <<interface>>

4.8.4.2 PartAsDesignedPartsList

The PartAsDesignedPartsList class defined in UoF Part Definition must also implement the following additional <>interfaces>>:

- AllowedProductConfigurationItem

4.8.4.3 PartAsDesignedPartsListEntry

The PartAsDesignedPartsListEntry class defined in UoF Part Definition must also implement the following additional <>interfaces>>:

- AllowedProductConfigurationItem

4.8.4.4 BreakdownElementUsageInBreakdown

The BreakdownElementUsageInBreakdown class defined in UoF Breakdown Structure must also implement the following additional <>interface>>:

- ProductVariantItem

4.8.4.5 HardwareElementPartRealization

The HardwareElementPartRealization class defined in UoF Breakdown Element Realization must also implement the following additional <>interfaces>>:

- ProductVariantItem
- AllowedProductConfigurationItem

4.8.4.6 SoftwareElementPartRealization

The SoftwareElementPartRealization class defined in UoF Breakdown Element Realization must also implement the following additional <>interfaces>>:

- ProductVariantItem
- AllowedProductConfigurationItem

4.9 UoF LSA Candidate

4.9.1 Overall description

The LSA Candidate UoF, [Fig 26](#), supports the selection of LSA candidates and the definition of the key performance indicators per selected LSA candidate.

Note

The recommendation is to select the LSA Candidates from a top-down perspective, ie, starting with the root node (breakdown element) for a given ProductVariant and then traverse through the Breakdown Structure and its part realizations, including BOMs (PartAsDesignedPartsList) and its entries (PartAsDesignedPartsListEntry).

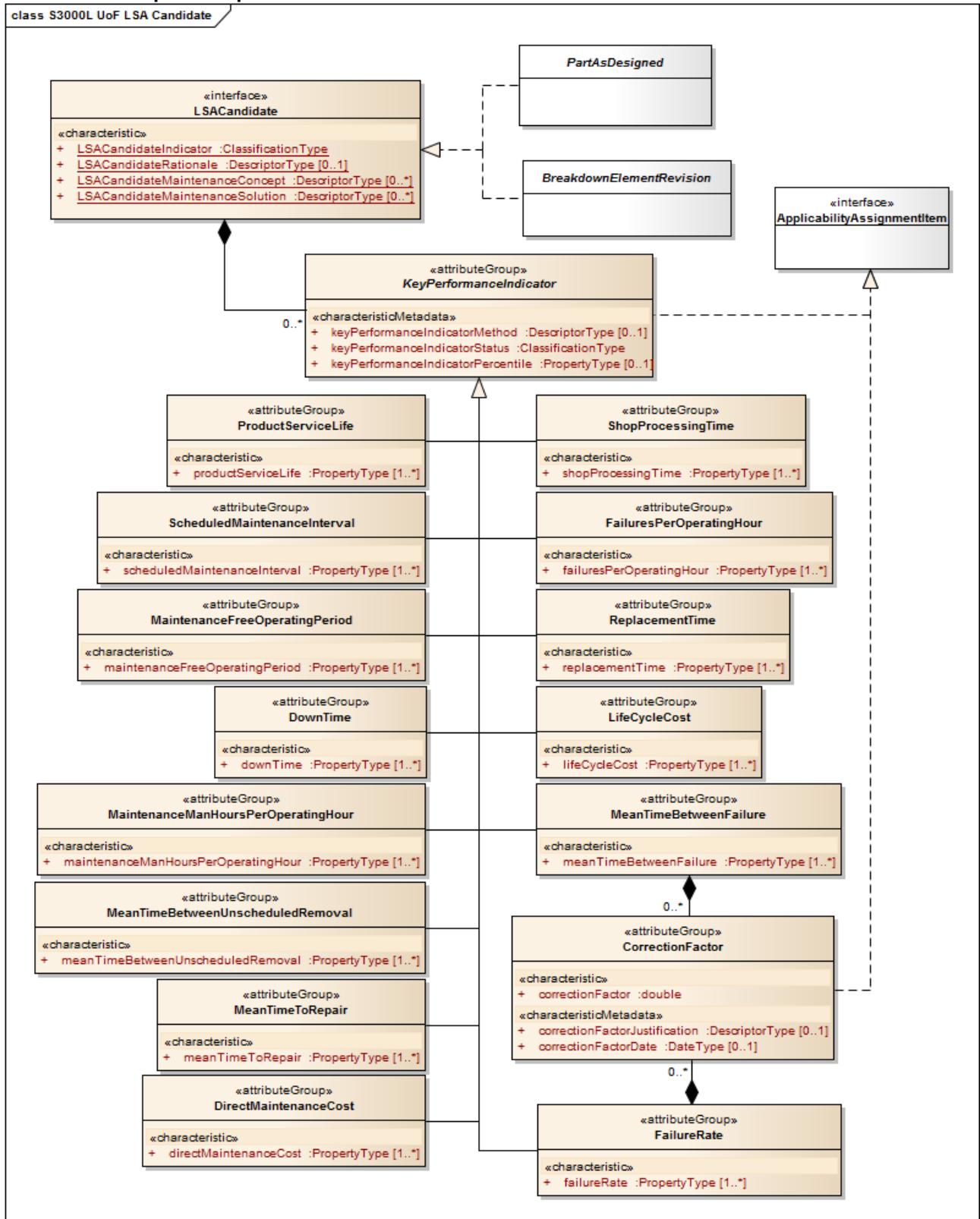
Note

Key performance indicators define the requirements for the project, and are often identified during the early stages of the LSA program.

Note

The process of distributing a key performance indicator between sub elements using eg, fractions is not explicitly supported in the data model. However, the data model supports the recording of the resulting values against the respective sub element. Key performance parameters that are the result of a distribution process, eg, by fraction, are recommended to set the value of the PropertyType valueDetermination attribute to "Distributed".

4.9.2 Graphical representation



ICN-B6865-S3000L0223-003-00

Fig 26 S3000L UoF LSA Candidate - class model

Applicable to: All

S3000L-A-19-00-0000-00A-040A-A

Chap 19

4.9.3 UoF LSA Candidate - New class and interface definitions

4.9.3.1 LSACandidate

The LSACandidate <> is implemented by classes that can be selected as LSA Candidates.

Classes that implement the LSACandidate <> are:

- BreakdownElementRevision
- PartAsDesigned

Since the LSACandidate <> must be implemented by BreakdownElementRevision and PartAsDesigned respectively, it means that instances of the following classes can be selected as LSA Candidates:

- HardwareElementRevision
- SoftwareElementRevision
- AggregatedElementRevision
- ZoneElementRevision
- HardwarePartAsDesigned
- SoftwarePartAsDesigned

Classes that implement the LSACandidate <> must also implement the following attributes:

- LSACandidateIndicator
- LSACandidateRationale (zero or one)
- LSACandidateMaintenanceConcept (zero, one or many)
- LSACandidateMaintenanceSolution (zero, one or many)

Classes that implement the LSACandidate <> must also implement the following association:

- An optional association with zero, one or many instances of the KeyPerformanceIndicator <>

Note

An associated KeyPerformanceIndicator must be of any its specializations (eg, MeanTimeBetweenFailure, DownTime).

Note

An LSACandidate can have more than one LSACandidateMaintenanceConcept and LSACandidateMaintenanceSolution description, respectively. The validity of the respective description can be determined by the assignment of ApplicabilityStatements. Refer to [Para 4.22](#).

4.9.3.2 KeyPerformanceIndicator

The KeyPerformanceIndicator <> defines eg, customer or user specific values per LSACandidate. The value for a given key performance indicator can be defined as being eg, specified, contracted, allocated, distributed or actual, using the valueDetermination attribute for the respective key performance value (valueDetermination is an attribute of the PropertyType primitive).

Note

Requirements are often stated as product design and performance data in the contract.

Note

Key performance indicators can be defined for LSA Candidates at any level of indenture.

Note

An instance of class KeyPerformanceIndicator has no meaning by itself, but only in the context of a specific LSACandidate.

Note

KeyPerformanceIndicator <>attributeGroup> is an abstract class, ie, an instantiation of KeyPerformanceIndicator must be done using any of its specializations:

- ProductServiceLife
- ScheduledMaintenanceInterval
- MaintenanceFreeOperatingPeriod
- DownTime
- MaintenanceManHoursPerOperatingHour
- MeanTimeBetweenUnscheduledRemoval
- MeanTimeToRepair
- DirectMaintenanceCost
- ShopProcessingTime
- FailuresPerOperatingHour
- ReplacementTime
- LifeCycleCost
- MeanTimeBetweenFailure
- FailureRate

KeyPerformanceIndicator attributes:

- keyPerformanceIndicatorMethod (zero or one)
- keyPerformanceIndicatorStatus
- keyPerformanceIndicatorPercentile (zero or one)

Note

The KeyPerformanceIndicator class does not contain any key performance indicator values. These are contained within the respective subclass of KeyPerformanceIndicator, eg, ProductServiceLife.

KeyPerformanceIndicator associations:

- An instance of KeyPerformanceIndicator is always defined for a specific LSACandidate

KeyPerformanceIndicator interfaces:

- KeyPerformanceIndicator implements the ApplicabilityAssignmentItem <>interface>

KeyPerformanceIndicator special recommendations:

- Where an instance of KeyPerformanceIndicator is dependent upon eg, operator or operational environment, it must be distinguished by the assignment of an ApplicabilityStatement to the KeyPerformanceIndicator. Refer to [Para 4.22](#).

4.9.3.3

ProductServiceLife

The ProductServiceLife <>attributeGroup> is a specialization of KeyPerformanceIndicator, and represents eg, the number of years the LSA Candidate is expected to be in service.

ProductServiceLife attributes:

- keyPerformanceIndicatorMethod (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorStatus (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorPercentile (inherited from class KeyPerformanceIndicator)
- productServiceLife (one or many)

ProductServiceLife associations:

- An instance of ProductServiceLife is always defined for a specific LSACandidate (inherited from class KeyPerformanceIndicator)

ProductServiceLife interfaces:

- ProductServiceLife implements the ApplicabilityAssignmentItem <> (inherited from class KeyPerformanceIndicator)

ProductServiceLife special recommendations:

- An LSACandidate can have multiple product service life values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of productServiceLife (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).
- An LSACandidate can have multiple product service life values being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one 'specified' product service life value, and one 'actual' product service life value. These values must be distinguished using the valueDetermination attribute for the respective instance of productServiceLife.

Note

valueDetermination is an attribute of the PropertyType primitive.

- If an instance of ProductServiceLife has multiple productServiceLife values without any additional applicability statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first

4.9.3.4

ScheduledMaintenanceInterval

The ScheduledMaintenanceInterval <> is a specialization of KeyPerformanceIndicator, and represents the number of operational units (eg, rounds, miles, hours) between scheduled maintenance.

ScheduledMaintenanceInterval attributes:

- keyPerformanceIndicatorMethod (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorStatus (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorPercentile (inherited from class KeyPerformanceIndicator)
- scheduledMaintenanceInterval (one or many)

ScheduledMaintenanceInterval associations:

- An instance of ScheduledMaintenanceInterval is always defined for a specific LSACandidate (inherited from class KeyPerformanceIndicator)

ScheduledMaintenanceInterval interfaces:

- ScheduledMaintenanceInterval implements the ApplicabilityAssignmentItem <> (inherited from class KeyPerformanceIndicator)

ScheduledMaintenanceInterval special recommendations:

- An LSACandidate can have multiple scheduled maintenance interval values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of scheduledMaintenanceInterval (ie, to the respective instance of PropertyType. Refer to [Para 4.22](#)).
- An LSACandidate can have multiple scheduled maintenance interval values being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one 'specified' scheduled maintenance interval value, and one 'actual' scheduled maintenance interval value. These values must be distinguished using the valueDetermination attribute for the respective instance of scheduledMaintenanceInterval.

Note

valueDetermination is an attribute of the PropertyType primitive.

- If an instance of ScheduledMaintenanceInterval has multiple scheduledMaintenanceInterval values without any additional applicability statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first

4.9.3.5 MaintenanceFreeOperatingPeriod

The MaintenanceFreeOperatingPeriod <>attributeGroup>> is a specialization of KeyPerformanceIndicator, and represents the acceptable maintenance free operating period, where maintenance free operating period is the interval in which no maintenance actions occur.

MaintenanceFreeOperatingPeriod attributes:

- keyPerformanceIndicatorMethod (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorStatus (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorPercentile (inherited from class KeyPerformanceIndicator)
- maintenanceFreeOperatingPeriod (one or many)

MaintenanceFreeOperatingPeriod associations:

- An instance of MaintenanceFreeOperatingPeriod is always defined for a specific LSACandidate (inherited from class KeyPerformanceIndicator)

MaintenanceFreeOperatingPeriod interfaces:

- MaintenanceFreeOperatingPeriod implements the ApplicabilityAssignmentItem <>interface>> (inherited from class KeyPerformanceIndicator)

MaintenanceFreeOperatingPeriod special recommendations:

- An LSACandidate can have multiple maintenance free operating period values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of maintenanceFreeOperatingPeriod (ie, to the respective instance of PropertyType. Refer to [Para 4.22](#).
- An LSACandidate can have multiple maintenance free operating periods being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one 'specified' maintenance free operating period value, and one 'actual' maintenance free operating period value. These values must be distinguished using the valueDetermination attribute for the respective instance of maintenanceFreeOperatingPeriod.

Note

valueDetermination is an attribute of the PropertyType primitive.

- If an instance of MaintenanceFreeOperatingPeriod has multiple maintenanceFreeOperatingPeriod values without any additional applicability statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first

4.9.3.6 DownTime

The DownTime <>attributeGroup>> is a specialization of KeyPerformanceIndicator, and represents the acceptable mean down time, where mean down time is the time where an item is non-operational.

Downtime attributes:

- keyPerformanceIndicatorMethod (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorStatus (inherited from class KeyPerformanceIndicator)

- keyPerformanceIndicatorPercentile (inherited from class KeyPerformanceIndicator)
- downTime (one or many)

DownTime associations:

- An instance of DownTime is always defined for a specific LSACandidate (inherited from class KeyPerformanceIndicator)

DownTime interfaces:

- DownTime implements the ApplicabilityAssignmentItem <> (inherited from class KeyPerformanceIndicator)

DownTime special recommendations:

- An LSACandidate can have multiple down time values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of downTime (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).
- An LSACandidate can have multiple down time values being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one 'specified' down time value, and one 'actual' down time value. These values must be distinguished using the valueDetermination attribute for the respective instance of downtime.

Note

valueDetermination is an attribute of the PropertyType primitive.

- If an instance of DownTime has multiple downTime values without any additional applicability statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first

4.9.3.7 MaintenanceManHoursPerOperatingHour

The MaintenanceManHoursPerOperatingHour <> is a specialization of KeyPerformanceIndicator, and represents the acceptable maintenance man hours per operating hour, where maintenance man hours per operating hour is the ratio of maintenance man-hours expended to the operating interval (as defined by the measurement base) of the system/equipment.

MaintenanceManHoursPerOperatingHour attributes:

- keyPerformanceIndicatorMethod (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorStatus (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorPercentile (inherited from class KeyPerformanceIndicator)
- maintenanceManHoursPerOperatingHour (one or many)

MaintenanceManHoursPerOperatingHour associations:

- An instance of MaintenanceManHoursPerOperatingHour is always defined for a specific LSACandidate (inherited from class KeyPerformanceIndicator)

MaintenanceManHoursPerOperatingHour interfaces:

- MaintenanceManHoursPerOperatingHour implements the ApplicabilityAssignmentItem <> (inherited from class KeyPerformanceIndicator)

MaintenanceManHoursPerOperatingHour special recommendations:

- An LSACandidate can have multiple maintenance man hours per operating hour values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of

- maintenanceManHoursPerOperatingHour (ie, to the respective instance of **.PropertyType**). Refer to [Para 4.22](#).
- An LSACandidate can have multiple maintenance man hours per operating hour values being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one ‘specified’ maintenance man hours per operating hour value, and one ‘actual’ maintenance man hours per operating hour value. These values must be distinguished using the **valueDetermination** attribute for the respective instance of **maintenanceManHoursPerOperatingHour**.

Note

valueDetermination is an attribute of the **.PropertyType** primitive.

- If an instance of **MaintenanceManHoursPerOperatingHour** has multiple **maintenanceManHoursPerOperatingHour** values without any additional applicability statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first

4.9.3.8 MeanTimeBetweenUnscheduledRemoval

The **MeanTimeBetweenUnscheduledRemoval <>attributeGroup>** is a specialization of **KeyPerformanceIndicator**, and represents the total number of operational units (eg, miles, rounds, hours) divided by the total number of items removed from that system during a stated period of time. This term is defined to exclude removals as either being scheduled or performed in order to facilitate other maintenance and removals for product improvement.

MeanTimeBetweenUnscheduledRemoval attributes:

- **keyPerformanceIndicatorMethod** (inherited from class **KeyPerformanceIndicator**)
- **keyPerformanceIndicatorStatus** (inherited from class **KeyPerformanceIndicator**)
- **keyPerformanceIndicatorPercentile** (inherited from class **KeyPerformanceIndicator**)
- **meanTimeBetweenUnscheduledRemoval** (one or many)

MeanTimeBetweenUnscheduledRemoval associations:

- An instance of **MeanTimeBetweenUnscheduledRemoval** is always defined for a specific LSACandidate (inherited from class **KeyPerformanceIndicator**)

MeanTimeBetweenUnscheduledRemoval interfaces:

- **MeanTimeBetweenUnscheduledRemoval** implements the **ApplicabilityAssignmentItem <>interface>** (inherited from class **KeyPerformanceIndicator**)

MeanTimeBetweenUnscheduledRemoval special recommendations:

- An LSACandidate can have multiple mean time between unscheduled removal values depending on, for example operational environment. These values must be distinguished by the assignment of an **ApplicabilityStatement** to the respective instance of **meanTimeBetweenUnscheduledRemoval** value (ie, to the respective instance of **.PropertyType**). Refer to [Para 4.22](#).
- An LSACandidate can have multiple mean time between unscheduled removal values being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one ‘specified’ mean time between unscheduled removal value and one ‘actual’ mean time between unscheduled removal value. These values must be distinguished using the **valueDetermination** attribute for the respective instance of **meanTimeBetweenUnscheduledRemoval**.

Note

valueDetermination is an attribute of the **PropertyParams** primitive.

- If an instance of **MeanTimeBetweenUnscheduledRemoval** has multiple **meanTimeBetweenUnscheduledRemoval** values without any additional applicability

statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first

4.9.3.9 MeanTimeToRepair

The MeanTimeToRepair <>attributeGroup>> is a specialization of KeyPerformanceIndicator, and represents the total elapsed time for corrective maintenance divided by the total number of corrective maintenance actions during a given period of time.

MeanTimeToRepair attributes:

- keyPerformanceIndicatorMethod (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorStatus (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorPercentile (inherited from class KeyPerformanceIndicator)
- meanTimeToRepair (one or many)

MeanTimeToRepair associations:

- An instance of MeanTimeToRepair is always defined for a specific LSACandidate (inherited from class KeyPerformanceIndicator).

MeanTimeToRepair interfaces:

- MeanTimeToRepair implements the ApplicabilityAssignmentItem <>interface>> (inherited from class KeyPerformanceIndicator).

MeanTimeToRepair special recommendations:

- An LSACandidate can have multiple mean time to repair values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of meanTimeToRepair (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#). An LSACandidate can have multiple mean time to repair values being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one 'specified' mean time to repair value, and one 'actual' mean time to repair value. These values must be distinguished using the valueDetermination attribute for the respective instance of meanTimeToRepair.

Note

valueDetermination is an attribute of the PropertyType primitive.

- If an instance of MeanTimeToRepair has multiple meanTimeToRepair values without any additional applicability statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first

4.9.3.10 DirectMaintenanceCost

The DirectMaintenanceCost <>attributeGroup>> is a specialization of KeyPerformanceIndicator, and represents costs which include the shop maintenance man-hours, shop test man-hours, repair cost (incl. required material). Statistical values which represent the occurrence rate like Mean Time Between Failure (MTBF) and Mean Time Between Unscheduled Removal (MTBUR) must be considered for that cost element.

DirectMaintenanceCost attributes:

- keyPerformanceIndicatorMethod (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorStatus (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorPercentile (inherited from class KeyPerformanceIndicator)
- directMaintenanceCost (one or many)

DirectMaintenanceCost associations:

- An instance of DirectMaintenanceCost is always defined for a specific LSACandidate (inherited from class KeyPerformanceIndicator)

DirectMaintenanceCost interfaces:

- DirectMaintenanceCost implements the ApplicabilityAssignmentItem <<interface>> (inherited from class KeyPerformanceIndicator)

DirectMaintenanceCost special recommendations:

- An LSACandidate can have multiple direct maintenance cost values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of the directMaintenanceCost (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).
- An LSACandidate can have multiple direct maintenance cost values being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one 'specified' direct maintenance cost value, and one 'actual' direct maintenance cost value. These values must be distinguished using the valueDetermination attribute for the respective instance of directMaintenanceCost.

Note

valueDetermination is an attribute of the PropertyType primitive.

- If an instance of DirectMaintenanceCost has multiple directMaintenanceCost values without any additional applicability statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first

4.9.3.11 ShopProcessingTime

The ShopProcessingTime <<attributeGroup>> is a specialization of KeyPerformanceIndicator, and represents the duration from the start of the repair activities in the repair shop until the closing of the repair procedure without considering any shipping and delay times.

ShopProcessingTime attributes:

- keyPerformanceIndicatorMethod (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorStatus (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorPercentile (inherited from class KeyPerformanceIndicator)
- shopProcessingTime (one or many)

ShopProcessingTime associations:

- An instance of ShopProcessingTime is always defined for a specific LSACandidate (inherited from class KeyPerformanceIndicator)

ShopProcessingTime interfaces:

- ShopProcessingTime implements the ApplicabilityAssignmentItem <<interface>> (inherited from class KeyPerformanceIndicator)

ShopProcessingTime special recommendations:

- An LSACandidate can have multiple shop processing time values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of shopProcessingTime (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).
- An LSACandidate can have multiple shop processing time values being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one 'specified' shop processing time value, and one 'actual' shop processing time value. These values must be distinguished using the valueDetermination attribute for the respective instance of shopProcessingTime.

Note

valueDetermination is an attribute of the PropertyType primitive.

- If an instance of ShopProcessingTime has multiple shopProcessingTime values without any additional applicability statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first

4.9.3.12 FailuresPerOperatingHour

The FailuresPerOperatingHour <>attributeGroup>> is a specialization of KeyPerformanceIndicator, and represents the failure rate, expressed in failures per operating hour. The FailuresPerOperatingHour defines a specified value which will be the maximum acceptable failure behavior of a component.

FailuresPerOperatingHour attributes:

- keyPerformanceIndicatorMethod (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorStatus (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorPercentile (inherited from class KeyPerformanceIndicator)
- failuresPerOperatingHour (one or many)

FailuresPerOperatingHour associations:

- An instance of FailuresPerOperatingHour is always defined for a specific LSACandidate (inherited from class KeyPerformanceIndicator).

FailuresPerOperatingHour interfaces:

- FailuresPerOperatingHour implements the ApplicabilityAssignmentItem <>interface>> (inherited from class KeyPerformanceIndicator).

FailuresPerOperatingHour special recommendations:

- An LSACandidate can have multiple failures per operating hour values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instances of failuresPerOperatingHour (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).
- An LSACandidate can have multiple failures per operating hour values being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one 'specified' failures per operating hour value, and one 'actual' failures per operating hour value. These values must be distinguished using the valueDetermination attribute for the respective instance of failuresPerOperatingHour.

Note

valueDetermination is an attribute of the PropertyType primitive.

- If an instance of FailuresPerOperatingHour has multiple failuresPerOperatingHour values without any additional applicability statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first

4.9.3.13 ReplacementTime

The ReplacementTime <>attributeGroup>> is a specialization of KeyPerformanceIndicator, and represents the duration of the replacement of a (eg, faulty) component within any technical system by another (eg, new) component.

Note

The contracted/allocated/specified ReplacementTime defines a specified value which will be the maximum acceptable duration for component replacement. In practice, this value can be used to document for example, a customer's requirement to be able to perform 98% of all replacement tasks below a specified value of two hours (= maximum replacement time).

ReplacementTime attributes:

- keyPerformanceIndicatorMethod (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorStatus (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorPercentile (inherited from class KeyPerformanceIndicator)
- replacementTime (one or many)

ReplacementTime associations:

- An instance of ReplacementTime is always defined for a specific LSACandidate (inherited from class KeyPerformanceIndicator)

ReplacementTime interfaces:

- ReplacementTime implements the ApplicabilityAssignmentItem <> (inherited from class KeyPerformanceIndicator)

ReplacementTime special recommendations:

- An LSACandidate can have multiple replacement time values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of replacementTime (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#). An LSACandidate can have multiple replacement time values being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one ‘specified’ replacement time value, and one ‘actual’ replacement time value. These values must be distinguished using the valueDetermination attribute for the respective instance of replacementTime.

Note

valueDetermination is an attribute of the PropertyType primitive.

- If an instance of ReplacementTime has multiple replacementTime values without any additional applicability statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first.

4.9.3.14 LifeCycleCost

The LifeCycleCost <> is a specialization of KeyPerformanceIndicator, and represents the total calculated cost for an LSA Candidate throughout its planned life.

LifeCycleCost attributes:

- keyPerformanceIndicatorMethod (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorStatus (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorPercentile (inherited from class KeyPerformanceIndicator)
- lifeCycleCost (one or many)

LifeCycleCost associations:

- An instance of LifeCycleCost is always defined for a specific LSACandidate (inherited from class KeyPerformanceIndicator)

LifeCycleCost interfaces:

- LifeCycleCost implements the ApplicabilityAssignmentItem <> (inherited from class KeyPerformanceIndicator)

LifeCycleCost special recommendations:

- An LSACandidate can have multiple life cycle cost values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of lifeCycleCost (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).

- An LSACandidate can have multiple life cycle cost values being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one 'specified' life cycle cost value, and one 'actual' life cycle cost value. These values must be distinguished using the valueDetermination attribute for the respective instance of lifeCycleCost.

Note

valueDetermination is an attribute of the PropertyType primitive.

- If an instance of LifeCycleCost has multiple lifeCycleCost values without any additional applicability statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first.

4.9.3.15 MeanTimeBetweenFailure

The MTBF <>attributeGroup>> is a specialization of KeyPerformanceIndicator, where the MTBF is the total operational life of a population of an LSA Candidate divided by the total number of failures within the population during a particular measurement interval. The definition holds for time, rounds, miles, events, or other measure of life units.

MeanTimeBetweenFailure attributes:

- keyPerformanceIndicatorMethod (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorStatus (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorPercentile (inherited from class KeyPerformanceIndicator)
- meanTimeBetweenFailure (one or many)

MeanTimeBetweenFailure associations:

- An instance of MeanTimeBetweenFailure is always defined for a specific LSACandidate (inherited from class KeyPerformanceIndicator)
- An optional association with zero, one or many instances of the <>attributeGroup>> CorrectionFactor

MeanTimeBetweenFailure interfaces:

- MeanTimeBetweenFailure implements the ApplicabilityAssignmentItem <>interface>> (inherited from class KeyPerformanceIndicator)

MeanTimeBetweenFailure special recommendations:

- An LSACandidate can have multiple mean time between failure values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of meanTimeBetweenFailure (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).
- An LSACandidate can have multiple mean time between failure values being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one 'specified' mean time between failure value, and one 'actual' mean time between failure value. These values must be distinguished using the valueDetermination attribute for the respective instance of meanTimeBetweenFailure.

Note

valueDetermination is an attribute of the PropertyType primitive.

- If an instance of MeanTimeBetweenFailure has multiple meanTimeBetweenFailure values without any additional applicability statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first

4.9.3.16 FailureRate

The FailureRate <>attributeGroup>> is a specialization of KeyPerformanceIndicator, where the failure rate is the total number of failures within a population of an LSA Candidate item divided

by the total functional life of the population during the measurement interval. The definition holds for time, rounds, miles, cycles or other measures of life units.

FailureRate attributes:

- keyPerformanceIndicatorMethod (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorStatus (inherited from class KeyPerformanceIndicator)
- keyPerformanceIndicatorPercentile (inherited from class KeyPerformanceIndicator)
- failureRate (one or many)

FailureRate associations:

- An instance of FailureRate is always defined for a specific LSACandidate (inherited from class KeyPerformanceIndicator)
- An optional association with zero, one or many instances of the <>attributeGroup>> CorrectionFactor

FailureRate interfaces:

- FailureRate implements the ApplicabilityAssignmentItem <>interface>> (inherited from class KeyPerformanceIndicator)

FailureRate special recommendations:

- An LSACandidate can have multiple failure rate values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instances of failureRate (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).
- An LSACandidate can have multiple failure rate values being defined during different stages of the LSA candidate life cycle. For example, one and the same LSA candidate can have one 'specified' failure rate value, and one 'actual' failure rate value. These values must be distinguished using the valueDetermination attribute for the respective instance of failureRate.

Note

valueDetermination is an attribute of the PropertyType primitive.

- If an instance of FailureRate has multiple failureRate values without any additional applicability statement, or multiple values with the same applicability statement, these values are then to be interpreted as whichever is reached first

4.9.3.17 CorrectionFactor

The CorrectionFactor <>attributeGroup>> defines a correction factor for an associated failure rate or mean time between failure value. Correction factors can be defined for eg, the following situations:

- Usage of the associated LSACandidate under specific conditions, eg, environment which causes additional stress to the system (sand, extreme temperatures, salty environment)
- Usage of a PartAsDesigned at a specific location in a Product Breakdown

CorrectionFactor attributes:

- correctionFactor
- correctionFactorJustification (zero or one)
- correctionFactorDate (zero or one)

The CorrectionFactor class must implement the following <>interface>>:

- ApplicabilityAssignmentItem. Refer to [Para 4.22](#).

CorrectionFactor associations:

- An instance of CorrectionFactor is always associated with a specific instance of either MeanTimeBetweenFailure or FailureRate

CorrectionFactor special recommendations:

- A CorrectionFactor is always dependent upon eg, operational environment or location within a Product. This condition must be made explicit by the assignment of an ApplicabilityStatement. Refer to [Para 4.22](#).

4.9.4 UoF LSA Candidate - Additions to referenced class and interface definitions

4.9.4.1 PartAsDesigned

The PartAsDesigned class defined in UoF Part Definition must also implement the following additional <>interface>>:

- LSACandidate

4.9.4.2 BreakdownElementRevision

The BreakdownElementRevision class defined in UoF Breakdown Structure must also implement the following additional <>interface>>:

- LSACandidate

4.10 UoF LSA Candidate Analysis Activity

4.10.1 Overall description

The LSA Candidate Analysis Activity UoF, [Fig 27](#), supports the selection and justification of LSA activities to be performed for the respective LSA Candidate.

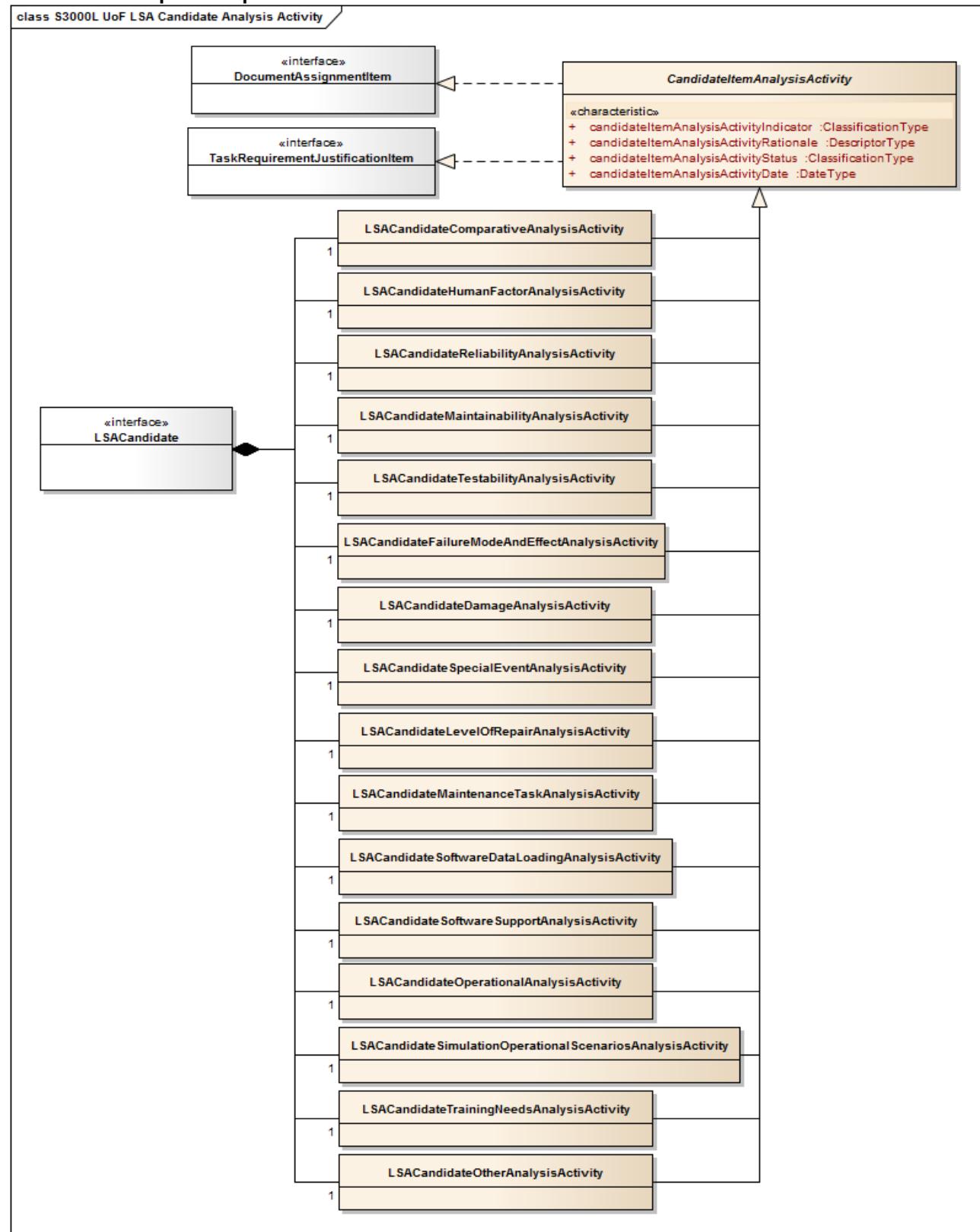
Note

LSA Candidate analysis activities are only to be documented for those PartAsDesigned and BreakdownElementRevisions that are identified as being LSA Candidates (full or partial).

Note

There is no requirement that all types of LSA activities must be performed for every single LSA Candidate, but there is a requirement to document the decision made for each combination of LSA Candidate and LSA activity.

4.10.2 Graphical representation



ICN-B6865-S3000L0224-003-00

Fig 27 S3000L UoF LSA Candidate Analysis Activity - class model

4.10.3 UoF LSA Candidate Analysis Activity - New class and interface definitions

4.10.3.1 CandidateItemAnalysisActivity

The CandidateItemAnalysisActivity class identifies the LSA activities that can be performed for the respective LSA Candidate, the rationale behind it, and the recorded status for the respective analysis activity.

Each identified LSA activity can also be associated with zero, one or many documents, eg, documents containing the results from the LSA activity.

Note

LSA activities are often selected during the LSA guidance conference.

Note

LSA activities can be defined for LSA Candidates at any level of indenture.

Note

The CandidateItemAnalysisActivity class is an abstract class, ie, the class itself will never be instantiated but any of its subclasses can. Subclasses of CandidateItemAnalysisActivity are:

- LSACandidateComparativeAnalysisActivity
- LSACandidateHumanFactorAnalysisActivity
- LSACandidateReliabilityAnalysisActivity
- LSACandidateMaintainabilityAnalysisActivity
- LSACandidateTestabilityAnalysisActivity
- LSACandidateFailureModeAndEffectAnalysisActivity
- LSACandidateDamageAnalysisActivity
- LSACandidateSpecialEventAnalysisActivity
- LSACandidateLevelOfRepairAnalysisActivity
- LSACandidateMaintenanceTaskAnalysisActivity
- LSACandidateSoftwareDataLoadingAnalysisActivity
- LSACandidateSoftwareSupportAnalysisActivity
- LSACandidateOperationalAnalysisActivity
- LSACandidateSimulationOperationalScenariosAnalysisActivity
- LSACandidateTrainingNeedsAnalysisActivity
- LSACandidateOtherAnalysisActivity

CandidateItemAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator
- candidateItemAnalysisActivityRationale
- candidateItemAnalysisActivityStatus
- candidateItemAnalysisActivityDate

Subclasses of CandidateItemAnalysisActivity class must also implement the following interfaces:

- The DocumentAssignmentItem <> which allows for any CandidateItemAnalysisActivity to refer to documents, or portions of documents, that contains eg, the analysis result. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <>. Refer to [Para 4.13](#).

4.10.3.2 LSACandidateComparativeAnalysisActivity

The LSACandidateComparativeAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding comparative analysis for the LSA Candidate under consideration.

Note

An instance of class LSACandidateComparativeAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateComparativeAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateComparativeAnalysisActivity associations:

- An association with the LSACandidate for which the LSA Candidate comparative analysis activity will be performed

The LSACandidateComparativeAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <<interface>>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <<interface>>. Refer to [Para 4.13](#).

4.10.3.3 LSACandidateHumanFactorAnalysisActivity

The LSACandidateHumanFactorAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding human factor analysis for the LSA Candidate under consideration.

Note

An instance of class LSACandidateHumanFactorAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateHumanFactorAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateHumanFactorAnalysisActivity associations:

- An association with the LSACandidate for which the LSA Candidate human factor analysis activity will be performed

The LSACandidateHumanFactorAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <<interface>>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <<interface>>. Refer to [Para 4.13](#).

4.10.3.4 LSACandidateReliabilityAnalysisActivity

The LSACandidateReliabilityAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding reliability analysis for the LSA Candidate under consideration.

Note

An instance of class LSACandidateReliabilityAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateReliabilityAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)

- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateReliabilityAnalysisActivity associations:

- An association with the LSACandidate for which the LSA Candidate reliability analysis activity will be performed

The LSACandidateReliabilityAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <<interface>>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <<interface>>. Refer to [Para 4.13](#).

4.10.3.5 LSACandidateMaintainabilityAnalysisActivity

The LSACandidateMaintainabilityAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding maintainability analysis for the LSA Candidate under consideration.

Note

An instance of class LSACandidateMaintainabilityAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateMaintainabilityAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateMaintainabilityAnalysisActivity associations:

- An association with the LSACandidate for which the LSA candidate maintainability analysis activity will be performed

The LSACandidateMaintainabilityAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <<interface>>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <<interface>>. Refer to [Para 4.13](#).

4.10.3.6 LSACandidateTestabilityAnalysisActivity

The LSACandidateTestabilityAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding testability analysis for the LSA Candidate under consideration.

Note

An instance of class LSACandidateTestabilityAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateTestabilityAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateTestabilityAnalysisActivity associations:

- An association with the LSACandidate for which the LSA candidate testability analysis activity will be performed

The LSACandidateTestabilityAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <>. Refer to [Para 4.13](#).

4.10.3.7 LSACandidateFailureModeAndEffectAnalysisActivity

The LSACandidateFailureModeAndEffectAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding failure mode and effect analysis (LSA FMEA) for the LSA Candidate under consideration.

Note

An instance of class LSACandidateFailureModeAndEffectAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateFailureModeAndEffectAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateFailureModeAndEffectAnalysisActivity associations:

- An association with the LSACandidate for which the LSA candidate failure mode and effect analysis activity will be performed

The LSACandidateFailureModeAndEffectAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <>. Refer to [Para 4.13](#).

4.10.3.8 LSACandidateDamageAnalysisActivity

The LSACandidateDamageAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding damage analysis for the LSA Candidate under consideration.

Note

An instance of LSACandidateDamageAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateDamageAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateDamageAnalysisActivity associations:

- An association with the LSACandidate for which the LSA candidate damage analysis activity will be performed

The LSACandidateDamageAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <>. Refer to [Para 4.20](#).

- The TaskRequirementJustificationItem <>. Refer to [Para 4.13](#).

4.10.3.9 LSACandidateSpecialEventAnalysisActivity

The LSACandidateSpecialEventAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding special events analysis for the LSA Candidate under consideration.

Note

An instance of class LSACandidateSpecialEventAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateSpecialEventAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateSpecialEventAnalysisActivity associations:

- An association with the LSACandidate for which the LSA candidate special event analysis activity will be performed

The LSACandidateSpecialEventAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <>. Refer to [Para 4.13](#).

4.10.3.10 LSACandidateLevelOfRepairAnalysisActivity

The LSACandidateLevelOfRepairAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding level of repair analysis (LORA) for the LSA Candidate under consideration.

Note

An instance of class LSACandidateLevelOfRepairAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateLevelOfRepairAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateLevelOfRepairAnalysisActivity associations:

- An association with the LSACandidate for which the LSA candidate level of repair analysis activity will be performed

The LSACandidateLevelOfRepairAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <>. Refer to [Para 4.13](#).

4.10.3.11 LSACandidateMaintenanceTaskAnalysisActivity

The LSACandidateMaintenanceTaskAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding maintenance task analysis for the LSA Candidate under consideration.

Note

An instance of class LSACandidateMaintenanceTaskAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateMaintenanceTaskAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateMaintenanceTaskAnalysisActivity associations:

- An association with the LSACandidate for which the LSA candidate maintenance task analysis activity will be performed

The LSACandidateMaintenanceTaskAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <<interface>>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <<interface>>.. Refer to [Para 4.13](#).

4.10.3.12 LSACandidateSoftwareDataLoadingAnalysisActivity

The LSACandidateSoftwareDataLoadingAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding software data loading analysis for the LSA Candidate under consideration.

Note

An instance of class LSACandidateSoftwareDataLoadingAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateSoftwareDataLoadingAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateSoftwareDataLoadingAnalysisActivity associations:

- An association with the LSACandidate for which the LSA candidate software data loading analysis activity will be performed

The LSACandidateSoftwareDataLoadingAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <<interface>>. Refer to [Para 4.20](#). The TaskRequirementJustificationItem <<interface>>. Refer to [Para 4.13](#).

4.10.3.13 LSACandidateSoftwareSupportAnalysisActivity

The LSACandidateSoftwareSupportAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding software support analysis for the LSA Candidate under consideration.

Note

An instance of class LSACandidateSoftwareSupportAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateSoftwareSupportAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)

- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateSoftwareSupportAnalysisActivity associations:

- An association with the LSACandidate for which the LSA candidate software support analysis activity will be performed

The LSACandidateSoftwareSupportAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <<interface>>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <<interface>>. Refer to [Para 4.13](#).

4.10.3.14 LSACandidateOperationalAnalysisActivity

The LSACandidateOperationalAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding operational analysis for the LSA Candidate under consideration.

Note

An instance of class LSACandidateOperationalAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateOperationalAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateOperationalAnalysisActivity associations:

- An association with the LSACandidate for which the LSA candidate operational analysis activity will be performed

The LSACandidateOperationalAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <<interface>>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <<interface>>. Refer to [Para 4.13](#).

4.10.3.15 LSACandidateSimulationOperationalScenariosAnalysisActivity

The LSACandidateSimulationOperationalScenariosAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding simulation of operational scenarios for the LSA Candidate under consideration.

Note

An instance of class LSACandidateSimulationOperationalScenariosAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateSimulationOperationalScenariosAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateSimulationOperationalScenariosAnalysisActivity associations:

- An association with the LSACandidate for which the LSA candidate simulation operational scenarios analysis activity will be performed

The LSACandidateSimulationOperationalScenariosAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <>. Refer to [Para 4.13](#).

4.10.3.16 LSACandidateTrainingNeedsAnalysisActivity

The LSACandidateTrainingNeedsAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding training needs analysis for the LSA Candidate under consideration.

Note

An instance of class LSACandidateTrainingNeedsAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateTrainingNeedsAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateTrainingNeedsAnalysisActivity associations:

- An association with the LSACandidate for which the LSA candidate training needs analysis activity will be performed

The LSACandidateTrainingNeedsAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <>. Refer to [Para 4.13](#).

4.10.3.17 LSACandidateOtherAnalysisActivity

The LSACandidateOtherAnalysisActivity class is a specialization of CandidateItemAnalysisActivity, and records the decision made regarding additional types of analysis activities that needs to be performed for the LSA Candidate under consideration, apart from those defined in [Para 4.10.3.1](#) to [Para 4.10.3.16](#).

Note

An instance of class LSACandidateOtherAnalysisActivity has no meaning by itself, but only in the context of a specific LSACandidate.

LSACandidateOtherAnalysisActivity attributes:

- candidateItemAnalysisActivityIndicator (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityRationale (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityStatus (inherited from class CandidateItemAnalysisActivity)
- candidateItemAnalysisActivityDate (inherited from class CandidateItemAnalysisActivity)

LSACandidateOtherAnalysisActivity associations:

- An association with the LSACandidate for which the LSA candidate other analysis activity will be performed

The LSACandidateOtherAnalysisActivity class must also implement the following interfaces (inherited from class CandidateItemAnalysisActivity):

- The DocumentAssignmentItem <<interface>>. Refer to [Para 4.20](#).
- The TaskRequirementJustificationItem <<interface>>. Refer to [Para 4.13](#).

4.10.4 UoF LSA Candidate Analysis Activity - Additions to referenced class and interface definitions

4.10.4.1 LSACandidate

Any class that implements the LSACandidate <<interface>> defined in UoF LSA Candidate must also implement the following additional associations:

- An association with LSACandidateComparativeAnalysisActivity
- An association with LSACandidateHumanFactorAnalysisActivity
- An association with LSACandidateReliabilityAnalysisActivity
- An association with LSACandidateMaintainabilityAnalysisActivity
- An association with LSACandidateTestabilityAnalysisActivity
- An association with LSACandidateFailureModeAndEffectAnalysisActivity
- An association with LSACandidateDamageAnalysisActivity
- An association with LSACandidateSpecialEventAnalysisActivity
- An association with LSACandidateLevelOfRepairAnalysisActivity
- An association with LSACandidateMaintenanceTaskAnalysisActivity
- An association with LSACandidateSoftwareDataLoadingAnalysisActivity
- An association with LSACandidateSoftwareSupportAnalysisActivity
- An association with LSACandidateOperationalAnalysisActivity
- An association with LSACandidateSimulationOperationalScenariosAnalysisActivity
- An association with LSACandidateTrainingNeedsAnalysisActivity
- An association with LSACandidateOtherAnalysisActivity

Note

Only those instances of BreakdownElementRevision and PartAsDesigned (ie, instances of those classes that implement the LSACandidate <<interface>>) that are selected as LSA Candidates must identify analysis activities to be performed.

4.11 UoF LSA FMEA

4.11.1 Overall description

The LSA FMEA UoF, [Fig 28](#), supports the recording of the results from the following LSA activities:

- LSA FMEA
- Testability analysis

Hardware items that have been identified as LSA candidates are the starting point for the LSA FMEA.

Technical failure modes causing one and the same combination of failure mode signature and maintenance procedure will be grouped into a single LSA failure mode. A failure mode signature is made up from all registered alarms and observations which are expected for the failure mode under consideration. Registered alarms are always detected by a detection mean, whilst observations can either be detected by a detection mean and eg, be presented on a display panel or be observed through a failure mode effect such as loss of function.

Each LSA failure mode will be associated with a requirement for a rectifying task, where the task requirement represents the need for an identified maintenance procedure. Each LSA failure mode can also be associated with a requirement for a troubleshooting procedure.

The target item (hardware element or part) for which the task requirement is identified need not be the same hardware element or part as for which the original technical failure modes was identified.

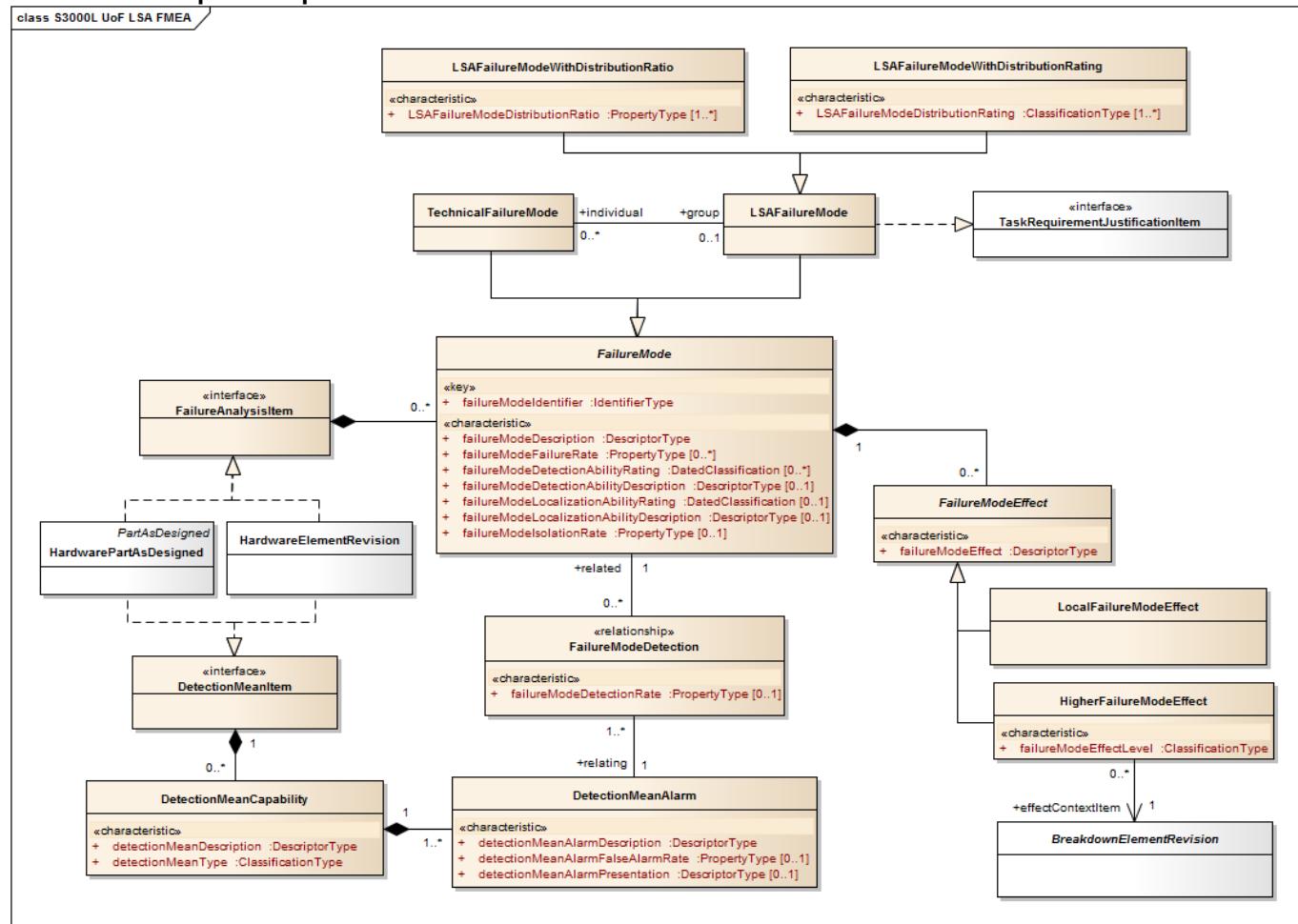
While grouping, failure rates for the item under analysis can either be distributed to the respective LSA failure mode or be defined explicitly for each LSA failure mode (attribute failureModeFailureRate). A distributed failure rate can be defined either by ratio, or by qualitative measures (rating).

Each possible failure mode can be detected by, either an automatic detection mean, Built-in-Test (BIT), or by other functional/physical symptoms. The failure mode effects defined in the technical FMEA/FMECA can be considered as a guideline for functional/physical symptoms. The ability to localize the unit that fails (without ambiguity) can be rated using qualitative measures.

Note

The data model does not mandate the recording of technical failure modes, ie, a project can choose to just identify LSA failure modes related to the need for failure detection (failure mode signature) and removal of Line Replaceable Unit (LRU) or Shop Replaceable Unit (SRU).

4.11.2 Graphical representation



ICN-B6865-S3000L0261-001-00

Fig 28 S3000L UoF LSA FMEA - class model

4.11.3 UoF LSA FMEA - New class and interface definitions

4.11.3.1 FailureAnalysisItem

The FailureAnalysisItem <<interface>> is implemented by classes that represent hardware items.

Classes that implement the FailureAnalysisItem <<interface>> are:

- HardwareElementRevision
- HardwarePartAsDesigned

Classes that implement the FailureAnalysisItem <<interface>> must implement the following association:

- An optional aggregation association with one or many instances of FailureMode

4.11.3.2 FailureMode

The FailureMode class supports the definition of possible failure modes for a specific hardware item.

Note

An instance of class FailureMode has no meaning by itself, but only in the context of a hardware item (FailureAnalysisItem).

Note

The FailureMode class is an abstract class, ie, the class itself will never be instantiated but any of its subclasses can. Subclasses of FailureMode are:

- TechnicalFailureMode
- LSAFailureMode.

FailureMode attributes:

- failureModelIdentifier
- failureModeDescription
- failureModeFailureRate (zero, one or many)
- failureModeDetectionAbilityRating (zero or one)
- failureModeDetectionAbilityDescription (zero or one)
- failureModeLocalizationAbilityRating (zero or one)
- failureModeLocalizationAbilityDescription (zero or one)
- failureModeIsolationRate (zero or one)

Note

The failureModeDetectionAbilityRating and failureModeLocalizationAbilityRating attributes are both of data type DatedClassification, ie, both ratings must have an associated date representing the date when the rating was performed.

FailureMode associations:

- An association with the hardware item (HardwarePartAsDesigned or HardwareElementRevision) for which the failure mode has been identified (via the FailureAnalysisItem <>interface>)
- An optional association with zero, one or many instances of FailureModeEffect that can be observed in order to detect the failure mode
- An optional association with zero, one or many instances of DetectionMeanAlarm that can be observed and/or recorded in order to detect the failure mode (via instances of the FailureModeDetection <>relationship> class)

FailureMode special recommendations:

- The failureModeFailureRate attribute can have multiple failure rate values, eg, depending on operational environment. These must be distinguished by the assignment of ApplicabilityStatement to the instances of failureModeFailureRate value (ie, the instances of PropertyType). Refer to [Para 4.22](#).

4.11.3.3

TechnicalFailureMode

The TechnicalFailureMode class is a specialization of FailureMode, and supports the definition of technical failure modes for a specific hardware item.

Note

An instance of class TechnicalFailureMode has no meaning by itself, but only in the context of a hardware item.

TechnicalFailureMode attributes:

- failureModelIdentifier (inherited from class FailureMode)
- failureModeDescription (inherited from class FailureMode)
- failureModeFailureRate (inherited from class FailureMode)
- failureModeDetectionAbilityRating (inherited from class FailureMode)

- failureModeDetectionAbilityDescription (inherited from class FailureMode)
- failureModeLocalizationAbilityRating (inherited from class FailureMode)
- failureModeLocalizationAbilityDescription (inherited from class FailureMode)
- failureModelIsolationRate (inherited from class FailureMode)

Note

The failureModeDetectionAbilityRating and failureModeLocalizationAbilityRating attributes are both of data type DatedClassification, ie, both ratings must have an associated date representing the date when the rating was performed.

TechnicalFailureMode associations:

- An association with the hardware item (HardwarePartAsDesigned or HardwareElementRevision) for which the technical failure mode has been identified (inherited from class FailureMode)
- An optional association with zero, one or many instances of FailureModeEffect that can be observed in order to detect the technical failure mode (inherited from class FailureMode)
- An optional association with zero, one or many instances of DetectionMeanAlarm that can be observed and/or recorded in order to detect the technical failure mode (inherited from class FailureMode)
- An optional association with zero or one LSAFailureMode (for grouping of technical failure modes)

TechnicalFailureMode special recommendations:

- The failureModeFailureRate attribute can have multiple failure rate values, eg, depending on operational environment. These must be distinguished by the assignment of ApplicabilityStatement to the instances of failureModeFailureRate value (ie, the instances of PropertyType). Refer to [Para 4.22](#).

4.11.3.4 LSAFailureMode

The LSAFailureMode class is a specialization of FailureMode, and supports the grouping of identified technical failure modes which leads to the same combination of failure mode signature (troubleshooting) and maintenance procedure.

Note

An instance of class LSAFailureMode has no meaning by itself, but only in the context of a hardware item.

The LSAFailureMode has two specializations which can be used to record the probability for an individual LSA failure mode to occur in relation to the entire population of LSA failure modes identified for the related hardware item (HardwarePartAsDesigned or HardwareElementRevision). These specializations are LSAFailureModeWithDistributionRatio and LSAFailureModeWithDistributionRating, respectively.

LSAFailureMode attributes:

- failureModelIdentifier (inherited from class FailureMode)
- failureModeDescription (inherited from class FailureMode)
- failureModeFailureRate (inherited from class FailureMode)
- failureModeDetectionAbilityRating (inherited from class FailureMode)
- failureModeDetectionAbilityDescription (inherited from class FailureMode)
- failureModeLocalizationAbilityRating (inherited from class FailureMode)
- failureModeLocalizationAbilityDescription (inherited from class FailureMode)
- failureModelIsolationRate (inherited from class FailureMode)

Note

The failureModeDetectionAbilityRating and failureModeLocalizationAbilityRating attributes are both of data type DatedClassification, ie, both ratings must have an associated date representing the date when the rating was performed.

LSAFailureMode associations:

- An association with the hardware item (HardwarePartAsDesigned or HardwareElementRevision) for which the LSA failure mode has been identified (inherited from class FailureMode)
- An optional association with zero, one or many instances of FailureModeEffect that can be observed in order to detect the LSA failure mode (inherited from class FailureMode)
- An optional association with zero, one or many instances of DetectionMeanAlarm that can be observed and/or recorded in order to detect the LSA failure mode (inherited from class FailureMode)
- An optional association with zero, one or many instances of TechnicalFailureMode which are grouped into the LSAFailureMode

LSAFailureMode interfaces:

- LSAFailureMode implements the TaskRequirementJustificationItem <<interface>>

LSAFailureMode special recommendations:

- The failureModeFailureRate attribute can have multiple failure rate values, eg, depending on operational environment. These must be distinguished by the assignment of ApplicabilityStatement to the instances of failureModeFailureRate value (ie, the instances of PropertyType). Refer to [Para 4.22](#).

4.11.3.5 LSAFailureModeWithDistributionRatio

The LSAFailureModeWithDistributionRatio class is a specialization of LSAFailureMode, and records the ratio for an individual LSA failure mode to occur in relation to the entire population of LSA failure modes identified for the related hardware item (HardwarePartAsDesigned or HardwareElementRevision).

Note

An instance of class LSAFailureModeWithDistributionRatio has no meaning by itself, but only in the context of a specific hardware item.

LSAFailureModeWithDistributionRatio attributes:

- failureModeIdentifier (inherited from class FailureMode)
- failureModeDescription (inherited from class FailureMode)
- failureModeFailureRate (inherited from class FailureMode)
- failureModeDetectionAbilityRating (inherited from class FailureMode)
- failureModeDetectionAbilityDescription (inherited from class FailureMode)
- failureModeLocalizationAbilityRating (inherited from class FailureMode)
- failureModeLocalizationAbilityDescription (inherited from class FailureMode)
- failureModeIsolationRate (inherited from class FailureMode)
- LSAFailureModeDistributionRatio

Note

The failureModeDetectionAbilityRating and failureModeLocalizationAbilityRating attributes are both of data type DatedClassification, ie, both ratings must have an associated date representing the date when the rating was performed.

LSAFailureModeWithDistributionRatio associations:

- An association with the hardware item (HardwarePartAsDesigned or HardwareElementRevision) for which the LSAFailureModeWithDistributionRatio has been identified (inherited from class FailureMode)
- An optional association with zero, one or many instances of FailureModeEffect that can be observed in order to detect the LSAFailureModeWithDistributionRatio (inherited from class FailureMode)
- An optional association with zero, one or many instances of DetectionMeanAlarm that can be observed and/or recorded in order to detect the LSAFailureModeWithDistributionRatio (inherited from class FailureMode)
- An optional association with zero, one or many instances of TechnicalFailureMode which are grouped into the LSAFailureModeWithDistributionRatio (inherited from class LSAFailureMode)

LSAFailureModeWithDistributionRatio interfaces:

- LSAFailureModeWithDistributionRatio implements the TaskRequirementJustificationItem <>interface>> (inherited from class LSAFailureMode)

LSAFailureModeWithDistributionRatio special recommendations:

- The failureModeFailureRate attribute must not be populated for an instance of LSAFailureModeWithDistributionRatio
- The LSAFailureModeDistributionRatio attribute can have multiple distribution ratio values, eg, depending on operational environment. These must be distinguished by the assignment of ApplicabilityStatement to the instances of LSAFailureModeDistributionRatio value (ie, the instances of PropertyType). Refer to [Para 4.22](#).

Note

The sum of distribution ratios applicable under the same conditions must always equal one for each item under analysis.

4.11.3.6 LSAFailureModeWithDistributionRating

The LSAFailureModeWithDistributionRating class is a specialization of LSAFailureMode, and records the rating for the occurrence of an individual LSA failure mode in relation to the entire population of LSA failure modes identified for the related hardware item (HardwarePartAsDesigned or HardwareElementRevision).

Note

An instance of class LSAFailureModeWithDistributionRating has no meaning by itself, but only in the context of a specific hardware item.

LSAFailureModeWithDistributionRating attributes:

- failureModeIdentifier (inherited from class FailureMode)
- failureModeDescription (inherited from class FailureMode)
- failureModeFailureRate (inherited from class FailureMode)
- failureModeDetectionAbilityRating (inherited from class FailureMode)
- failureModeDetectionAbilityDescription (inherited from class FailureMode)
- failureModeLocalizationAbilityRating (inherited from class FailureMode)
- failureModeLocalizationAbilityDescription (inherited from class FailureMode)
- failureModeIsolationRate (inherited from class FailureMode)
- LSAFailureModeDistributionRating

Note

The failureModeDetectionAbilityRating and failureModeLocalizationAbilityRating attributes are both of data type DatedClassification, ie, both ratings must have an associated date representing the date when the rating was performed.

LSAFailureModeWithDistributionRating associations:

- An association with the hardware item (HardwarePartAsDesigned or HardwareElementRevision) for which the LSAFailureModeWithDistributionRating has been identified (inherited from class FailureMode)
- An optional association with zero, one or many instances of FailureModeEffect that can be observed in order to detect the LSAFailureModeWithDistributionRating (inherited from class FailureMode)
- An optional association with zero, one or many instances of DetectionMeanAlarm that can be observed and/or recorded in order to detect the LSAFailureModeWithDistributionRating (inherited from class FailureMode)
- An optional association with zero, one or many instances of TechnicalFailureMode which are grouped into the LSAFailureModeWithDistributionRating (inherited from class LSAFailureMode)

LSAFailureModeWithDistributionRating interfaces:

- LSAFailureModeWithDistributionRating implements the TaskRequirementJustificationItem <>interface>> (inherited from class LSAFailureMode)

LSAFailureModeWithDistributionRating special recommendations:

- The failureModeFailureRate attribute must not be populated for an instance of LSAFailureModeWithDistributionRating
- The LSAFailureModeDistributionRating attribute can have multiple distribution rating values, eg, depending on operational environment. These must be distinguished by the assignment of ApplicabilityStatement to the instances of LSAFailureModeDistributionRating value (ie, the instances of ClassificationType). Refer to [Para 4.22](#).

4.11.3.7 FailureModeEffect

The FailureModeEffect class defines the consequences of an identified failure mode and its effect on the local/next higher/end item operation, function, or status.

Note

An instance of class FailureModeEffect has no meaning by itself, but only in the context of an identified FailureMode.

Note

The FailureModeEffect class is an abstract class, ie, an instantiation of FailureModeEffect must be done using any of its specializations:

- LocalFailureModeEffect
- HigherFailureModeEffect

FailureModeEffect attributes:

- failureModeEffect

FailureModeEffect associations:

- An association with the FailureMode that causes the identified FailureModeEffect

4.11.3.8 LocalFailureModeEffect

The LocalFailureModeEffect class is a specialization of FailureModeEffect, and identifies the consequences of each postulated failure/damage mode affecting the hardware item (HardwarePart or HardwareElementRevision) under analysis. It is possible for the local effect to be the failure mode itself.

Note

An instance of class LocalFailureModeEffect has no meaning by itself, but only in the context of an identified FailureMode.

LocalFailureModeEffect attributes:

- failureModeEffect (inherited from class FailureModeEffect)

LocalFailureModeEffect associations:

- An association with the FailureMode that causes the identified LocalFailureModeEffect (inherited from class FailureModeEffect)

4.11.3.9 HigherFailureModeEffect

The HigherFailureModeEffect class is a specialization of FailureModeEffect, and identifies the consequences of each failure/damage mode affecting the next higher indenture level, or the essential functions affecting system/equipment operating capability and mission completion capability.

Note

An instance of class HigherFailureModeEffect has no meaning by itself, but only in the context of an identified FailureMode.

HigherFailureModeEffect attributes:

- failureModeEffect (inherited from class FailureModeEffect)
- failureModeEffectLevel

HigherFailureModeEffect associations:

- An association with the FailureMode that causes the identified HigherFailureModeEffect (inherited from class FailureModeEffect)
- An association with the BreakdownElementRevision that represents a higher level indentured equipment, system or end item

4.11.3.10 DetectionMeanItem

The DetectionMeanItem <> is implemented by those classes whose instances can be used for the detection and/or localization of one or many failure modes.

Classes that implements the DetectionMeanItem <> are:

- HardwareElementRevision
- HardwarePartAsDesigned

Classes that implement the DetectionMeanItem <> must also implement the following association:

- An optional association with one or many instances of DetectionMeanCapability

4.11.3.11 DetectionMeanCapability

The DetectionMeanCapability class supports the definition and description of the detection mean capabilities realized by a specific hardware item.

Note

An instance of class DetectionMeanCapability has no meaning by itself, but only in the context of a specific hardware item (HardwarePartAsDesigned or HardwareElementRevision).

DetectionMeanCapability attributes:

- detectionMeanDescription
- detectionMeanType

DetectionMeanCapability associations:

- An association with the hardware item (HardwarePartAsDesigned or HardwareElementRevision) for which the DetectionMeanCapability has been defined
- An association with one or many instances of DetectionMeanAlarm implemented by the hardware item (DetectionMeanItem) under consideration

4.11.3.12 DetectionMeanAlarm

The DetectionMeanAlarm class supports the definition and description of alarms being implemented by a specific detection mean.

Note

An instance of class DetectionMeanAlarm has no meaning by itself, but only in the context of a specific DetectionMeanCapability.

DetectionMeanAlarm attributes:

- detectionMeanAlarmDescription
- detectionMeanFalseAlarmRate (zero or one)
- detectionMeanAlarmPresentation (zero or one)

DetectionMeanCapability associations:

- An association with the DetectionMeanCapability for which the DetectionMeanAlarm has been defined
- An optional association with zero, one or many instances of FailureMode which can be observed, detected and/or localized by the DetectionMeanAlarm under consideration (via the FailureModeDetection <> relationship class)

4.11.3.13 FailureModeDetection

The FailureModeDetection <> relationship class defines an association between an instance of DetectionMeanAlarm and an instance of FailureMode which can be observed, detected and/or localized by the detection mean alarm.

FailureModeDetection attribute:

- failureModeDetectionRate (zero or one)

FailureModeDetection associations:

- (relating) The instance of DetectionMeanAlarm that can support the observation, detection and/or localization of the related FailureMode
- (related) The instance of FailureMode that can be observed, detected and localized by the relating DetectionMeanAlarm

Note

There is one instance of FailureModeDetection per combination of DetectionMeanAlarm and FailureMode.

4.11.4 UoF LSA FMEA - Additions to referenced class and interface definitions

4.11.4.1 BreakdownElementRevision

The BreakdownElementRevision class defined in UoF Breakdown Structure must also implement the following additional associations:

- An optional association with one or many instances of HigherFailureModeEffect

4.11.4.2 HardwarePartAsDesigned

The HardwarePartAsDesigned class defined in UoF Part Definition must also implement the following additional interfaces:

- FailureAnalysisItem
- DetectionMeanItem

4.11.4.3 HardwareElementRevision

The HardwareElementRevision class defined in UoF Breakdown Element Realization must also implement the following additional interfaces:

- FailureAnalysisItem
- DetectionMeanItem

4.12 UoF Special Events and Damage

4.12.1 Overall description

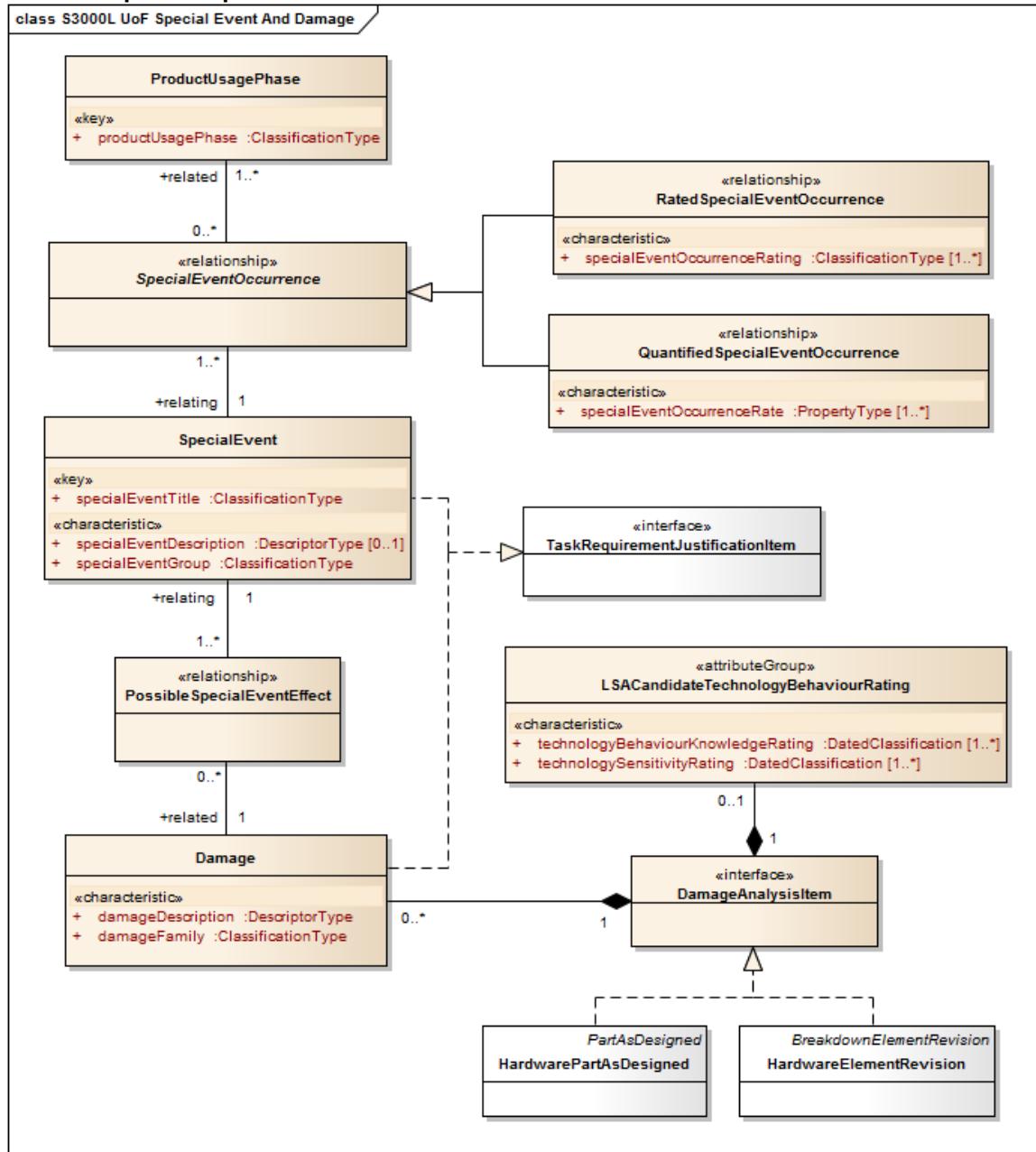
The Special Events and Damage UoF, [Fig 29](#), support the recording of the results from the following LSA activities:

- Special event analysis
- Damage analysis

Special events can be due to external or internal causes, eg, natural phenomenon or unusual use. The probability of an event occurring is related to the usage phase of the Product in scope for the LSA program, eg, transportation, operation, maintenance.

Each possible event can affect one or many LSA candidates and result in one or many damages. Identification of potential damages can also include a special analysis for each affected LSA candidate with respect to the knowledge of the technology used. This is referred to as technology behavior rating.

4.12.2 Graphical representation



ICN-B6865-S3000L0262-001-00

Fig 29 S3000L UoF Special Events and Damage - class model

4.12.3 UoF Special Events and Damage - New class and interface definitions

4.12.3.1 DamageAnalysisItem

The DamageAnalysisItem <<interface>> is implemented by classes that represent hardware items.

Classes that implements the DamageAnalysisItem <<interface>> are:

- HardwareElementRevision
- HardwarePartAsDesigned

Classes that implement the DamageAnalysisItem <<interface>> must also implement the following associations:

- An optional association with an instance of the LSACandidateTechnologyBehaviourRating <<attributeGroup>>
- An optional association with one or many instances of Damage

4.12.3.2 LSACandidateTechnologyBehaviourRating

The LSACandidateTechnologyBehaviourRating <<attributeGroup>> describes technologies used and its characteristics regarding knowledge of its behavior.

Note

An instance of class LSACandidateTechnologyBehaviourRating has no meaning by itself, but only in the context of a hardware item.

LSACandidateTechnologyBehaviourRating attributes:

- technologyBehaviourKnowledgeRating (one or many)
- technologySensitivityRating (one or many)

Note

Both the technologyBehaviourKnowledgeRating and the technologySensitivityRating attributes are of data type DatedClassification, ie, both ratings must have an associated date defining the date when the rating was performed. A hardware item can also have multiple ratings over time.

LSACandidateTechnologyBehaviourRating association:

- An association with the hardware item (HardwarePartAsDesigned or HardwareElementRevision) for which the technology behavior rating has been performed (via the DamageAnalysisItem <<interface>>)

4.12.3.3 Damage

Damage defines the loss or reduction of functionality caused by a special event (induced failure).

Damage attributes:

- damageDescription
- damageFamily

Damage associations:

- An association with the hardware item (HardwarePartAsDesigned or HardwareElementRevision) for which the damage mode has been identified (via the DamageAnalysisItem <<interface>>)
- An optional association with zero, one or many SpecialEvents that can cause the Damage (via the PossibleSpecialEventEffect <<relationship>> class)

Damage interfaces:

- Damage implements the TaskRequirementJustificationItem <<interface>>. This can document the requirement for a standard repair procedure and/or inspection

4.12.3.4 PossibleSpecialEventEffect

The PossibleSpecialEventEffect <<relationship>> class defines an association between an instance of SpecialEvent and the Damage which can be caused by the special event.

PossibleSpecialEventEffect associations:

- (relating) The instance of SpecialEvent that can cause the related Damage

- (related) The instance of Damage that can be caused by the relating SpecialEvent

Note

There is one instance of PossibleSpecialEventEffect per combination of SpecialEvent and Damage.

4.12.3.5 SpecialEvent

The SpecialEvent class identifies special events that can occur during different usage phases of the product, which can lead to damages on one or many hardware items (HardwarePartAsDesigned or HardwareElementRevision).

SpecialEvent attributes:

- specialEventTitle
- specialEventDescription (zero or one)
- specialEventGroup

SpecialEvent associations:

- An association with one or many instances of Damage which can be caused by the special event (via the PossibleSpecialEventEffect <<relationship>> class)
- An association with one or many instances of ProductUsagePhase, defining when the special event can occur (via the SpecialEventOccurrence <<relationship>> class)

SpecialEvent interfaces:

- SpecialEvent implements the TaskRequirementJustificationItem <<interface>>. This can document the requirement for an inspection procedure.

4.12.3.6 SpecialEventOccurrence

The SpecialEventOccurrence <<relationship>> class defines an association between an instance of SpecialEvent and an instance of ProductUsagePhase during which the special event can occur.

Note

The SpecialEventOccurrence class is an abstract class, ie, an instantiation of SpecialEventOccurrence must be either:

- RatedSpecialEventOccurrence
- QuantifiedSpecialEventOccurrence

Note

RatedSpecialEventOccurrence and QuantifiedSpecialEventOccurrence respectively, mandate the recording of the frequency for which the associated special event is expected to occur during a specific product usage phase, either by rating or by ratio.

SpecialEventOccurrence associations:

- (relating) The SpecialEvent that can occur during the related ProductUsagePhase
- (related) The ProductUsagePhase during which the relating SpecialEvent can occur

Note

There is one instance of SpecialEventOccurrence per combination of SpecialEvent and ProductUsagePhase.

4.12.3.7 RatedSpecialEventOccurrence

The RatedSpecialEventOccurrence <<relationship>> class is a specialization of the SpecialEventOccurrence class.

RatedSpecialEventOccurrence attributes:

- specialEventOccurrenceRating

RatedSpecialEventOccurrence associations:

- (relating) The SpecialEvent that can occur during the related ProductUsagePhase (inherited from class SpecialEventOccurrence)
- (related) The ProductUsagePhase during which the relating SpecialEvent can occur (inherited from class SpecialEventOccurrence)

RatedSpecialEventOccurrence special recommendations:

- The specialEventOccurrenceRating attribute can have multiple occurrence ratings depending on, for example operational environment. These ratings must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of specialEventOccurrenceRating ie, to the respective instance of ClassificationType. Refer to [Para 4.22](#).

4.12.3.8 QuantifiedSpecialEventOccurrence

The QuantifiedSpecialEventOccurrence <>relationship>> class is a specialization of the SpecialEventOccurrence class.

QuantifiedSpecialEventOccurrence attributes:

- specialEventOccurrenceRate.

QuantifiedSpecialEventOccurrence associations:

- (relating) The SpecialEvent that can occur during the related ProductUsagePhase (inherited from class SpecialEventOccurrence)
- (related) The ProductUsagePhase during which the relating SpecialEvent can occur (inherited from class SpecialEventOccurrence).

QuantifiedSpecialEventOccurrence special recommendations:

- The specialEventOccurrenceRate attribute can have multiple occurrence rate values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of specialEventOccurrenceRate (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).

4.12.3.9 ProductUsagePhase

The ProductUsagePhase class defines usage phases of interest for the Product in scope of the LSA program.

ProductUsagePhase attributes:

- productUsagePhase

ProductUsagePhase associations:

- An optional association with one or many SpecialEvents that can occur during the specified product usage phase (via the SpecialEventOccurrence <>relationship>> class)

4.12.4 UoF Special Events and Damage - Additions to referenced class and interface definitions

4.12.4.1 HardwarePartAsDesigned

The HardwarePartAsDesigned class defined in UoF Part Definition must also implement the following additional <>interface>>:

- DamageAnalysisItem

4.12.4.2 HardwareElementRevision

The HardwareElementRevision class defined in UoF Breakdown Element Realization must also implement the following additional <>interface>>:

- DamageAnalysisItem

4.13 UoF LSA Candidate Task Requirement

4.13.1 Overall description

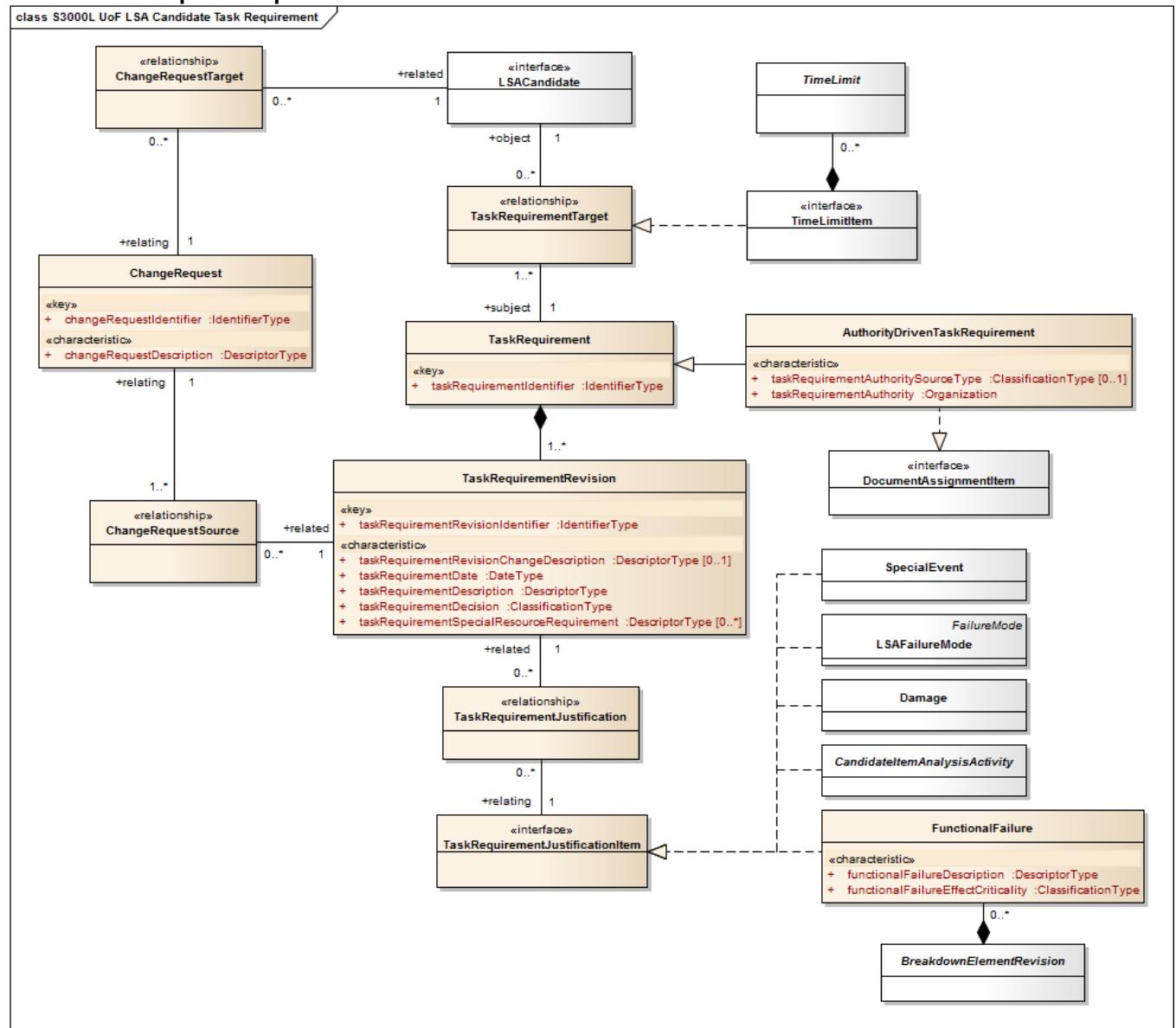
The LSA Candidate Task Requirement UoF, [Fig 30](#), supports early recordings of task requirements, which will be refined during eg, the detailed maintenance task analysis, software or data loading analysis.

Objectives for documenting task requirements are eg, to:

- Produce input to the LSA review, and together with the customer define maintenance concepts for the respective LSA Candidate before any detailed task analysis is being carried out
- Identify product design changes that will result in eg, increased maintainability or testability

A task requirement can also be associated with the source for the requirement, ie, the driver for the task. The requirement for a task can be derived from any type of LSA activity. Refer to [Para 4.10](#).

4.13.2 Graphical representation



ICN-B6865-S3000L0226-003-00

Fig 30 S3000L UoF LSA Candidate Task Requirement - class model

4.13.3 UoF LSA Candidate Task Requirement - New class and interface definitions

4.13.3.1 TaskRequirement

The TaskRequirement class supports the specification of task requirements prior to any detailed task analysis. There is also a specialization of TaskRequirement, namely AuthorityDrivenTaskRequirement.

TaskRequirement attributes:

- taskRequirementIdentifier

TaskRequirement associations:

- An association with its TaskRequirementRevisions (design iterations)
- An association with one or many LSACandidates for which the task is required (via the TaskRequirementTarget <<relationship>> class)

4.13.3.2 AuthorityDrivenTaskRequirement

The AuthorityDrivenTaskRequirement class is a specialization of class TaskRequirement, and will be used to record task requirements which are derived from regulations and/or other authoritative sources.

AuthorityDrivenTaskRequirement attributes:

- taskRequirementIdentifier (inherited from class TaskRequirement)
- taskRequirementAuthoritySourceType (zero or one)
- taskRequirementAuthority

AuthorityDrivenTaskRequirement interfaces:

- AuthorityDrivenTaskRequirement implements the DocumentAssignmentItem <<interface>>. Refer to [Para 4.20](#).

AuthorityDrivenTaskRequirement associations:

- An association with its TaskRequirementRevisions (inherited from class TaskRequirement)
- An association with one or many LSACandidates for which the task is required (inherited from class TaskRequirement)

AuthorityDrivenTaskRequirement special recommendations:

- Each instance of AuthorityDrivenTaskRequirement must have at least one document reference from which the task requirement is derived, using the DocumentAssignment capability defined in UoF Document (via the DocumentAssignmentItem <<interface>>). Refer to [Para 4.20](#).

4.13.3.3 TaskRequirementRevision

The TaskRequirementRevision class defines a revision (design iteration) of a TaskRequirement.

TaskRequirementRevision attributes:

- taskRequirementRevisionIdentifier
- taskRequirementRevisionChangeDescription (zero or one)
- taskRequirementDate
- taskRequirementDescription
- taskRequirementDecision
- taskRequirementSpecialResourceRequirement (zero, one or many)

TaskRequirementRevision associations:

- An association with the TaskRequirement of which the TaskRequirementRevision is a revision
- An optional association with one or many TaskRequirementJustificationItems, from which the TaskRequirement is derived (via the TaskRequirementJustification <<relationship>> class)
- An optional association with one or many product design ChangeRequests which are required in order to be able to carry out the task in an effective way (via instances of the ChangeRequestSource <<relationship>> class)

4.13.3.4 TaskRequirementTarget

The TaskRequirementTarget <<relationship>> class defines an association between an instance of TaskRequirement and an instance of LSACandidate for which the task requirement is identified.

TaskRequirementTarget interfaces:

- TaskRequirementTarget implements the TimeLimitItem <<interface>>. Refer to [Para 4.16](#).

TaskRequirementTarget associations:

- (subject) The TaskRequirement which is defined for the related LSACandidate
- (object) The LSACandidate for which the TaskRequirement is identified
- An optional association with one or many TimeLimits that defines the limit which must not be exceeded in between two occurrences of the defined TaskRequirement (via the TimeLimitItem <>interface>>. Refer to [Para 4.16](#).

Note

There is one instance of TaskRequirementTarget per combination of TaskRequirement and LSACandidate.

4.13.3.5 TaskRequirementJustificationItem

The TaskRequirementJustificationItem <>interface>> is implemented by classes that can justify a task requirement.

Classes that implement the TaskRequirementJustificationItem <>interface>> are:

- FunctionalFailure
- LSAFailureMode (UoF LSA FMEA). Refer to [Para 4.11](#).
- Damage (UoF Special Event And Damage). Refer to [Para 4.12](#).
- SpecialEvent (UoF Special Event And Damage) . Refer to [Para 4.12](#).
- CandidateItemAnalysisActivity (UoF LSA Candidate Analysis Activity). Refer to [Para 4.10](#).

Classes that implement the TaskRequirementJustificationItem <>interface>> must also implement the following associations:

- An optional association with one or many instances of TaskRequirementRevision (via the TaskRequirementJustification <>relationship>> class)

4.13.3.6 TaskRequirementJustification

The TaskRequirementJustification <>relationship>> class defines associations between instances of TaskRequirement and items that justify the task requirement, ie, instances of classes that implement the TaskRequirementJustificationItem <>interface>>.

TaskRequirementJustification associations:

- (relating) The analysis result that justifies the TaskRequirement (via the TaskRequirementJustificationItem <>interface>>)
- (related) The TaskRequirementRevision that documents the identified task requirement

Note

There is one instance of TaskRequirementJustification per combination of TaskRequirementRevision and item that justifies the task requirement.

4.13.3.7 ChangeRequest

The ChangeRequest class supports the recording of needed changes to the product design.

ChangeRequest attributes:

- changeRequestIdentifier
- changeRequestDescription

ChangeRequest associations:

- An association with one or many TaskRequirementRevisions for which the need of a product design change has been identified (via the ChangeRequestSource <>relationship>> class)
- An optional association with one or many LSACandidates which are affected by the defined change request (via the ChangeRequestTarget <>relationship>> class)

4.13.3.8 ChangeRequestSource

The ChangeRequestSource <> relationship class defines an association between a ChangeRequest and the TaskRequirementRevision for which the change request has been identified.

ChangeRequestSource associations:

- (relating) The ChangeRequest which is defined for the related TaskRequirementRevision
- (related) The TaskRequirementRevision for which the required product design change has been identified

Note

There is one instance of ChangeRequestSource per combination of ChangeRequest and TaskRequirement.

4.13.3.9 ChangeRequestTarget

The ChangeRequestTarget <> relationship class defines an association between a ChangeRequest and the LSACandidate which is affected by the defined change.

ChangeRequestTarget associations:

- (relating) The ChangeRequest which is defined for the related LSACandidate
- (related) The LSACandidate which is affected by the defined change request

Note

There is one instance of ChangeRequestTarget per combination of ChangeRequest and LSACandidate.

4.13.3.10 FunctionalFailure

The FunctionalFailure class documents functional failures identified during Preventive Maintenance Analysis (PMA).

Note

Functional failures can also be identified based on the outcome of Scheduled Maintenance Analysis (SMA) or Reliability Centered Maintenance (RCM).

FunctionalFailure attributes:

- functionalFailureDescription
- functionalFailureEffectCriticality

FunctionalFailure interfaces:

- FunctionalFailure implements the TaskRequirementJustificationItem <interface>

FunctionalFailure associations:

- An association with the BreakdownElementRevision for which the functional failure has been identified

Note

There is one instance of TaskRequirementTarget per combination of TaskRequirement and LSACandidate.

4.13.4 UoF LSA Candidate Task Requirement - Additions to referenced class and interface definitions

4.13.4.1 LSACandidate

Classes that implement the LSACandidate <> interface defined in UoF LSA Candidate, must also implement the following additional associations:

- An optional association with one or many TaskRequirements (via the TaskRequirementTarget <> relationship class)
- An optional association with one or many ChangeRequests (via the ChangeRequestTarget <> relationship class)

4.13.4.2 CandidateItemAnalysisActivity

The CandidateItemAnalysisActivity class defined in UoF LSA Candidate Analysis Activity must also implement the following additional <> interface>:

- TaskRequirementJustificationItem

4.13.4.3 LSAFailureMode

The LSAFailureMode class defined in UoF LSA FMEA must also implement the following additional <> interface>:

- TaskRequirementJustificationItem

4.13.4.4 SpecialEvent

The SpecialEvent class defined in UoF Special Event And Damage must also implement the following additional <> interface>:

- TaskRequirementJustificationItem

4.13.4.5 Damage

The Damage class defined in UoF Special Event And Damage must also implement the following additional <> interface>:

- TaskRequirementJustificationItem

4.14 UoF Task

4.14.1 Overall description

The Task UoF, [Fig 31](#), supports detailed definitions of tasks, both maintenance tasks, supporting tasks and operational tasks.

The task procedure is described using a set of subtasks. A subtask can either be defined and described within the task under consideration, or be a reference to another task.

Note

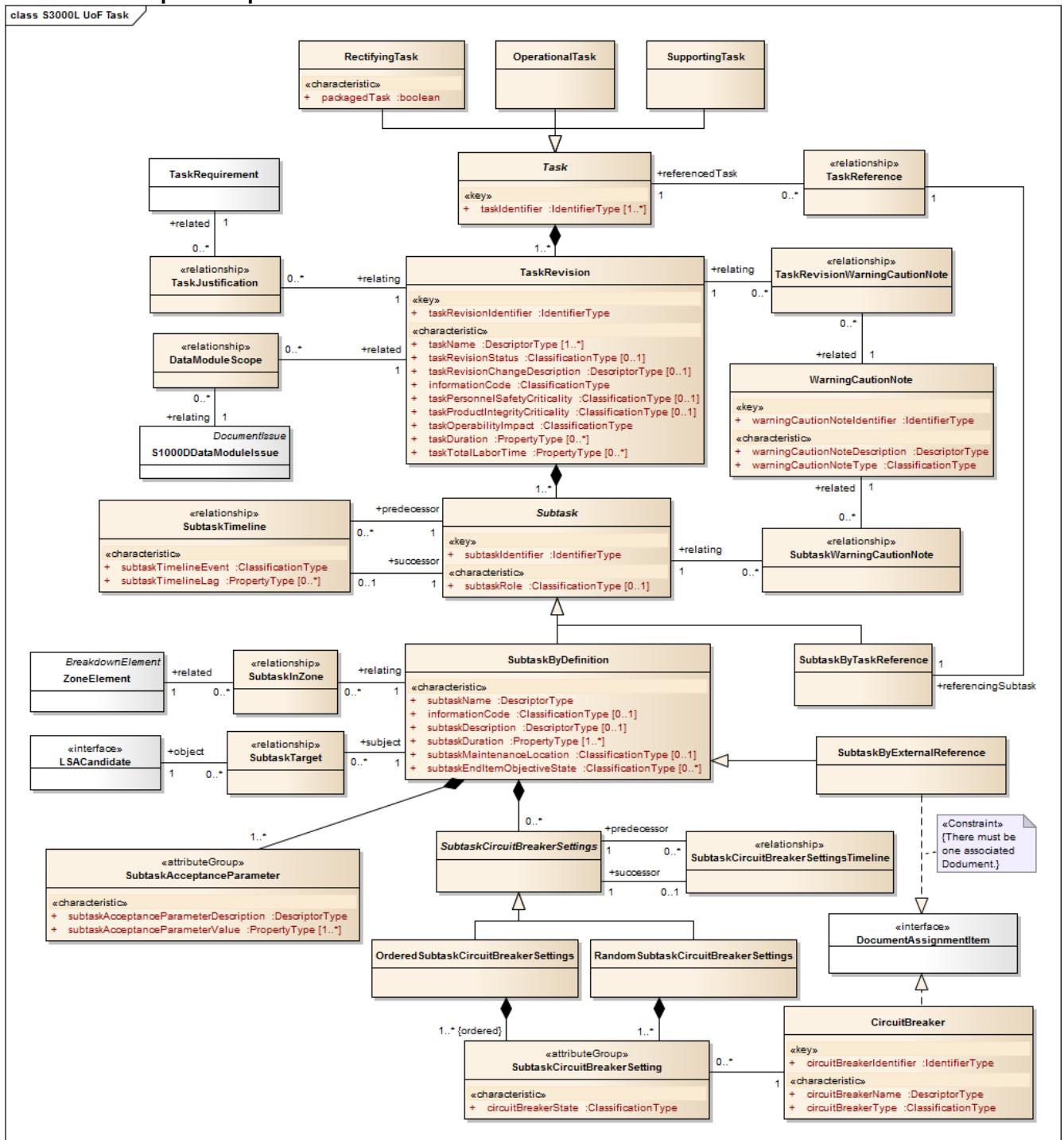
A subtask that is defined and described within a task cannot be referenced by any other task.

There is also the possibility to define a time line (schedule) for subtasks within a task. This can be used for the purpose of optimizing resource requirements and/or be the basis for creating job cards, eg, in a fleet management system.

Note

The Task UoF has been designed to support an integration between S3000L and the S1000D procedure and schedule schemes, respectively.

4.14.2 Graphical representation



ICN-B6865-S3000L0227-003-00

Fig 31 S3000L UoF Task - class model

4.14.3 UoF Task - New class and interface definitions

4.14.3.1 Task

The Task class supports the detailed specification of a task.

Applicable to: All

S3000L-A-19-00-0000-00A-040A-A

Chap 19

Note

Task is an abstract class, ie, an instantiation of Task must be either, a RectifyingTask, an OperationalTask or a SupportingTask. Refer to [Chap 12](#) for definitions and usages of the respective task type.

Task attributes:

- taskIdentifier (one or many)

Task associations:

- An association with its TaskRevisions (design iterations)
- An optional association with zero, one or many instances of Subtask that refers to the Task under consideration as a Subtask (via the TaskReference <>relationship>> class)

4.14.3.2 RectifyingTask

The RectifyingTask class is a specialization of class Task.

Note

Each maintenance activity is driven by an event. This event can be a failure, damage, special event or a time limit (interval). All these events require a maintenance action that resolves the event. Each task that is able to resolve an event must be defined as a rectifying task.

RectifyingTask attributes:

- taskIdentifier (inherited from class Task)
- packagedTask

RectifyingTask associations:

- An association with its TaskRevisions (inherited from class Task)
- An optional association with zero, one or many instances of Subtask that refers to the RectifyingTask under consideration as a Subtask (inherited from class Task)

4.14.3.3 SupportingTask

The SupportingTask class is a specialization of class Task.

Note

A supporting task does not "solve" an event, but may be used as a subtask within one or many rectifying tasks. An example of a supporting task is open hatch, or jack a car.

SupportingTask attributes:

- taskIdentifier (inherited from class Task)

SupportingTask associations:

- An association with its TaskRevisions (inherited from class Task)
- An optional association with zero, one or many instances of Subtask that refers to the SupportingTask under consideration as a Subtask (inherited from class Task)

4.14.3.4 OperationalTask

The OperationalTask class is a specialization of class Task.

Note

Operational tasks are tasks that are required for operational purposes, eg, fueling. An operational task may also be used as a subtask within one or many rectifying tasks.

OperationalTask attributes:

- taskIdentifier (inherited from class Task)

OperationalTask associations:

- An association with its TaskRevisions (inherited from class Task)
- An optional association with zero, one or many instances of Subtask that refers to the OperationalTask under consideration as a Subtask (inherited from class Task)

4.14.3.5 TaskRevision

The TaskRevision class defines an explicit revision (design iteration) of a Task.

TaskRevision attributes:

- taskRevisionIdentifier
- taskName (one or many)
- taskRevisionStatus (zero or one)
- taskRevisionChangeDescription (zero or one)
- informationCode
- taskPersonnelSafetyCriticality (zero or one)
- taskProductIntegrityCriticality (zero or one)
- taskOperabilityImpact
- taskDuration (zero, one or many)
- taskTotalLabourTime (zero, one or many)

TaskRevision associations:

- An association with the Task of which the TaskRevision is a revision
- An optional association with zero, one or many instances of WarningCautionNote that applies to the entire task (via the TaskRevisionWarningCautionNote <>relationship>> class)
- An optional association with zero, one or many S1000D data module issues, which contains the detailed procedures for the performance of the task (via the DataModuleScope <>relationship>> class)
- An optional association with zero, one or many instances of TaskRequirement which forms the justification for the task revision (via the TaskJustification <>relationship>> class)
- An association with one or many Subtasks which contains the details for the task revision

Note

A detailed procedure for the performance of the task can be documented in zero, one or many S1000D data modules. The Task UoF supports cross references between task revisions being defined in accordance with S3000L and data module issues being produced in accordance with the S1000D procedure and schedule schemes, respectively.

Note

A TaskRevision in S3000L can be split into many S1000D data modules. One S1000D data module can cover many S3000L tasks. However, projects are strongly recommended to aim for a one to one correspondence.

Note

Where the concept of master data modules is used in the S1000D Common Source DataBase (CSDB), it is recommended to just keep the cross references between the S3000L tasks and the master data modules in the CSDB.

TaskRevision special recommendations:

- The value for the taskDuration and taskTotalLabourTime attributes can use either of the following PropertyType representations:
 - SingleValuePropertyType
 - ValueWithTolerancesPropertyType

- ValueRangePropertyType

- The taskDuration and taskTotalLabourTime attributes can have multiple values depending on, for example maintenance level. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of taskDuration and taskTotalLabourTime, respectively (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).

4.14.3.6 TaskJustification

The TaskJustification <> relationship class defines associations between instances of TaskRevision and the TaskRequirements that justifies the task.

TaskJustification associations:

- (relating) The TaskRevision that is justified by the related TaskRequirement
- (related) The TaskRequirement that justifies the relating TaskRevision

Note

There is one instance of TaskJustification per combination of TaskRevision and TaskRequirement.

4.14.3.7 DataModuleScope

The DataModuleScope <> relationship class defines associations between instances of S1000DDataModuleIssue and the TaskRevisions which are documented in the data module (completely or partially).

DataModuleScope associations:

- (relating) The S1000DDataModuleIssue that completely or partially documents the related TaskRevision
- (related) The TaskRevision that is documented in the relating S1000DDataModuleIssue

Note

There is one instance of DataModuleScope per combination of S1000DDataModuleIssue and TaskRevision.

4.14.3.8 Subtask

The Subtask class defines steps to be performed within a TaskRevision. Subtasks are used to provide detailed information about the Task.

Note

Subtask is an abstract class, ie, an instantiation of Subtask needs to be either, a SubtaskByDefinition, a SubtaskByTaskReference, or a SubtaskByExternalReference (subclass of class SubtaskByDefinition).

Note

Subtasks within a TaskRevision can be time lined (scheduled) using the SubtaskTimeline <> relationship class.

Subtask attributes:

- subtaskId
- subtaskRole (zero or one)

Subtask associations:

- An association with the TaskRevision to which the Subtask belongs
- An optional association with zero, one or many instances of WarningCautionNote that applies to the Subtask under consideration (via instances of the SubtaskWarningCautionNote <> relationship class)

- An optional association with zero or one related Subtask, on which the starting point for the Subtask under consideration is dependent (used for time lining of the overall Task eg, preceding subtask in a GANTT-schema)
- An optional association with zero, one or many Subtasks, whose starting points are dependent upon the Subtask under consideration (used for time lining of the overall Task, eg, succeeding subtasks in a GANTT-schema)

Subtask special recommendations:

- Where a Subtask is dependent upon eg, maintenance level, it must be distinguished by the assignment of an ApplicabilityStatement to the Subtask instance. Refer to [Para 4.22](#).

Note

Alternative Subtasks within a Task must be distinguished by the assignment of ApplicabilityStatements.

4.14.3.9 SubtaskTimeline

The SubtaskTimeline <>relationship> class can define timeline associations between two instances of Subtask. The timeline association enables the definition of time dependencies between two Subtasks within the same TaskRevision.

The SubtaskTimeline class defines the event to which relating (successor) Subtask refers, eg, start or end of the related (predecessor) Subtask. It also defines a possible lag, ie, duration from the time the related event occurs and the time when the relating Subtask can be initiated (started).

Note

Subtasks that don't relate to any other Subtask, via the SubtaskTimeline association, must be regarded as being performed in sequence according to their subtaskIdentifiers (low to high).

Note

This class supports the creation of a GANTT-schema for a Task.

SubtaskTimeline attributes:

- subtaskTimelineEvent
- subtaskTimelineLag (zero, one or many)

SubtaskTimeline associations:

- (predecessor) The Subtask on which the successor Subtask instance is dependent
- (successor) The Subtask which is dependent upon its predecessor Subtask instance

SubtaskTimeline special recommendations:

- The subtaskTimelineLag attribute can have multiple values depending on, for example environmental conditions. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of subtaskTimelineLag (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).

4.14.3.10 WarningCautionNote

The WarningCautionNote class defines advices concerning safety, legal and health aspects.

WarningCautionNote attributes:

- warningCautionNotIdentifier
- warningCautionNoteDescription
- warningCautionNoteType

WarningCautionNote associations:

- An optional association with zero, one or many TaskRevisions to which the WarningCautionNote applies (via the TaskRevisionWarningCautionNote <> relationship class)
- An optional association with zero, one or many Subtasks to which the WarningCautionNote applies (via the SubtaskWarningCautionNote <> relationship class)

4.14.3.11 TaskRevisionWarningCautionNote

The TaskRevisionWarningCautionNote <> relationship class defines associations between instances of TaskRevision and WarningCautionNotes that applies to the respective task revision.

TaskRevisionWarningCautionNote associations:

- (relating) The TaskRevision that needs the advice concerning safety, legal and health aspects
- (related) The WarningCautionNote that contains the advice concerning safety, legal and health aspects

Note

There is one instance of TaskRevisionWarningCautionNote per relevant combination of TaskRevision and WarningCautionNote.

TaskRevisionWarningCautionNote special recommendations:

- Where a TaskRevisionWarningCautionNote is dependent upon eg, maintenance level or external conditions, it must be distinguished by the assignment of an ApplicabilityStatement to the TaskRevisionWarningCautionNote instance. Refer to [Para 4.22](#).

4.14.3.12 SubtaskWarningCautionNote

The SubtaskWarningCautionNote <> relationship class defines associations between instances of Subtask and WarningCautionNotes that applies to the respective subtask.

SubtaskWarningCautionNote associations:

- (relating) The Subtask that needs the advice concerning safety, legal and health aspects
- (related) The WarningCautionNote that contains the advice concerning safety, legal and health aspects

Note

There is one instance of SubtaskWarningCautionNote per relevant combination of Subtask and WarningCautionNote.

SubtaskWarningCautionNote special recommendations:

- Where a SubtaskWarningCautionNote is dependent upon eg, maintenance level or external conditions, it must be distinguished by the assignment of an ApplicabilityStatement to the SubtaskWarningCautionNote instance. Refer to [Para 4.22](#).

4.14.3.13 SubtaskByTaskReference

The SubtaskByTaskReference class is a specialization of Subtask. SubtaskByTaskReference will be used when the Subtask is defined as a Task in its own right.

Note

SubtaskByTaskReference is the only mechanism in S3000L that supports reuse of task steps and/or task procedures in between Tasks.

SubtaskByTaskReference attributes:

- subtaskIdIdentifier (inherited from class Subtask)
- subtaskRole (inherited from class Subtask)

SubtaskByTaskReference associations:

- An association with the TaskRevision to which the SubtaskByTaskReference belongs (inherited from class Subtask)
- An optional association with zero, one or many instances of WarningCautionNote that applies to the SubtaskByTaskReference under consideration (inherited from class Subtask)
- An optional association with zero or one related Subtask, on which the starting point for the SubtaskByTaskReference under consideration is dependent (inherited from class Subtask)
- An optional association with zero, one or many Subtasks, whose starting points are dependent upon the SubtaskByTaskReference under consideration (inherited from class Subtask)
- An association with the Task that is referenced as a Subtask (via the TaskReference <>relationship> class)

SubtaskByTaskReference special recommendations:

- See special recommendations for Subtask, [Para 4.14.3.8](#)

4.14.3.14 TaskReference

The TaskReference <>relationship> class defines associations between instances of SubtaskByTaskReference and the Task that is referenced as a subtask.

TaskReference associations:

- (referencingSubtask) The SubtaskByTaskReference that is referring to a Task as a subtask
- (referencedTask) The Task that is incorporated as a subtask in another task

Note

There is one instance of TaskReference per combination of SubtaskByTaskReference and Task.

4.14.3.15 SubtaskByDefinition

The SubtaskByDefinition class is a specialization of Subtask. A SubtaskByDefinition provides a detailed characterization of the subtask to be included in the overall TaskRevision. A SubtaskByDefinition cannot be referenced from any other Task.

Note

SubtaskByDefinition can be specialized into a SubtaskByExternalReference.

SubtaskByDefinition attributes:

- subtaskIdentifier (inherited from class Subtask)
- subtaskRole (inherited from class Subtask)
- subtaskName
- informationCode (zero or one)
- subtaskDescription (zero or one)
- subtaskDuration (one or many)
- subtaskMaintenanceLocation (zero or one)
- subtaskEndItemObjectiveState (zero one or many)

SubtaskByDefinition associations:

- An association with the TaskRevision to which the SubtaskByDefinition belongs (inherited from class Subtask)
- An optional association with zero, one or many instances of WarningCautionNote that applies to the SubtaskByDefinition under consideration (inherited from class Subtask)
- An optional association with zero or one related Subtask, on which the starting point for the SubtaskByDefinition under consideration is dependent (inherited from class Subtask)

- An optional association with zero, one or many Subtasks, whose starting points are dependent upon the SubtaskByDefinition under consideration (inherited from class Subtask)
- An optional association with zero, one or many zonal locations (ZoneElement) in which the subtask will be performed (via the SubtaskInZone <>relationship>> class)
- An optional association with zero, one or many instances of SubtaskAcceptanceParameter <>attributeGroup>>, defining criteria that need to be fulfilled before the subtask can be closed
- An optional association with zero, one or many instances of LSACandidate on which the subtask will be performed (via the SubtaskTarget <>relationship>> class)
- An optional association with zero, one or many instances of SubtaskCircuitBreakerSettings which defines the circuit breaker states that must be obtained before the subtask can be closed

SubtaskByDefinition special recommendations:

- The value for the subtaskDuration attribute can use either of the following PropertyType representations:
 - SingleValue.PropertyType
 - ValueWithTolerances.PropertyType
 - ValueRange.PropertyType
- The subtaskDuration attribute can have multiple values depending on, for example. maintenance level. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of subtaskDuration (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).
- See special recommendations for Subtask, [Para 4.14.3.8](#)

4.14.3.16 SubtaskInZone

The SubtaskInZone <>relationship>> class defines associations between instances of SubtaskByDefinition and the zonal location (ZoneElement) in which the respective subtask will be performed.

Note

There is one instance of SubtaskInZone per relevant combination of SubtaskByDefinition and zonal location (ZoneElement).

SubtaskInZone associations:

- (relating) The SubtaskByDefinition that will be performed in a specific zonal location
- (related) The ZoneElement that represents the zonal location in which the subtask will be performed

4.14.3.17 SubtaskAcceptanceParameter

The SubtaskAcceptanceParameter <>attributeGroup>> defines criteria that needs to be fulfilled before the subtask can be closed.

Note

An instance of class SubtaskAcceptanceParameter has no meaning by itself, but only in the context of a specific instance of SubtaskByDefinition.

SubtaskAcceptanceParameter attributes:

- subtaskAcceptanceParameterDescription
- subtaskAcceptanceParameterValue (one or many)

SubtaskAcceptanceParameter associations:

- An association with the specific instance of SubtaskByDefinition for which the SubtaskAcceptanceParameter has been defined

SubtaskAcceptanceParameter special recommendations:

- The value for the subtaskAcceptanceParameterValue attribute can use either of the following PropertyType representations:
 - SingleValue.PropertyType
 - ValueWithTolerances.PropertyType
 - ValueRange.PropertyType
- The subtaskAcceptanceParameterValue can have multiple values depending on, for example operational environment. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of subtaskAcceptanceParameterValue (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).

4.14.3.18 SubtaskTarget

The SubtaskTarget <> relationship class defines an association between an instance of SubtaskByDefinition and the object (instance of a class that implements the LSACandidate <> interface) on which a SubtaskByDefinition will be performed.

Note

There is one instance of SubtaskTarget per relevant combination of SubtaskByDefinition and LSACandidate.

SubtaskTarget associations:

- (subject) The SubtaskByDefinition that will be performed on the related object (instance of a class that implements the LSACandidate <> interface)
- (object) The LSACandidate on which the subtask will be performed

4.14.3.19 SubtaskCircuitBreakerSettings

The SubtaskCircuitBreakerSettings class identifies a set of circuit breakers that must be set in specific states as part of the SubtaskByDefinition.

Note

The SubtaskCircuitBreakerSettings class is an abstract class, ie, an instantiation of SubtaskCircuitBreakerSettings must be either:

- OrderedSubtaskCircuitBreakerSettings
- RandomSubtaskCircuitBreakerSettings

Note

An instance of a subclass of SubtaskCircuitBreakerSettings has no meaning by itself, but only in the context of a specific instance of SubtaskByDefinition.

Note

A mixture of ordered and unordered SubtaskCircuitBreakerSettings can be defined by having multiple instances of SubtaskCircuitBreakerSettings which then are organized into a timeline using instances of the SubtaskCircuitBreakerSettingsTimeline class.

Note

Multiple instances of SubtaskCircuitBreakerSettings within the same SubtaskByDefinition without explicit relationships between each other (via the SubtaskCircuitBreakerSettingsTimeline <> relationship class) must be regarded as being performed in parallel.

SubtaskCircuitBreakerSettings association:

- An association with the SubtaskByDefinition to which the SubtaskCircuitBreakerSettings belongs

- An optional association with zero or one related SubtaskCircuitBreakerSettings, of which the SubtaskCircuitBreakerSettings under consideration is dependent (used for time lining of SubtaskCircuitBreakerSettings via instances of the SubtaskCircuitBreakerSettingsTimeline <>relationship>> class)
- An optional association with zero, one or many SubtaskCircuitBreakerSettings, whose starting is dependent upon the finalization of the SubtaskCircuitBreakerSettings under consideration (used for time lining of SubtaskCircuitBreakerSettings via instances of the SubtaskCircuitBreakerSettingsTimeline <>relationship>> class)

4.14.3.20 SubtaskCircuitBreakerSettingsTimeline

The SubtaskCircuitBreakerSettingsTimeline <>relationship>> class defines a time dependency between two instances of SubtaskCircuitBreakerSettings within the same SubtaskByDefinition.

SubtaskCircuitBreakerSettingsTimeline associations:

- (predecessor) The instance of SubtaskCircuitBreakerSettings on which the successor SubtaskCircuitBreakerSettings instance is dependent
- (successor) The SubtaskCircuitBreakerSettings instance which is dependent upon its predecessor SubtaskCircuitBreakerSettings instance

4.14.3.21 OrderedSubtaskCircuitBreakerSettings

The OrderedSubtaskCircuitBreakerSettings class is a specialization of the SubtaskCircuitBreakerSettings class. OrderedSubtaskCircuitBreakerSettings must be used when the order in which individual circuit breakers are set are of importance.

OrderedSubtaskCircuitBreakerSettings associations:

- An association with the SubtaskByDefinition to which the OrderedSubtaskCircuitBreakerSettings belongs (inherited from SubtaskCircuitBreakerSettings)
- An optional association with zero or one related SubtaskCircuitBreakerSettings, of which the OrderedSubtaskCircuitBreakerSettings under consideration is dependent (inherited from class SubtaskCircuitBreakerSettings)
- An optional association with zero, one or many SubtaskCircuitBreakerSettings, whose starting is dependent upon the finalization of the OrderedSubtaskCircuitBreakerSettings under consideration (inherited from class SubtaskCircuitBreakerSettings)
- An ordered association with one or many instances of the SubtaskCircuitBreakerSetting <>attributeGroup>>

Note

The ordered association with one or many instances of SubtaskCircuitBreakerSetting defines the order and state in which individual circuit breakers must be set.

4.14.3.22 RandomSubtaskCircuitBreakerSettings

The RandomSubtaskCircuitBreakerSettings class is a specialization of the SubtaskCircuitBreakerSettings class. RandomSubtaskCircuitBreakerSettings must be used when the order in which individual circuit breakers are set are of no importance.

RandomSubtaskCircuitBreakerSettings associations:

- An association with the SubtaskByDefinition to which the RandomSubtaskCircuitBreakerSettings belongs (inherited from SubtaskCircuitBreakerSettings)
- An optional association with zero or one related SubtaskCircuitBreakerSettings, of which the RandomSubtaskCircuitBreakerSettings under consideration is dependent (inherited from class SubtaskCircuitBreakerSettings)

- An optional association with zero, one or many SubtaskCircuitBreakerSettings, whose starting is dependent upon the finalization of the RandomSubtaskCircuitBreakerSettings under consideration (inherited from class SubtaskCircuitBreakerSettings)
- An unordered association with one or many instances of the SubtaskCircuitBreakerSetting <>attributeGroup>>

4.14.3.23 SubtaskCircuitBreakerSetting

The SubtaskCircuitBreakerSetting <>attributeGroup>> defines the state that the individual circuit breaker must be set in.

Note

An instance of class SubtaskCircuitBreakerSetting has no meaning by itself, but only in the context of a specific instance of either OrderedSubtaskCircuitBreakerSettings or RandomSubtaskCircuitBreakerSettings.

SubtaskCircuitBreakerSetting attributes:

- circuitBreakerState

SubtaskCircuitBreakerSetting associations:

- An association with the specific instance of either OrderedSubtaskCircuitBreakerSettings or RandomSubtaskCircuitBreakerSettings for which the circuitBreakerState must be obtained
- An association with the individual CircuitBreaker that must be set in the defined state

4.14.3.24 CircuitBreaker

The CircuitBreaker class identifies individual circuit breakers that are installed on the Product.

CircuitBreaker attributes:

- circuitBreakerIdentifier
- circuitBreakerName
- circuitBreakerType

The CircuitBreaker class must also implement the following <>interface>>:

- DocumentAssignmentItem. Refer to [Para 4.20](#).

CircuitBreaker associations:

- An optional association with zero, one or many instances of SubtaskCircuitBreakerSetting, ie, states that the circuit breaker under consideration must be in before a specific instance of SubtaskByDefinition can be closed

CircuitBreaker special recommendations:

- Circuit breaker location can be defined using the assignment of a Document, where the documentAssignmentRole is set to “Source”, and the documentPortion attribute describes the location

4.14.3.25 SubtaskByExternalReference

The SubtaskByExternalReference class is a specialization of SubtaskByDefinition.

SubtaskByExternalReference must be used whenever the complete description of the Subtask is described in an external source, ie, outside the scope of the LSA program under consideration.

SubtaskByExternalReference attributes:

- subtaskIdentifier (inherited from class Subtask)
- subtaskRole (inherited from class Subtask)
- subtaskName (inherited from class SubtaskByDefinition)

- informationCode (inherited from class SubtaskByDefinition)
- subtaskDescription (inherited from class SubtaskByDefinition)
- subtaskDuration (inherited from class SubtaskByDefinition)
- subtaskMaintenanceLocation (inherited from class SubtaskByDefinition)
- subtaskEndItemObjectiveState (inherited from class SubtaskByDefinition).

The SubtaskByExternalReference class must also implement the following <>interface>>:

- DocumentAssignmentItem. Refer to [Para 4.20](#).

SubtaskByExternalReference associations:

- An association with the TaskRevision to which the SubtaskByExternalReference belongs (inherited from class Subtask)
- An optional association with zero, one or many instances of WarningCautionNote that applies to the SubtaskByExternalReference under consideration (inherited from class Subtask)
- An optional association with zero or one related Subtask, on which the starting point for the SubtaskByExternalReference under consideration is dependent (inherited from class Subtask)
- An optional association with zero, one or many Subtasks, whose starting points are dependent upon the SubtaskByExternalReference under consideration (inherited from class Subtask)
- An optional association with zero, one or many zonal locations (ZoneElement) in which the subtask will be performed (inherited from class SubtaskByDefinition)
- An optional association with zero, one or many instances of SubtaskAcceptanceParameter, defining criteria that need to be fulfilled before the subtask can be closed (inherited from class SubtaskByDefinition)
- An optional association with zero, one or many instances of LSACandidate on which the subtask will be performed (inherited from class SubtaskByDefinition)
- An optional association with zero, one or many instances of SubtaskCircuitBreakerSettings which defines the circuit breaker states that must be obtained before the subtask can be closed (inherited from class SubtaskByDefinition)

SubtaskByExternalReference special recommendations:

- An instance of SubtaskByExternalReference must be associated with at least one instance of ExternalDocument or S1000DPublicationModule (via the DocumentAssignmentItem <>interface>>)
- See special recommendations for SubtaskByDefinition, refer to [Para 4.14.3.15](#)

4.14.4 UoF Task - Additions to referenced class and interface definitions

4.14.4.1 TaskRequirement

The TaskRequirement class defined in UoF LSA Candidate Task Requirement must also implement the following additional association:

- An optional association with zero, one or many Tasks which are justified by the TaskRequirement (via instances of the TaskJustification <>relationship>> class)

4.14.4.2 S1000DDDataModuleIssue

The S1000DDDataModuleIssue class defined in UoF Document must also implement the following additional association:

- An optional association with zero, one or many instances of TaskRevision, which are described in the identified issue of the S1000D data module (via instances of the DataModuleScope <>relationship>> class)

4.14.4.3 LSACandidate

Classes that implements the LSACandidate <>interface>> defined in UoF LSA Candidate, must also implement the following additional association:

- An optional association with zero, one or many instances of SubtaskByDefinition for which the LSACandidate is the target object (via instances of the SubtaskTarget <>relationship>> class)

4.14.4.4 ZoneElement

The ZoneElement class defined in UoF Breakdown Zone Element must also implement the following additional association:

- An optional association with zero, one or many instances of SubtaskByDefinition which will be performed in the identified zonal location (via instances of the SubtaskInZone <>relationship>> class)

4.15 UoF Task Resources

4.15.1 Overall description

The Task Resources UoF, [Fig 32](#), supports a detailed definition of resources needed, either per subtask, or aggregated per task.

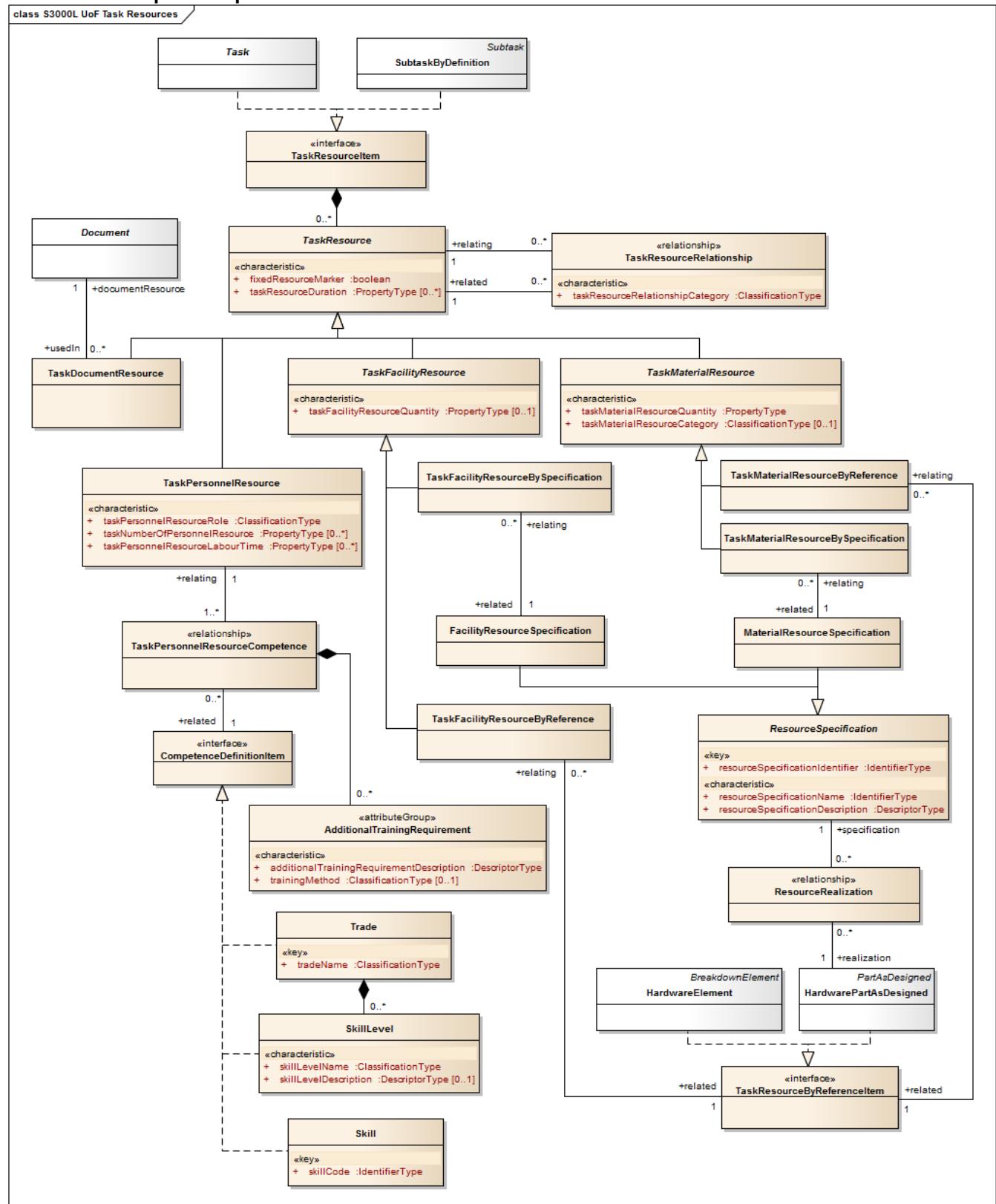
Task Resources can be either:

- Material resources, eg, spares, tools, consumables
- Facilities, eg, hangars, power
- Personnel resources, including eg, skills or additional training requirements
- Documents needed for eg, recording of test results

Material and facility resources can be defined both in terms of specifications, and in terms of part identifiers.



4.15.2 Graphical representation



ICN-B6865-S3000L0228-003-00

Fig 32 S3000L UoF Task Resources - class model

Applicable to: All

S3000L-A-19-00-0000-00A-040A-A

Chap 19

4.15.3 UoF Task Resources - New class and interface definitions

4.15.3.1 TaskResourceItem

The TaskResourceItem <> interface enables resources to be assigned to classes that implement this <> interface.

Classes that implement the TaskResourceItem <> interface are:

- Task
- SubtaskByDefinition

Classes that implement the TaskResourceItem <> must also implement the following association:

- An optional association with zero, one or many TaskResources, ie, resources needed for the performance of a specific Task or SubtaskByDefinition instance

4.15.3.2 TaskResource

The TaskResource class supports the identification of resources needed for the performance of a specific Task or SubtaskByDefinition.

Note

TaskResource is an abstract class, ie, an instantiation of TaskResource must be done using any of the following specializations:

- TaskMaterialResourceBySpecification
- TaskMaterialResourceByReference
- TaskFacilityResourceBySpecification
- TaskFacilityResourceByReference
- TaskPersonnelResource
- TaskDocumentResource

Note

Each instance of TaskResource is uniquely defined for a specific instance of Task or SubtaskByDefinition. Reuse of resource definitions is enabled through the use of ResourceSpecifications, HardwarePartAsDesigned, Skills, Trades and Documents.

Note

An instance of class TaskResource has no meaning by itself, but only in the context of a specific Task or SubtaskByDefinition.

TaskResource attributes:

- fixedResourceMarker
- taskResourceDuration (zero, one or many)

TaskResource associations:

- An association with the Task or SubtaskByDefinition instance, for which the resource is needed (via the TaskResourceItem <> interface)
- An instance of TaskResource can relate to one or many other instances of TaskResource, via the TaskResourceRelationship <> class, eg, "person A uses tool B"
- An instance of TaskResource can be related to from one or many other instances of TaskResource via the TaskResourceRelationship <> class, eg, "tool B is used by person A"

TaskResource special recommendations:

- Where a TaskResource is dependent upon eg, maintenance level or external condition, it must be distinguished by the assignment of an ApplicabilityStatement to the TaskResource instance. Refer to [Para 4.22](#).

- The taskResourceDuration attribute can have multiple values depending on, for example maintenance level or external condition. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of taskResourceDuration (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).

4.15.3.3 TaskResourceRelationship

The TaskResourceRelationship <>relationship>> class defines associations between two instances of TaskResource within the same instance of Task or SubtaskByDefinition. An example of a task resource relationship is eg, person A "uses" tool B.

TaskResourceRelationship attributes:

- taskResourceRelationshipCategory

TaskResourceRelationship associations:

- (relating) The instance of TaskResource which relates to another instance of TaskResource
- (related) The instance of TaskResource that is being related to from another instance of TaskResource

4.15.3.4 TaskMaterialResource

The TaskMaterialResource class is a specialization of TaskResource and identifies material resources needed for the performance of a specific Task or SubtaskByDefinition.

Note

TaskMaterialResource is an abstract class, ie, an instantiation of TaskMaterialResource must be done using any of its specializations:

- TaskMaterialResourceBySpecification
- TaskMaterialResourceByReference

Note

Each instance of TaskMaterialResource is uniquely defined for a specific instance of Task or SubtaskByDefinition. Reuse of material resource definitions is enabled through the use of MaterialResourceSpecifications and HardwarePartAsDesigned.

TaskMaterialResource attributes:

- fixedResourceMarker (inherited from class TaskResource)
- taskResourceDuration (inherited from class TaskResource)
- taskMaterialResourceQuantity
- taskMaterialResourceCategory (zero or one)

TaskMaterialResource associations:

- An association with the instance of Task or SubtaskByDefinition for which the associated material resource is needed (inherited from class TaskResource)
- An instance of TaskMaterialResource can relate to one or many other instances of TaskResource, via the TaskResourceRelationship <>relationship>> class, eg, "tool B is used by person A" (inherited from class TaskResource)
- An instance of TaskMaterialResource can be related to from one or many other instances of TaskResource via the TaskResourceRelationship <>relationship>> class, eg, "person A uses tool B" (inherited from class TaskResource)

TaskMaterialResource special recommendations:

- See special recommendations for Task Resource, [Para 4.15.3.2](#)

4.15.3.5 TaskMaterialResourceByReference

The TaskMaterialResourceByReference class is a specialization of TaskMaterialResource, and identifies a specific hardware item (HardwarePartAsDesigned or HardwareElement) as the material resource needed for the performance of a specific Task or SubtaskByDefinition.

Note

Each instance of TaskMaterialResourceByReference is uniquely defined for a specific instance of Task or SubtaskByDefinition.

TaskMaterialResourceByReference attributes:

- fixedResourceMarker (inherited from class TaskResource)
- taskResourceDuration (inherited from class TaskResource)
- taskMaterialResourceQuantity (inherited from class TaskMaterialResource)
- taskMaterialResourceCategory (inherited from class TaskMaterialResource)

TaskMaterialResourceByReference associations:

- An association with the instance of Task or SubtaskByDefinition for which the associated material resource is needed (inherited from class TaskResource)
- An instance of TaskMaterialResourceByReference can relate to one or many other instances of TaskResource, via the TaskResourceRelationship <> relationship class, eg, "tool B is used by person A" (inherited from class TaskResource)
- An instance of TaskMaterialResourceByReference can be related to from one or many other instances of TaskResource via the TaskResourceRelationship <> relationship class, eg, "person A uses tool B" (inherited from class TaskResource)
- An association with the HardwarePartAsDesigned or HardwareElement that plays the role of a material resource (via the TaskResourceByReferenceItem <> interface)

TaskMaterialResourceByReference special recommendations:

- See special recommendations for Task Resource, [Para 4.15.3.2](#)

4.15.3.6 TaskMaterialResourceBySpecification

The TaskMaterialResourceBySpecification class is a specialization of TaskMaterialResource, and identifies the material resource needed to perform a specific Task or SubtaskByDefinition, through a MaterialResourceSpecification.

Note

Each instance of TaskMaterialResourceBySpecification is uniquely defined for a specific instance of Task or SubtaskByDefinition. Reuse of material resource specifications is enabled through the use of MaterialResourceSpecification.

TaskMaterialResourceBySpecification attributes:

- fixedResourceMarker (inherited from class TaskResource)
- taskResourceDuration (inherited from class TaskResource)
- taskMaterialResourceQuantity (inherited from class TaskMaterialResource)
- taskMaterialResourceCategory (inherited from class TaskMaterialResource)

TaskMaterialResourceBySpecification associations:

- An association with the instance of Task or SubtaskByDefinition for which the associated material resource is needed (inherited from class TaskResource)
- An instance of TaskMaterialResourceBySpecification can relate to one or many other instances of TaskResource, via the TaskResourceRelationship <> relationship class, eg, "tool B is used by person A" (inherited from class TaskResource)
- An instance of TaskMaterialResourceBySpecification can be related to from one or many other instances of TaskResource via the TaskResourceRelationship <> relationship class, eg, "person A uses tool B" (inherited from class TaskResource)

- An association with the instance of MaterialResourceSpecification that specifies the material resource

TaskMaterialResourceBySpecification special recommendations:

- See special recommendations for Task Resource, [Para 4.15.3.2](#)

4.15.3.7 ResourceSpecification

The ResourceSpecification class specifies a resource which can be realized by one or many HardwarePartAsDesigned.

Note

Use of ResourceSpecifications allows for more generic task/subtask definitions, ie, the task/subtask does not need to be changed based on customer specific tool sets.

Note

ResourceSpecification is an abstract class, ie, an instantiation of ResourceSpecification must be either a FacilityResourceSpecification or a MaterialResourceSpecification.

ResourceSpecification attributes:

- resourceSpecificationIdentifier
- resourceSpecificationName
- resourceSpecificationDescription

ResourceSpecification associations:

- An optional association with zero, one or many HardwarePartAsDesigned that realizes the ResourceSpecification, eg, actual tools, equipment, spares, that can be used when the task will be performed (via an instance of the ResourceRealization <<relationship>> class)

4.15.3.8 ResourceRealization

The ResourceRealization <<relationship>> class defines relationships between ResourceSpecifications and the HardwarePartAsDesigned that fulfills the respective specification.

ResourceRealization associations:

- (specification) The ResourceSpecification that has a HardwarePartAsDesigned realization
- (realization) The HardwarePartAsDesigned instance that fulfills the resource specification (also known as the part that realizes the specification)

Note

There is one instance of ResourceRealization per relevant combination of ResourceSpecification and HardwarePartAsDesigned.

ResourceRealization special recommendations:

- Where a ResourceRealization is dependent upon eg, customer, maintenance level or external conditions, it must be distinguished by the assignment of an ApplicabilityStatement to the ResourceRealization instance. Refer to [Para 4.22](#).

4.15.3.9 MaterialResourceSpecification

The MaterialResourceSpecification class is a specialization of ResourceSpecification, and identifies specifications of material resources (eg, spares, consumables, tools) which can be realized by one or many HardwarePartAsDesigned.

Note

Use of MaterialResourceSpecifications allows for more generic task/subtask definitions, ie, the task/subtask does not need to be changed based on customer specific tool sets.

MaterialResourceSpecification attributes:

- resourceSpecificationIdentifier (inherited from class ResourceSpecification)
- resourceSpecificationName (inherited from class ResourceSpecification)
- resourceSpecificationDescription (inherited from class ResourceSpecification)

MaterialResourceSpecification associations:

- An optional association with zero, one or many HardwarePartAsDesigned that realizes the MaterialResourceSpecification, eg, actual tools, equipment, spares, that can be used when the task will be performed (inherited from class ResourceSpecification)
- An optional association with zero, one or many TaskMaterialResourceBySpecifications, ie, individual usages of the specified material resource in a specific Task or SubtaskByDefinition

MaterialResourceSpecification special recommendations:

- See special recommendations for ResourceSpecification, [Para 4.15.3.8](#)

4.15.3.10 TaskFacilityResource

The TaskFacilityResource class is a specialization of TaskResource and identifies facility resources needed for the performance of a specific Task or SubtaskByDefinition.

Note

TaskFacilityResource is an abstract class, ie, an instantiation of TaskFacilityResource must be done using any of its specializations:

- TaskFacilityResourceBySpecification
- TaskFacilityResourceByReference

Note

Each instance of TaskFacilityResource is uniquely defined for a specific instance of Task or SubtaskByDefinition. Reuse of facility resource definitions is enabled through the use of FacilityResourceSpecifications and HardwarePartAsDesigned.

TaskFacilityResource attributes:

- fixedResourceMarker (inherited from class TaskResource)
- taskResourceDuration (inherited from class TaskResource)
- taskFacilityResourceQuantity (zero or one)

TaskFacilityResource associations:

- An association with the instance of Task or SubtaskByDefinition for which the associated facility resource is needed (inherited from class TaskResource)
- An instance of TaskFacilityResource can relate to one or many other instances of TaskResource, via the TaskResourceRelationship <> relationship class, eg, "power utility A powers equipment B" (inherited from class TaskResource)
- An instance of TaskFacilityResource can be related to from one or many other instances of TaskResource via the TaskResourceRelationship <> relationship class, eg, "equipment B must be powered by power utility A" (inherited from class TaskResource)

TaskFacilityResource special recommendations:

- See special recommendations for TaskResource, [Para 4.15.3.2](#)

4.15.3.11 TaskFacilityResourceByReference

The TaskFacilityResourceByReference class is a specialization of TaskFacilityResource, and identifies a specific hardware item (HardwarePartAsDesigned or HardwareElement) as the facility resource needed for the performance of a specific Task or SubtaskByDefinition.

Note

Each instance of TaskFacilityResourceByReference is uniquely defined for a specific instance of Task or SubtaskByDefinition.

TaskFacilityResourceByReference attributes:

- fixedResourceMarker (inherited from class TaskResource)
- taskResourceDuration (inherited from class TaskResource)
- taskFacilityResourceQuantity (inherited from class TaskFacilityResource)

TaskFacilityResourceByReference associations:

- An association with the instance of Task or SubtaskByDefinition for which the associated facility resource is needed (inherited from class TaskResource)
- An instance of TaskFacilityResourceByReference can relate to one or many other instances of TaskResource, via the TaskResourceRelationship <> relationship class, eg, “power utility A powers equipment B” (inherited from class TaskResource)
- An instance of TaskFacilityResourceByReference can be related to from one or many other instances of TaskResource via the TaskResourceRelationship <> relationship class, eg, “equipment B must be powered by power utility A” (inherited from class TaskResource)
- An association with the HardwarePartAsDesigned or HardwareElement that plays the role of a facility resource (via the TaskResourceByReferenceItem <> interface)

TaskFacilityResourceByReference special recommendations:

- See special recommendations for TaskResource, [Para 4.15.3.2](#)

4.15.3.12 TaskFacilityResourceBySpecification

The TaskFacilityResourceBySpecification class is a specialization of TaskFacilityResource, and identifies the facility resource needed to perform a specific Task or SubtaskByDefinition through a FacilityResourceSpecification.

Note

Each instance of TaskFacilityResourceBySpecification is uniquely defined for a specific instance of Task or SubtaskByDefinition. Reuse of facility resource specifications is enabled through the use of FacilityResourceSpecification.

TaskFacilityResourceBySpecification attributes:

- fixedResourceMarker (inherited from class TaskResource)
- taskResourceDuration (inherited from class TaskResource)
- taskFacilityResourceQuantity (inherited from class TaskFacilityResource)

TaskFacilityResourceBySpecification associations:

- An association with the instance of Task or SubtaskByDefinition for which the associated facility resource is needed (inherited from class TaskResource)
- An instance of TaskFacilityResourceBySpecification can relate to one or many other instances of TaskResource, via the TaskResourceRelationship <> relationship class, eg, “power utility A powers equipment B” (inherited from class TaskResource)
- An instance of TaskFacilityResourceBySpecification can be related to from one or many other instances of TaskResource via the TaskResourceRelationship <> relationship class, eg, “equipment B must be powered by power utility A” (inherited from class TaskResource)
- An association with the instance of FacilityResourceSpecification that specifies the required facility resource

TaskFacilityResourceBySpecification special recommendations:

- See special recommendations for TaskResource, [Para 4.15.3.2](#)

4.15.3.13 FacilityResourceSpecification

The FacilityResourceSpecification class is a specialization of ResourceSpecification, and identifies specifications of facility resources (eg, power, internet, hangar, dock) which can be realized by one or many HardwarePartAsDesigned.

Note

Use of FacilityResourceSpecifications allows for more generic task/subtask definitions, ie, the task/subtask does not need to be changed based on customer specific realizations of facilities.

FacilityResourceSpecification attributes:

- resourceSpecificationIdentifier (inherited from class ResourceSpecification)
- resourceSpecificationName (inherited from class ResourceSpecification)
- resourceSpecificationDescription (inherited from class ResourceSpecification)

FacilityResourceSpecification associations:

- An optional association with zero, one or many HardwarePartAsDesigned that realizes the FacilityResourceSpecification, ie, actual equipment that can be used when the task -will be performed (inherited from class ResourceSpecification)
- An optional association with zero, one or many TaskFacilityResourceBySpecifications, ie, individual usages of the specified facility resource in a specific Task or SubtaskByDefinition

FacilityResourceSpecification special recommendations:

- See special recommendations for ResourceSpecification, [Para 4.15.3.8](#)

4.15.3.14 TaskResourceByReferenceItem

The TaskResourceByReferenceItem <> is implemented by classes that represent hardware which can be used as a resource in a task.

Classes that implements the TaskResourceByReferenceItem <> are:

- HardwarePartAsDesigned
- HardwareElement

Classes that implement the TaskResourceByReferenceItem <> must also implement the following associations:

- An optional association with one or many instances of TaskMaterialResourceByReference
- An optional association with one or many instances of TaskFacilityResourceByReference

4.15.3.15 TaskPersonnelResource

The TaskPersonnelResource class is a specialization of TaskResource, and identifies personnel resources needed for the performance of a specific Task or SubtaskByDefinition.

Note

Each instance of TaskPersonnelResource is uniquely defined for a specific instance of Task or SubtaskByDefinition. Reuse of personnel resource definitions is enabled through the use of Trades, SkillLevels and Skills.

TaskPersonnelResource attributes:

- fixedResourceMarker (inherited from class TaskResource)
- taskResourceDuration (inherited from class TaskResource)
- taskPersonnelResourceRole
- taskNumberOfPersonnelResource (zero, one or many)
- taskPersonnelResourceLabourTime (zero, one or many)

TaskPersonnelResource associations:

- An association with the instance of Task or SubtaskByDefinition for which the associated personnel resource is needed (inherited from class TaskResource)
- An instance of TaskPersonnelResource can relate to one or many other instances of TaskResource, via the TaskResourceRelationship <> relationship class, eg, "person A uses tool B" (inherited from class TaskResource)
- An instance of TaskPersonnelResource can be related to from one or many other instances of TaskResource via the TaskResourceRelationship <> relationship class, eg, "tool B is used by person A" (inherited from class TaskResource)
- An optional association with zero, one or many competences (Trades, SkillLevels or Skills) that is required by the person that will the role of the TaskPersonnelResource (via the TaskPersonnelResourceCompetence <> relationship class)

TaskPersonnelResource special recommendations:

- The taskResourceDuration, taskNumberOfPersonnelResource and taskPersonnelResourceLabourTime attributes can have multiple values depending on, for example maintenance level or external conditions. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of taskResourceDuration, taskNumberOfPersonnelResource and taskPersonnelResourceLabourTime, respectively (ie, to the respective instance of PropertyType). Refer to [Para 4.22](#).
- See special recommendations for ResourceSpecification, [Para 4.15.3.8](#)

4.15.3.16 CompetenceDefinitionItem

The CompetenceDefinitionItem <> interface enables Skills, SkillLevels and Trades to be associated with the definition of a personnel resource that is needed to perform a Task or a SubtaskByDefinition.

Classes that implements the CompetenceDefinitionItem <> interface are:

- Trade
- Skill
- SkillLevel

Classes that implement the CompetenceDefinitionItem <> interface must also implement the following association:

- An optional association with zero, one or many instances of TaskPersonnelResource, ie, definitions of personnel resources needed for the performance of a specific Task or SubtaskByDefinition (via the TaskPersonnelResourceCompetence <> relationship class)

4.15.3.17 Trade

The Trade class identifies types of occupations.

Trade attributes:

- tradeName

Trade associations:

- An optional association with zero, one or many SkillLevels defined for the Trade

Trade interfaces:

- The Trade class implements the CompetenceDefinitionItem <> interface

4.15.3.18 Skill

The Skill class identifies specific skills as defined by codes according to agreed encoding systems.

Skill attributes:

- skillCode

Skill interfaces:

- The Skill class implements the CompetenceDefinitionItem <>interface>>

4.15.3.19 SkillLevel

The SkillLevel class defines specific skill levels for a defined Trade, eg, advanced, intermediate or master.

Note

An instance of class SkillLevel has no meaning by itself, but only in the context of a specific Trade.

SkillLevel attributes:

- skillLevelName
- skillLevelDescription (zero or one)

SkillLevel associations:

- An association with the Trade for which the SkillLevel is defined

SkillLevel interfaces:

- The SkillLevel class implements the CompetenceDefinitionItem <>interface>>

4.15.3.20 TaskPersonnelResourceCompetence

The TaskPersonnelResourceCompetence <>relationship>> class defines relationships between instances of TaskPersonnelResource and the specific competences that are required for the person that will play the role as the personnel resource.

TaskPersonnelResourceCompetence associations:

- (relating) The instance of TaskPersonnelResource which defines the need of a personnel resource with a specific competence
- (related) The competence that has the qualifications required for performing or participating in the task in the role as defined by the TaskPersonnelResource (via the CompetenceDefinitionItem <>interface>>)
- An optional association with zero, one or many instances of the AdditionalTrainingRequirements <>attributeGroup>>, defining additional training required for the defined competence in order to be qualified for the performance of, or participation in, the task in the role defined by the TaskPersonnelResource

Note

There is one instance of TaskPersonnelResourceCompetence per relevant combination of TaskPersonnelResource and competence (ie, Trade, Skill or SkillLevel).

TaskPersonnelResourceCompetence special recommendations:

- Where a TaskPersonnelResourceCompetence is dependent upon eg, customer, maintenance level, or external conditions, it must be distinguished by the assignment of an ApplicabilityStatement to the TaskPersonnelResourceCompetence instance. Refer to [Para 4.22](#).

4.15.3.21 AdditionalTrainingRequirement

The AdditionalTrainingRequirement <>attributeGroup>> identifies additional training required for the given competence (Trade, SkillLevel or Skill), in order for the defined competence to be qualified to play the role of the personnel resource.

Note

An instance of class AdditionalTrainingRequirement has no meaning by itself, but only in the context of an association between a specific required personnel resource, and a defined competence (ie, in the context of a specific instance of TaskPersonnelResourceCompetence).

AdditionalTrainingRequirement attributes:

- additionalTrainingRequirementDescription
- trainingMethod (zero or one)

AdditionalTrainingRequirement associations:

- An instance of AdditionalTrainingRequirement must always be associated with a specific instance of TaskPersonnelResourceCompetence

4.15.3.22 TaskDocumentResource

The TaskDocument class is a specialization of TaskResource, and identifies Documents needed for the performance of a specific Task or SubtaskByDefinition.

Note

Each instance of TaskDocumentResource is uniquely defined for a specific instance of Task or SubtaskByDefinition. Reuse of document resource definitions is enabled through the use of Documents.

TaskDocumentResource attributes:

- fixedResourceMarker (inherited from class TaskResource)
- taskResourceDuration (inherited from class TaskResource)

TaskDocumentResource associations:

- An association with a specific instance of Task or SubtaskByDefinition for which the associated document resource is needed (inherited from class TaskResource)
- An instance of TaskDocument can relate to one or many other instances of TaskResource via the TaskResourceRelationship <> class (inherited from class TaskResource)
- An instance of TaskDocument can be related to from one or many other instances of TaskResource via the TaskResourceRelationship <> class (inherited from class TaskResource)
- An association with the Document that will be used as a task resource

TaskDocumentResource special recommendations:

- See special recommendations for Task Resource, [Para 4.15.3.2](#)

4.15.4 UoF Task Resources - Additions to referenced class and interface definitions**4.15.4.1 Task**

The Task class defined in UoF Task must also implement the following additional <>:

- TaskResourceItem

4.15.4.2 SubtaskByDefinition

The SubtaskByDefinition class defined in UoF Task must also implement the following additional <>:

- TaskResourceItem

4.15.4.3 Document

The Document class defined in UoF Document must also implement the following additional association:

- An optional association with zero, one or many instances of TaskDocumentResource, ie, situations where the Document will be used as resource in a task

4.15.4.4 HardwarePartAsDesigned

The HardwarePartAsDesigned class defined in UoF Part Definition must also implement the following additional association:

- An optional association with zero, one or many instances of ResourceSpecification (via the ResourceRealization <>relationship> class), where the HardwarePartAsDesigned meets the requirements in a ResourceSpecification, and hence can be used as a resource in one or many tasks

The HardwarePartAsDesigned class defined in UoF Part Definition must also implement the following additional <>interface>:

- TaskResourceByReferenceItem

4.15.4.5 HardwareElement

The HardwareElement class defined in UoF Breakdown Element Realization must also implement the following additional <>interface>:

- TaskResourceByReferenceItem

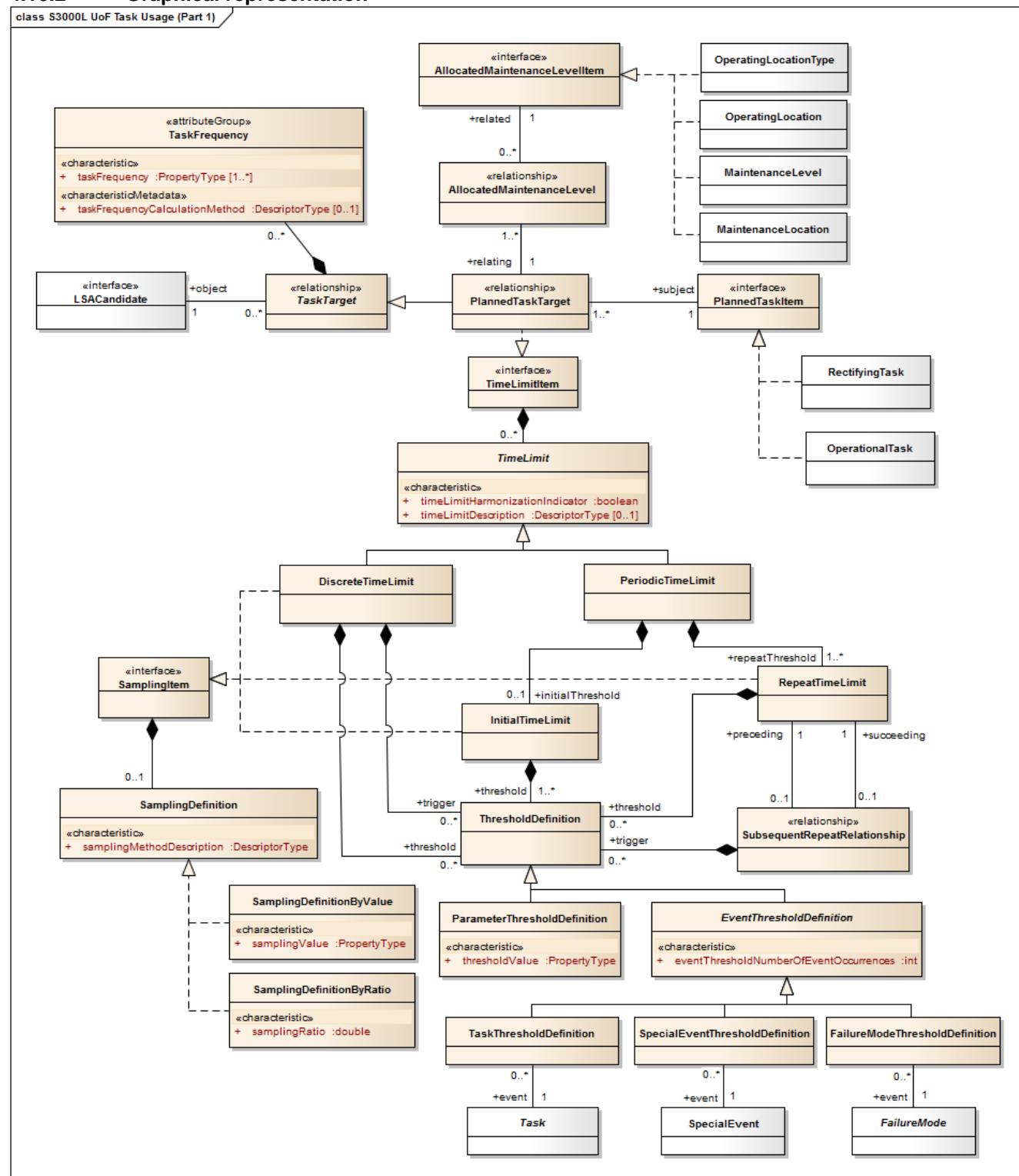
4.16 UoF Task Usage (Part 1)

4.16.1 Overall description

The Task Usage (Part 1) UoF, [Fig 33](#), supports detailed definitions for:

- Identifying the breakdown element or part on which an operational or rectifying task will be performed
- Thresholds and triggers defining the conditions that initiates an operational or rectifying task
- The frequency by which an operational or rectifying task is performed
- Where the operational or rectifying task will be performed

4.16.2 Graphical representation



ICN-B6865-S3000L0229-003-00

Fig 33 S3000L_UoF_Task_Usage (Part 1) - class model

4.16.3 UoF Task Usage (Part 1) - New class and interface definitions

4.16.3.1 TaskTarget

The TaskTarget <<relationship>> class defines a relationship between a Task and the object on which the Task will be performed. The object must be an instance of a class that implements the LSACandidate <<interface>>. Refer to [Para 4.9](#).

Note

TaskTarget is an abstract class, ie, an instantiation of TaskTarget needs to be either a PlannedTaskTarget or SupportingTaskTarget. Refer to [Para 4.17](#).

TaskTarget associations:

- (object) The task target item (LSACandidate) on which the related task will be performed
- An optional association with zero, one or many instances of the TaskFrequency <<attributeGroup>>

4.16.3.2 TaskFrequency

The TaskFrequency <<attributeGroup>> defines the frequency of performance or occurrence for the associated Task.

Note

An instance of class TaskFrequency has no meaning by itself, but only in the context of a specific TaskTarget (task usage).

Note

Task frequency (eg, annualization) must be defined whenever a task has more than one time limit.

TaskFrequency attributes:

- taskFrequency (one or many)
- taskFrequencyCalculationMethod (zero or one)

TaskFrequency associations:

- An instance of TaskFrequency must always be associated with a specific instance of TaskTarget, ie, a specific combination of Task and LSACandidate, on which the task will be performed

TaskFrequency special recommendations:

- Where a TaskFrequency is dependent upon eg, external conditions, customer, it must be distinguished by the assignment of an ApplicabilityStatement to the TaskFrequency instance. Refer to [Para 4.22](#).
- The taskFrequency attribute can have multiple values depending on, for example external conditions. These values must be distinguished by the assignment of an ApplicabilityStatement to the respective instance of taskFrequency ie, to the respective instance of PropertyType. Refer to [Para 4.22](#).

4.16.3.3 PlannedTaskItem

The PlannedTaskItem <<interface>> is implemented by classes that can be associated with a PlannedTaskTarget, ie, be associated with task limits and preferred locations for task execution.

Classes that implement the PlannedTaskItem <<interface>>:

- RectifyingTask
- OperationalTask

Classes that implement the PlannedTaskItem <<interface>> must also implement the following association:

- An association with one or many instances of PlannedTaskTarget

4.16.3.4 PlannedTaskTarget

The PlannedTaskTarget <<relationship>> class is a specialization of the TaskTarget class which adds information in terms of task limits and preferred support organization for task execution.

Note

There is one instance of PlannedTaskTarget per relevant combination of PlannedTaskItem and LSACandidate.

Note

The PlannedTaskTarget class must be used to associate RectifyingTasks and OperationalTasks with the items on which they will be performed.

PlannedTaskTarget interfaces:

- PlannedTaskTarget implements the TimeLimitItem <<interface>>

PlannedTaskTarget associations:

- (object) The task target item (instance of a class that implements the LSACandidate <<interface>>) on which the related task will be performed (inherited from class TaskTarget)
- (subject) The task (instance of RectifyingTask or OperationalTask) that is planned to be performed on the LSACandidate (instance of BreakdownElementRevision or Part)
- An optional association with zero, one or many instances of TaskFrequency (inherited from class TaskTarget)
- An association with one or many support organizations where the associated task can be performed on related LSACandidate (via the AllocatedMaintenanceLevel <<relationship>> class)
- An optional association with zero, one or many TimeLimits, ie, thresholds (intervals) that defines when the associated PlannedTaskItem will be performed on the LSACandidate (via the TimeLimitItem <<interface>>)

PlannedTaskTarget recommendations:

- Where an instance of PlannedTaskTarget is dependent upon eg, external conditions, customer, it must be distinguished by the assignment of an ApplicabilityStatement to the PlannedTaskTarget instance. Refer to [Para 4.22](#).

Note

Associating a PlannedTaskTarget with a specific maintenance level, by default means that any maintenance level above can carry out the same Task.

4.16.3.5 AllocatedMaintenanceLevel

The AllocatedMaintenanceLevel <<relationship>> class defines an association between an instance of PlannedTaskTarget and an identified support organization where the associated PlannedTaskItem will be carried out.

AllocatedMaintenanceLevel associations:

- (relating) The instance of PlannedTaskTarget (ie, specific combination of a RectifyingTask or OperationalTask and an LSACandidate) that will be performed at the related support organization
- (related) The support organization (MaintenanceLevel, MaintenanceLocation, OperatingLocationType, or OperatingLocation) that must have the capability to perform the associated task (via the AllocatedMaintenanceLevelItem <<interface>>)

Note

There is one instance of AllocatedMaintenanceLevel per relevant combination of PlannedTaskUsage and support organization.

AllocatedMaintenanceLevel special recommendations:

- Where the AllocatedMaintenanceLevel is dependent upon eg, operational scenario, it must be distinguished by the assignment of an ApplicabilityStatement to the AllocatedMaintenanceLevel instance. Refer to [Para 4.22](#).

4.16.3.6 AllocatedMaintenanceLevelItem

The AllocatedMaintenanceLevelItem <> is implemented by classes that can represent support organizations where a rectifying or operational task can be performed.

Classes that implement the AllocatedMaintenanceLevelItem <> are:

- MaintenanceLevel
- MaintenanceLocation
- OperatingLocationType
- OperatingLocation

Classes that implement the AllocatedMaintenanceLevelItem <> must also implement the following association:

- An optional association with zero, one or many instances of AllocatedMaintenanceLevel

4.16.3.7 TimeLimit

The TimeLimit class identifies thresholds (intervals) which defines when the associated task (via PlannedTaskTarget) will be performed on the identified LSACandidate.

Note

An instance of class TimeLimit has no meaning by itself, but only in the context of a specific PlannedTaskTarget or TaskRequirementTarget. Refer to [Para 4.13](#).

Note

TimeLimit is an abstract class, ie, an instantiation of TimeLimit must be an instantiation of either DiscreteTimeLimit or PeriodicTimeLimit.

TimeLimit attributes:

- timeLimitHarmonizationIndicator
- timeLimitDescription (zero or one)

TimeLimit associations:

- An instance of TimeLimit must always be associated with a specific instance of a class that implements the TimeLimitItem <>, ie, an instance of either PlannedTaskTarget or TaskRequirementTarget. Refer to [Para 4.13](#).

TimeLimit special recommendations:

- Where an instance of TimeLimit is dependent upon eg, external conditions or customer, it must be distinguished by the assignment of an ApplicabilityStatement to the TimeLimit instance. Refer to [Para 4.22](#).

4.16.3.8 TimeLimitItem

The TimeLimitItem <> is implemented by classes that can have an associated interval or limit that must not be exceeded.

Classes that implement the TimeLimitItem <> are:

- TaskRequirementTarget. Refer to [Para 4.13](#).
- PlannedTaskTarget

Classes that implement the TimeLimitItem <<interface>> must also implement the following association:

- An optional association with zero, one or many instances of TimeLimit

4.16.3.9 SamplingDefinition

The SamplingDefinition class identifies the method to be used for selecting a limited set of manufactured products, on which a task will be performed when a specific time limit has been reached.

Note

An instance of class SamplingDefinition has no meaning by itself, but only in the context of a specific DiscreteTimeLimit, InitialTimeLimit or RepeatTimeLimit (via the SamplingItem <<interface>>).

Note

An instance of SamplingDefinition can be (but need not be) specialized into either SamplingDefinitionByRatio or SamplingDefinitionByValue.

SamplingDefinition attributes:

- samplingMethodDescription

SamplingDefinition associations:

- An instance of SamplingDefinition must always be associated with a specific instance of a class that implements the SamplingItem <<interface>>

4.16.3.10 SamplingDefinitionByRatio

The SamplingDefinitionByRatio class is a specialization of the SamplingDefinition class, and identifies that the method used for selecting manufactured products is done by ratio.

Note

An instance of class SamplingDefinitionByRatio has no meaning by itself, but only in the context of a specific DiscreteTimeLimit, InitialTimeLimit or RepeatTimeLimit.

SamplingDefinitionByRatio attributes:

- samplingMethodDescription (inherited from class SamplingDefinition)
- samplingRatio

SamplingDefinitionByRatio associations:

- An instance of SamplingDefinitionByRatio must always be associated with a specific instance of a class that implements the SamplingItem <<interface>> (inherited from class SamplingDefinition)

4.16.3.11 SamplingDefinitionByValue

The SamplingDefinitionByValue class is a specialization of the SamplingDefinition class, and identifies that the method used for selecting manufactured products is done by value, ie, a definitive number of manufactured products.

Note

An instance of class SamplingDefinitionByValue has no meaning by itself, but only in the context of a specific DiscreteTimeLimit, InitialTimeLimit or RepeatTimeLimit.

SamplingDefinitionByValue attributes:

- samplingMethodDescription (inherited from class SamplingDefinition)
- samplingValue

SamplingDefinitionByValue associations:

- An instance of SamplingDefinitionByValue must always be associated with a specific instance of a class that implements the SamplingItem <> (inherited from class SamplingDefinition)

4.16.3.12 SamplingItem

The SamplingItem <> is implemented by classes that can be associated with a defined sampling method.

Classes that implement the Sampling <> are:

- DiscreteTimeLimit
- InitialTimeLimit
- RepeatTimeLimit

Classes that implement the Sampling interface must implement the following associations:

- An optional association with zero or one instance of SamplingDefinition (or subclasses thereof)

4.16.3.13 DiscreteTimeLimit

The DiscreteTimeLimit class is a specialization of the TimeLimit class, and is used to identify limits that are distinct from each other, ie, there is no scheduled next occurrence for the associated task.

Note

An instance of class DiscreteTimeLimit has no meaning by itself, but only in the context of a specific instance of PlannedTaskTarget or TaskRequirementTarget.

Note

DiscreteTimeLimit need not contain a computable expression, ie, it can just contain a timeLimitDescription without any computable triggers or thresholds.

DiscreteTimeLimit attributes:

- timeLimitHarmonizationIndicator (inherited from class TimeLimit)
- timeLimitDescription (inherited from class TimeLimit)

DiscreteTimeLimit implements the following <>:

- SamplingItem

DiscreteTimeLimit associations:

- An instance of DiscreteTimeLimit must always be associated with a specific instance of PlannedTaskTarget or TaskRequirementTarget (inherited from class TimeLimit)
- An optional trigger association with zero, one or many instances of ThresholdDefinition
- An optional threshold association with zero, one or many instances of ThresholdDefinition

Note

The threshold association defines a specified interval, from a defined event, to the next required performance of the PlannedTaskTarget. The defined event is identified by the trigger association.

Note

The trigger association describes the event to which the threshold is related, eg, bird strike or engine replacement. If the DiscreteTimeLimit does not define any trigger association, it

must be assumed that the trigger event equals the hardwarePartMaintenanceStart defined for HardwarePart in UoF Part Definition. The trigger activates the threshold.

Note

Multiple instances of ThresholdDefinitions for both the trigger and the threshold association mean 'whichever is reached first'.

DiscreteTimeLimit special recommendations:

- Where an instance of DiscreteTimeLimit is dependent upon eg, external conditions or customer, it must be distinguished by the assignment of an ApplicabilityStatement to the DiscreteTimeLimit instance. Refer to [Para 4.22](#).

Note

The DiscreteTimeLimit can be used to define "Perform once" and "On condition" time limits as defined in S1000D.

4.16.3.14 PeriodicTimeLimit

The PeriodicTimeLimit class is a specialization of the TimeLimit class, and is used to identify limits that are repeated with a specific interval, ie, there is a next scheduled occurrence for the associated task.

Note

An instance of class PeriodicTimeLimit has no meaning by itself, but only in the context of a specific instance of PlannedTaskTarget or TaskRequirementTarget.

PeriodicTimeLimit attributes:

- timeLimitHarmonizationIndicator (inherited from class TimeLimit)
- timeLimitDescription (inherited from class TimeLimit)

PeriodicTimeLimit associations:

- An instance of PeriodicTimeLimit must always be associated with a specific instance of PlannedTaskTarget or TaskRequirementTarget (inherited from class TimeLimit)
- An optional initial threshold association with an instance of InitialTimeLimit
- An association with one or many instances of RepeatTimeLimit

Note

The InitialTimeLimit association defines a specified interval for the first required performance of the PlannedTaskItem (ie, RectifyingTask or OperationalTask). The interval must be assumed to be triggered by the condition that is defined as the hardwarePartMaintenanceStart for HardwarePart in UoF Part Definition. Refer to [Para 4.4](#).

Note

Multiple instances of RepeatTimeLimit for the repeat association require that each instance of RepeatTimeLimit is ordered using the SubsequentRepeatRelationship class.

PeriodicTimeLimit special recommendations:

- Where an instance of PeriodicTimeLimit is dependent upon eg, external conditions or customer, it must be distinguished by the assignment of an ApplicabilityStatement to the PeriodicTimeLimit instance. Refer to [Para 4.22](#).

Note

The PeriodicTimeLimit class can be used to define explicit combinations of initial "Perform once" and "Perform periodically" time limits as defined in S1000D.

4.16.3.15 InitialTimeLimit

The InitialTimeLimit class defines a specific interval for the first scheduled performance of a periodic task.

Note

An instance of class InitialTimeLimit has no meaning by itself, but only in the context of a specific PeriodicTimeLimit.

InitialTimeLimit implements the following <>interface>>:

- SamplingItem

InitialTimeLimit associations:

- An instance of InitialTimeLimit is always associated with a specific instance of PeriodicTimeLimit
- A threshold association with one or many instances of ThresholdDefinition

Note

The threshold association defines a specified interval for the first performance of the associated RectifyingTask or OperationalTask.

Note

Multiple instances of ThresholdDefinition for the threshold association mean "whichever is reached first".

4.16.3.16 RepeatTimeLimit

The RepeatTimeLimit class defines task limits that are repeated with a specific interval.

Note

An instance of class RepeatTimeLimit has no meaning by itself, but only in the context of a specific PeriodicTimeLimit.

Note

RepeatTimeLimit must contain a computable expression.

RepeatTimeLimit implements the following <>interface>>:

- SamplingItem

RepeatTimeLimit associations:

- An instance of RepeatTimeLimit is always associated with a specific instance of PeriodicTimeLimit
- A threshold association with one or many instances of ThresholdDefinition
- An optional association with zero or one succeeding instance of RepeatTimeLimit, via the SubsequentRepeatRelationship <>relationship>> class
- An optional association with zero or one preceding instance of RepeatTimeLimit, via the SubsequentRepeatRelationship <>relationship>> class

Note

The threshold association defines a specified interval for the repeated performance of the associated RectifyingTask or OperationalTask.

Note

Multiple instances of ThresholdDefinition for the threshold association mean 'whichever is reached first'.

Note

The RepeatTimeLimit class can be used to define "Perform periodically" time limits as defined in S1000D.



4.16.3.17 SubsequentRepeatRelationship

The SubsequentRepeatRelationship class defines an ordered relationship between two instances of RepeatTimeLimit, where the succeeding RepeatTimeLimit replaces the preceding RepeatTimeLimit at a specific point defined by the SubsequentRepeatRelationship trigger.

Note

The definition of when an instance of RepeatTimeLimit will be succeeded by another instance of RepeatTimeLimit (ie, replaced by another instance) is determined by the instance of ThresholdDefinition defined via the SubsequentRepeatRelationship trigger association.

SubsequentRepeatRelationship associations:

- (preceding) The instance of RepeatTimeLimit that is replaced by the succeeding RepeatTimeLimit
- (succeeding) The instance of RepeatTimeLimit that is replacing the preceding RepeatTimeLimit
- A trigger association with one or many instances of ThresholdDefinition

Note

The trigger association describes the event when the instance of RepeatTimeLimit identified by the succeeding association will replace the instance of RepeatTimeLimit identified by the preceding association.

Note

Multiple instances of ThresholdDefinition for the trigger association mean "whichever is reached first".

4.16.3.18 ThresholdDefinition

The ThresholdDefinition class defines the value or event that constitutes a threshold or trigger.

Note

An instance of class ThresholdDefinition has no meaning by itself, but only in the context of a specific instance of either:

- DiscreteTimeLimit, as a trigger
- DiscreteTimeLimit, as a threshold
- InitialTimeLimit, as an initial threshold
- RepeatTimeLimit, as a threshold
- SubsequentRepeatRelationship, as a trigger

Note

ThresholdDefinition is an abstract class, ie, an instantiation of ThresholdDefinition must be either a ParameterThresholdDefinition or an EventThresholdDefinition.

ThresholdDefinition associations:

- An instance of ThresholdDefinition can only be associated with one specific instance of one of the following:
 - DiscreteTimeLimit, as a trigger
 - DiscreteTimeLimit, as a threshold
 - InitialTimeLimit, as an initial threshold
 - RepeatTimeLimit, as a threshold
 - SubsequentRepeatRelationship, as a trigger

4.16.3.19 ParameterThresholdDefinition

The ParameterThresholdDefinition class is a specialization of class ThresholdDefinition and is used to represent a value with unit that constitutes a threshold or trigger.

Note

An instance of class ParameterThresholdDefinition has no meaning by itself, but only in the context of a specific instance of either:

- DiscreteTimeLimit, as a trigger
- DiscreteTimeLimit, as a threshold
- InitialTimeLimit, as an initial threshold
- RepeatTimeLimit, as a threshold
- SubsequentRepeatRelationship, as a trigger

Note

An example of ParameterThresholdDefinition is 1000 flight hours.

ParameterThresholdDefinition attributes:

- thresholdValue

ParameterThresholdDefinition associations:

- An instance of ParameterThresholdDefinition is always associated with a specific instance of one of the following (inherited from class ThresholdDefinition):
 - DiscreteTimeLimit, as a trigger
 - DiscreteTimeLimit, as a threshold
 - InitialTimeLimit, as an initial threshold
 - RepeatTimeLimit, as a threshold
 - SubsequentRepeatRelationship, as a trigger

4.16.3.20 EventThresholdDefinition

The EventThresholdDefinition class is a specialization of class ThresholdDefinition and is used to represent the number of occurrences of an explicit event that constitutes a threshold or trigger.

Note

An instance of class EventThresholdDefinition has no meaning by itself, but only in the context of a specific instance of either:

- DiscreteTimeLimit, as a trigger
- DiscreteTimeLimit, as a threshold
- InitialTimeLimit, as an initial threshold
- RepeatTimeLimit, as a threshold
- SubsequentRepeatRelationship, as a trigger

Note

EventThresholdDefinition is an abstract class, ie, an instantiation of EventThresholdDefinition must be either a TaskThresholdDefinition, a SpecialEventThresholdDefinition or a FailureModeThresholdDefinition.

EventThresholdDefinition attributes:

- eventThresholdNumberOfEventOccurrences

EventThresholdDefinition associations:

- An instance of EventThresholdDefinition is always associated with a specific instance of one of the following (inherited from class ThresholdDefinition):
 - DiscreteTimeLimit, as a trigger
 - DiscreteTimeLimit, as a threshold
 - InitialTimeLimit, as an initial threshold
 - RepeatTimeLimit, as a threshold

- SubsequentRepeatRelationship, as a trigger

4.16.3.21 FailureModeThresholdDefinition

The FailureModeThresholdDefinition class is a specialization of class EventThresholdDefinition. FailureModeThresholdDefinition is used to represent the number of occurrences of a specific FailureMode (UoF LSA FMEA) that constitutes the threshold or trigger. Refer to [Para 4.11](#).

Note

An instance of class FailureModeThresholdDefinition has no meaning by itself, but only in the context of a specific instance of either:

- DiscreteTimeLimit, as a trigger
- DiscreteTimeLimit, as a threshold
- InitialTimeLimit, as an initial threshold
- RepeatTimeLimit, as a threshold
- SubsequentRepeatRelationship, as a trigger

FailureModeThresholdDefinition attributes:

- eventThresholdNumberOfEventOccurrences (inherited from class EventThresholdDefinition)

FailureModeThresholdDefinition associations:

- An instance of FailureModeThresholdDefinition is always associated with a specific instance of one of the following (inherited from class ThresholdDefinition):
 - DiscreteTimeLimit, as a trigger
 - DiscreteTimeLimit, as a threshold
 - InitialTimeLimit, as an initial threshold
 - RepeatTimeLimit, as a threshold
 - SubsequentRepeatRelationship, as a trigger.
- An association with an instance of FailureMode that constitutes the threshold or trigger.

4.16.3.22 SpecialEventThresholdDefinition

The SpecialEventThresholdDefinition class is a specialization of class EventThresholdDefinition. SpecialEventThresholdDefinition is used to represent the number of occurrences of a specific SpecialEvent (UoF Special Event And Damage) that constitutes the threshold or trigger.

Note

An instance of class SpecialEventThresholdDefinition has no meaning by itself, but only in the context of a specific instance of either:

- DiscreteTimeLimit, as a trigger
- DiscreteTimeLimit, as a threshold
- InitialTimeLimit, as an initial threshold
- RepeatTimeLimit, as a threshold
- SubsequentRepeatRelationship, as a trigger

SpecialEventThresholdDefinition attributes:

- eventThresholdNumberOfEventOccurrences (inherited from class EventThresholdDefinition)

SpecialEventThresholdDefinition associations:

- An instance of SpecialEventThresholdDefinition is always associated with a specific instance of one of the following (inherited from class ThresholdDefinition):

- DiscreteTimeLimit, as a trigger
 - DiscreteTimeLimit, as a threshold
 - InitialTimeLimit, as an initial threshold
 - RepeatTimeLimit, as a threshold
 - SubsequentRepeatRelationship, as a trigger
- An association with an instance of SpecialEvent that constitutes the threshold or trigger

4.16.3.23 TaskThresholdDefinition

The TaskThresholdDefinition class is a specialization of class EventThresholdDefinition. TaskThresholdDefinition is used to represent the number of occurrences of a specific Task (UoF Task) that constitutes the threshold or trigger. Refer to [Para 4.14](#).

Note

An instance of class TaskThresholdDefinition has no meaning by itself, but only in the context of a specific instance of either:

- DiscreteTimeLimit, as a trigger
- DiscreteTimeLimit, as a threshold
- InitialTimeLimit, as an initial threshold
- RepeatTimeLimit, as a threshold
- SubsequentRepeatRelationship, as a trigger

TaskThresholdDefinition attributes:

- eventThresholdNumberOfEventOccurrences (inherited from class EventThresholdDefinition)

TaskThresholdDefinition associations:

- An instance of TaskThresholdDefinition is always associated with a specific instance of one of the following (inherited from class ThresholdDefinition):
 - DiscreteTimeLimit, as a trigger
 - DiscreteTimeLimit, as a threshold
 - InitialTimeLimit, as an initial threshold
 - RepeatTimeLimit, as a threshold
 - SubsequentRepeatRelationship, as a trigger
- An association with an instance of Task that constitutes the threshold or trigger

4.16.4 UoF Task Usage (Part 1) - Additions to referenced class and interface definitions

4.16.4.1 LSACandidate

Classes that implement the LSACandidate <> interface defined in UoF LSA Candidate must also implement the following additional association:

- An optional association with zero, one or many instances of Task which will be performed on the LSA candidate (via instances of the TaskTarget <> relationship class)

4.16.4.2 Task

The Task class defined in UoF Task must also implement the following additional association:

- An optional association with zero, one or many instances of TaskThresholdDefinition

4.16.4.3 RectifyingTask

The RectifyingTask class defined in UoF Task must also implement the following additional <> interface:

- PlannedTaskItem

OperationalTask

The OperationalTask class defined in UoF Task must also implement the following additional <>interface>>:

- PlannedTaskItem

4.16.4.4 MaintenanceLevel

The MaintenanceLevel class defined in UoF Product Usage Context must also implement the following additional <>interface>>:

- AllocatedMaintenanceLevelItem

4.16.4.5 MaintenanceLocation

The MaintenanceLocation class defined in UoF Product Usage Context must also implement the following additional <>interface>>:

- AllocatedMaintenanceLevelItem

4.16.4.6 OperatingLocationType

The OperatingLocationType class defined in UoF Product Usage Context must also implement the following additional <>interface>>:

- AllocatedMaintenanceLevelItem

4.16.4.7 OperatingLocation

The OperatingLocation class defined in UoF Product Usage Context must also implement the following additional <>interface>>:

- AllocatedMaintenanceLevelItem

4.16.4.8 SpecialEvent

The SpecialEvent class defined in UoF Special Event And Damage must also implement the following additional association:

- An optional association with zero, one or many instances of SpecialEventThresholdDefinition

4.16.4.9 FailureMode

The FailureMode class defined in UoF LSA FMEA must also implement the following additional association:

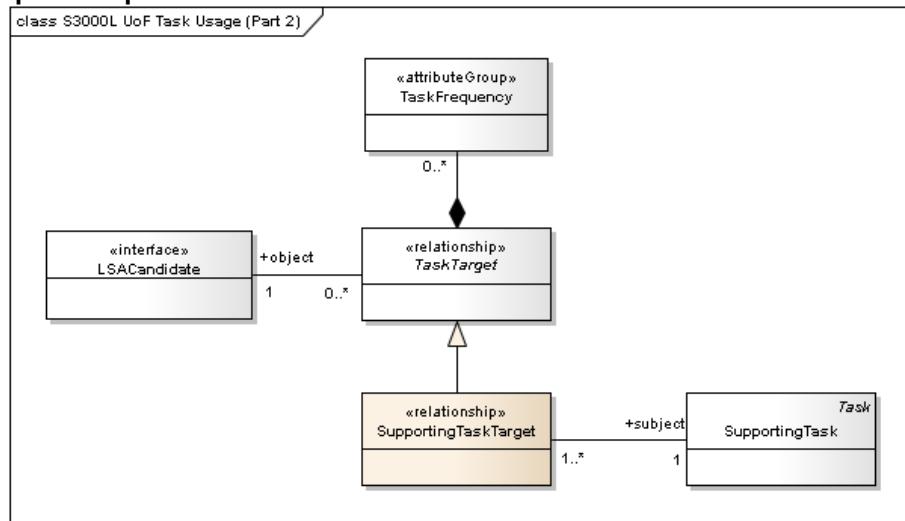
- An optional association with zero, one or many instances of FailureModeThresholdDefinition

4.17 UoF Task Usage (Part 2)

4.17.1 Overall description

The Task Usage (Part 2) UoF, [Fig 34](#), supports the creation of associations between supporting tasks and the objects on which the supporting task will be performed.

4.17.2 Graphical representation



ICN-B6865-S3000L0230-002-00

Fig 34 S3000L UoF Task Usage (Part 2) - class model

4.17.3 UoF Task Usage (Part 2) - New class and interface definitions

4.17.3.1 SupportingTaskTarget

The SupportingTaskTarget <<relationship>> class is a specialization of the TaskTarget class. SupportingTaskTarget defines associations between instances of SupportingTask and specific task target items (ie, instances of the classes that implement the LSACandidate <<interface>>) on which the SupportingTask will be performed.

Note

There is one instance of SupportingTaskTarget per relevant combination of SupportingTask and LSACandidate.

SupportingTaskTarget associations:

- (object) The task target item (instance of a class that implement the LSACandidate <<interface>>) on which the related supporting task will be performed (inherited from class TaskTarget)
- (subject) The SupportingTask instance that will be performed on the task target item (instance of a class that implement the LSACandidate <<interface>>)
- An optional association with zero, one or many instances of TaskFrequency (inherited from class TaskUsage)

4.17.4 UoF Task Usage (Part 2) - Additions to referenced class and interface definitions

4.17.4.1 SupportingTask

The SupportingTask class defined in UoF Task must also implement the following additional association:

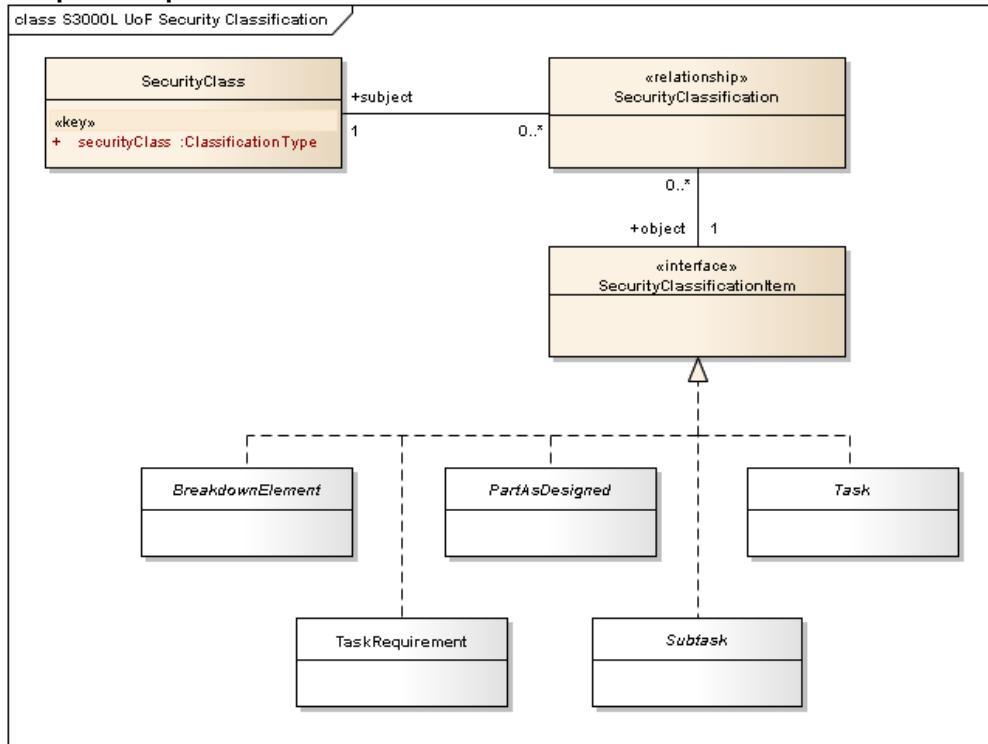
- An association with one or many instances of SupportingTaskTarget in order to identify the items on which the SupportingTask will be performed

4.18 UoF Security Classification

4.18.1 Overall description

The Security Classification UoF, [Fig 35](#), supports the assignment of security classes to objects that need special handling with respect to eg, availability, distribution or presentation.

4.18.2 Graphical representation



ICN-B6865-S3000L0231-003-00

Fig 35 S3000L UoF Security Classification - class model

4.18.3 UoF Security Classification - New class and interface definitions

4.18.3.1 SecurityClass

The `SecurityClass` class identifies security classes that can be assigned to objects within an LSA program.

`SecurityClass` attributes:

- `securityClass`

`SecurityClass` associations:

- An optional association with zero, one or many instances of `BreakdownElement`, `PartAsDesigned`, `Task`, `Subtask` and/or `TaskRequirement` (ie, classes that implement the `SecurityClassificationItem <<interface>>`)

Note

Security classification system to be used must be determined by the project.

4.18.3.2 SecurityClassificationItem

The `SecurityClassificationItem <<interface>>` is implemented by classes that can be candidates for security classification.

Classes that implements the `SecurityClassificationItem <<interface>>` are:

- `BreakdownElement`
- `PartAsDesigned`
- `Task`
- `Subtask`
- `TaskRequirement`

Classes that implement the SecurityClassificationItem <<interface>> must also implement the following association:

- An optional association with zero, one or many instances of SecurityClass (via the SecurityClassification <<relationship>> class)

4.18.3.3 SecurityClassification

The SecurityClassification <<relationship>> class defines a relationship between an instance of SecurityClass and the object that is classified (ie, an instance of a class that implements the SecurityClassificationItem <<interface>>).

SecurityClassification associations:

- (subject) The SecurityClass being applied to the classified item (instance of a class that implements the SecurityClassificationItem <<interface>>)
- (object) The classified item (instance of a class that implements the SecurityClassificationItem <<interface>>) to which the SecurityClass applies

Note

There is one instance of SecurityClassification per relevant combination of SecurityClass and instance to which the SecurityClass is applied (ie, relevant instances of BreakdownElement, PartAsDesigned, Task, Subtask or TaskRequirement).

SecurityClassification special recommendations

- Where an instance of SecurityClassification is dependent upon eg, customer, it must be distinguished by the assignment of an ApplicabilityStatement to the SecurityClassification instance. Refer to [Para 4.22](#).

4.18.4 UoF Security Classification - Additions to referenced class and interface definitions

4.18.4.1 BreakdownElement

The BreakdownElement class defined in UoF Breakdown Structure must also implement the following additional <<interface>>:

- SecurityClassificationItem

4.18.4.2 PartAsDesigned

The PartAsDesigned class defined in UoF Part Definition must also implement the following additional <<interface>>:

- SecurityClassificationItem

4.18.4.3 Task

The Task class defined in UoF Task must also implement the following additional <<interface>>:

- SecurityClassificationItem

4.18.4.4 Subtask

The Subtask class defined in UoF Task must also implement the following additional <<interface>>:

- SecurityClassificationItem

4.18.4.5 TaskRequirement

The TaskRequirement class defined in UoF LSA Candidate Task Requirement must also implement the following additional <<interface>>:

- SecurityClassificationItem

4.19 UoF Organization Assignment

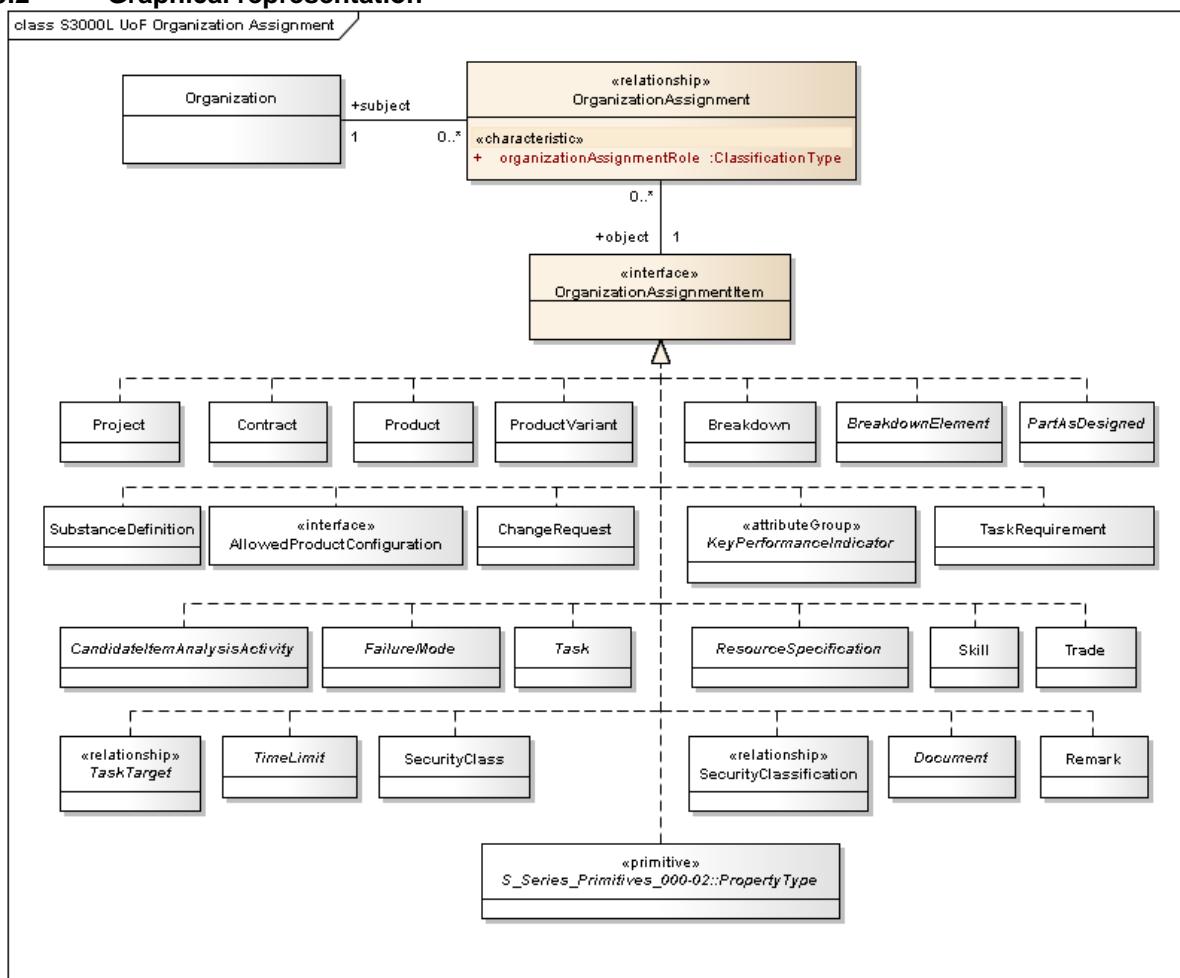
4.19.1 Overall description

The Organization Assignment UoF, [Fig 36](#), supports a flexible way of assigning different types of organizational information to different types of objects in the data model.

Note

There are some previously proposed (even mandatory) associations with the Organization class in the data model. This UoF adds the ability to define any type of additional organizational information which is of relevance for an instance of classes that implement the OrganizationAssignmentItem interface. An example of additional organizational information can be design responsible organization for a specific part. This is not explicitly defined in the data model but can be added using this UoF, by setting the OrganizationAssignment attribute organizationAssignmentRole to 'designAuthority'.

4.19.2 Graphical representation



ICN-B6865-S3000L0232-003-00

Fig 36 S3000L UoF Organization Assignment - class model

4.19.3 UoF Organization Assignment - New class and interface definitions

4.19.3.1 OrganizationAssignmentItem

The OrganizationAssignmentItem <>interface>> is implemented by classes that can be associated with optional organizational information.

Classes and <>interfaces>> that implement the OrganizationAssignmentItem <>interface>> are:

Applicable to: All

S3000L-A-19-00-0000-00A-040A-A

Chap 19

- UoF Product and Project
 - Project
 - Contract
 - Product
 - ProductVariant
- UoF Breakdown Structure
 - Breakdown
 - BreakdownElement
- UoF Part Definition
 - PartAsDesigned
 - SubstanceDefinition
- UoF Product Design Configuration
 - AllowedProductConfiguration
- UoF LSA Candidate
 - KeyPerformanceIndicator
- UoF LSA Candidate Analysis Activity
 - CandidateItemAnalysisActivity
- UoF LSA FMEA
 - FailureMode
- UoF LSA Candidate Task Requirement
 - ChangeRequest
 - TaskRequirement
- UoF Task
 - Task
- UoF Task Resources
 - ResourceSpecification
 - Trade
 - Skill
- UoF Task Usage (Part 1)
 - TaskTarget
 - TimeLimit
- UoF Security Classification
 - SecurityClass
 - SecurityClassification
- UoF Document
 - Document
- UoF Remark
 - Remark
- S-Series Primitives (Data Types)
 - PropertyType

Note

Organizations can also be assigned to any class that is a subclass of the enumerated classes, eg, to a RectifyingTask which is a subclass of Task.

Classes that implement the OrganizationAssignmentItem <>interface>> must also implement the following association:

- An optional association with zero, one or many instances of Organization (via the OrganizationAssignment <>relationship>> class)

4.19.3.2 OrganizationAssignment

The OrganizationAssignment <>relationship>> class defines a relationship between a specific Organization and an instance of any class that implements the OrganizationAssignmentItem <>interface>>.

Note

The role of the Organization being assigned is determined by organizationAssignmentRole attribute.

OrganizationAssignment attributes:

- organizationAssignmentRole

OrganizationAssignment associations:

- (subject) The Organization being associated with an object (instance of any class that implements the OrganizationAssignmentItem <>)
- (object) The instance to which the Organization is assigned

Note

There is one instance of OrganizationAssignment per relevant combination of Organization and instance to which the Organization is being assigned (ie, instance of any class that implements the OrganizationAssignmentItem <>).

OrganizationAssignment special recommendations:

- Where an instance of OrganizationAssignment is dependent upon eg, customer, it must be distinguished by the assignment of an ApplicabilityStatement to the OrganizationAssignment instance. Refer to [Para 4.22](#).

4.19.4 UoF Organization Assignment - Additions to referenced class and interface definitions

4.19.4.1 Project

The Project class defined in UoF Product and Project must also implement the following additional <>:

- OrganizationAssignmentItem

4.19.4.2 Contract

The Contract class defined in UoF Product and Project must also implement the following additional <>:

- OrganizationAssignmentItem

4.19.4.3 Product

The Product class defined in UoF Product and Project must also implement the following additional <>:

- OrganizationAssignmentItem

4.19.4.4 ProductVariant

The ProductVariant class defined in UoF Product and Project must also implement the following additional <>:

- OrganizationAssignmentItem

4.19.4.5 Organization

The Organization class defined in UoF Product and Project must also implement the following additional association:

- An optional association with zero, one or many instances of any class that implement the OrganizationAssignmentItem <> (via the OrganizationAssignment <> class)

-
- 4.19.4.6 **Breakdown**
The Breakdown class defined in UoF Breakdown Structure must also implement the following additional <>interface>>:
– OrganizationAssignmentItem
- 4.19.4.7 **BreakdownElement**
The BreakdownElement class defined in UoF Breakdown Structure must also implement the following additional <>interface>>:
– OrganizationAssignmentItem
- 4.19.4.8 **PartAsDesigned**
The PartAsDesigned class defined in UoF Part Definition must also implement the following additional <>interface>>:
– OrganizationAssignmentItem
- 4.19.4.9 **SubstanceDefinition**
The SubstanceDefinition class defined in UoF Part Definition must also implement the following additional <>interface>>:
– OrganizationAssignmentItem
- 4.19.4.10 **AllowedProductConfiguration**
Classes that implement the AllowedProductConfiguration <>interface>> defined in UoF Product Design Configuration must also implement the following additional <>interface>>:
– OrganizationAssignmentItem
- 4.19.4.11 **KeyPerformanceIndicator**
The KeyPerformanceIndicator <>attributeGroup>> defined in UoF LSA Candidate must also implement the following additional <>interface>>:
– OrganizationAssignmentItem
- 4.19.4.12 **CandidateItemAnalysisActivity**
The CandidateItemAnalysisActivity class defined in UoF LSA Candidate Analysis Activity must also implement the following additional <>interface>>:
– OrganizationAssignmentItem
- 4.19.4.13 **FailureMode**
The FailureMode class defined in UoF LSA FMEA must also implement the following additional <>interface>>:
– OrganizationAssignmentItem
- 4.19.4.14 **ChangeRequest**
The ChangeRequest class defined in UoF LSA Candidate Task Requirement must also implement the following additional <>interface>>:
– OrganizationAssignmentItem
- 4.19.4.15 **TaskRequirement**
The TaskRequirement class defined in UoF LSA Candidate Task Requirement must also implement the following additional <>interface>>:
– OrganizationAssignmentItem

4.19.4.16 Task

The Task class defined in UoF Task must also implement the following additional <><interface>>:

- OrganizationAssignmentItem

4.19.4.17 ResourceSpecification

The ResourceSpecification class defined in UoF Task Resources must also implement the following additional <><interface>>:

- OrganizationAssignmentItem

4.19.4.18 Trade

The Trade class defined in UoF Task Resources must also implement the following additional <><interface>>:

- OrganizationAssignmentItem

4.19.4.19 Skill

The Skill class defined in UoF Task Resources must also implement the following additional <><interface>>:

- OrganizationAssignmentItem

4.19.4.20 TaskTarget

The TaskTarget <><relationship>> class defined in UoF Task Usage (Part 1) must also implement the following additional <><interface>>:

- OrganizationAssignmentItem

4.19.4.21 TimeLimit

The TimeLimit class defined in UoF Task Usage (Part 1) must also implement the following additional <><interface>>:

- OrganizationAssignmentItem

4.19.4.22 SecurityClass

The SecurityClass class defined in UoF Security Classification must also implement the following additional <><interface>>:

- OrganizationAssignmentItem

4.19.4.23 SecurityClassification

The SecurityClassification <><relationship>> class defined in UoF Security Classification must also implement the following additional <><interface>>:

- OrganizationAssignmentItem

4.19.4.24 Document

The Document class defined in UoF Document must also implement the following additional <><interface>>:

- OrganizationAssignmentItem

4.19.4.25 Remark

The Remark class defined in UoF Remark must also implement the following additional <><interface>>:

- OrganizationAssignmentItem

4.19.4.26 PropertyType

The PropertyType <> primitive defined in S-Series Primitives (Data Types) must also implement the following additional <> interface:

- OrganizationAssignmentItem

Note

The assignment of an Organization to an instance of PropertyType enables the recording of organizational information against any property value, eg, to identify the Organization that recorded a specific property value.

4.20 UoF Document

4.20.1 Overall description

The Document UoF, [Fig 37](#) and [Fig 38](#), supports a flexible way of assigning documents or document issues to different types of objects in the data model.

Note

There are some previously proposed (even mandatory) associations with documents in the data model. This UoF adds the ability to define any type of additional document information which is of relevance for an instance of classes that implement the DocumentAssignmentItem <> interface. An example of additional document information can be a 'Design drawing' for a Part. This is not explicit in the data model but can be added using this UoF, by setting the documentAssignmentRole attribute of class DocumentAssignment to eg, 'designDrawing'.

Note

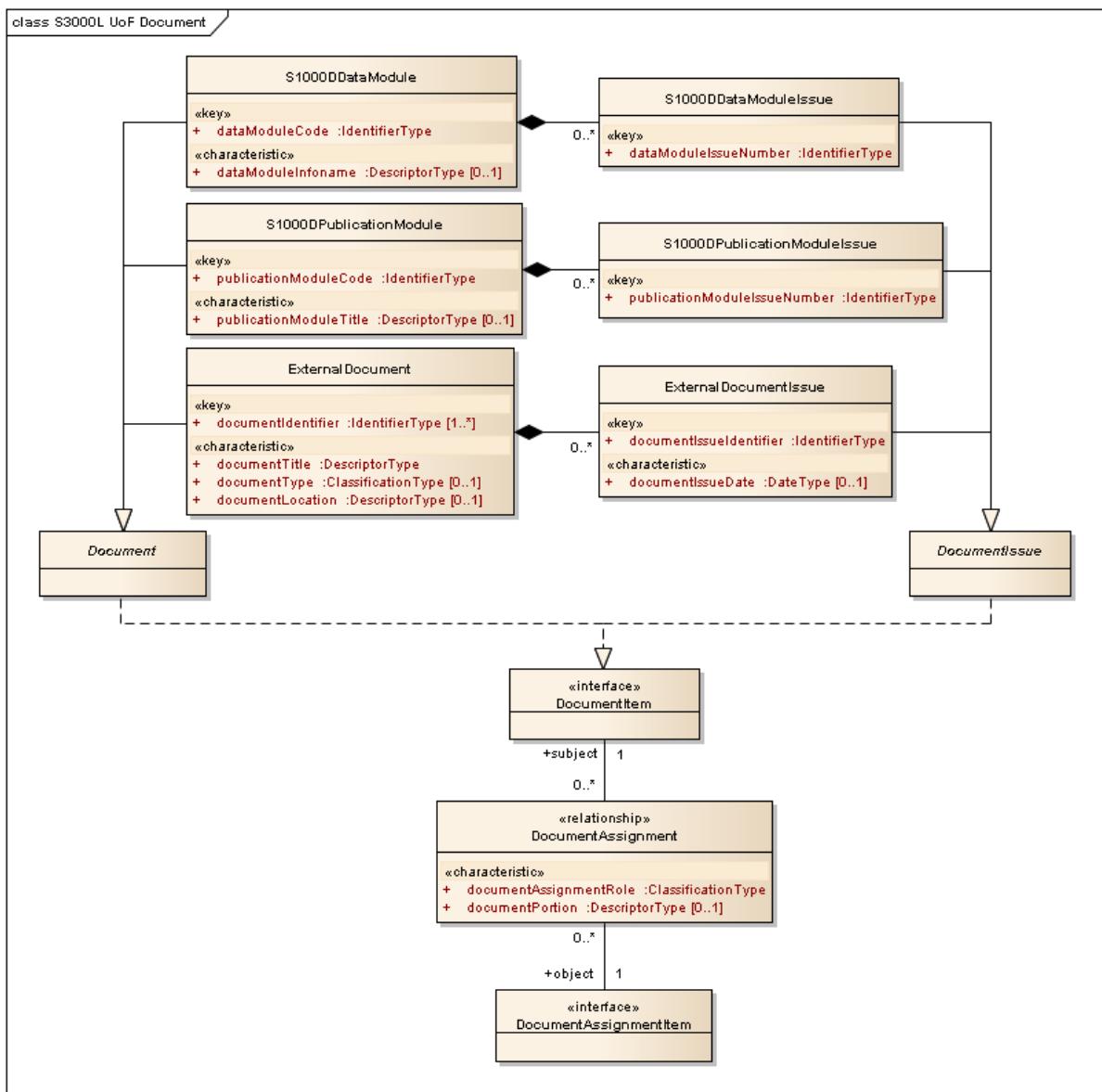
The S3000L data model defines three specializations of Document:

- S1000D data module
- S1000D publication module
- External document, which contains more meta information about the document itself

4.20.2 Graphical representation

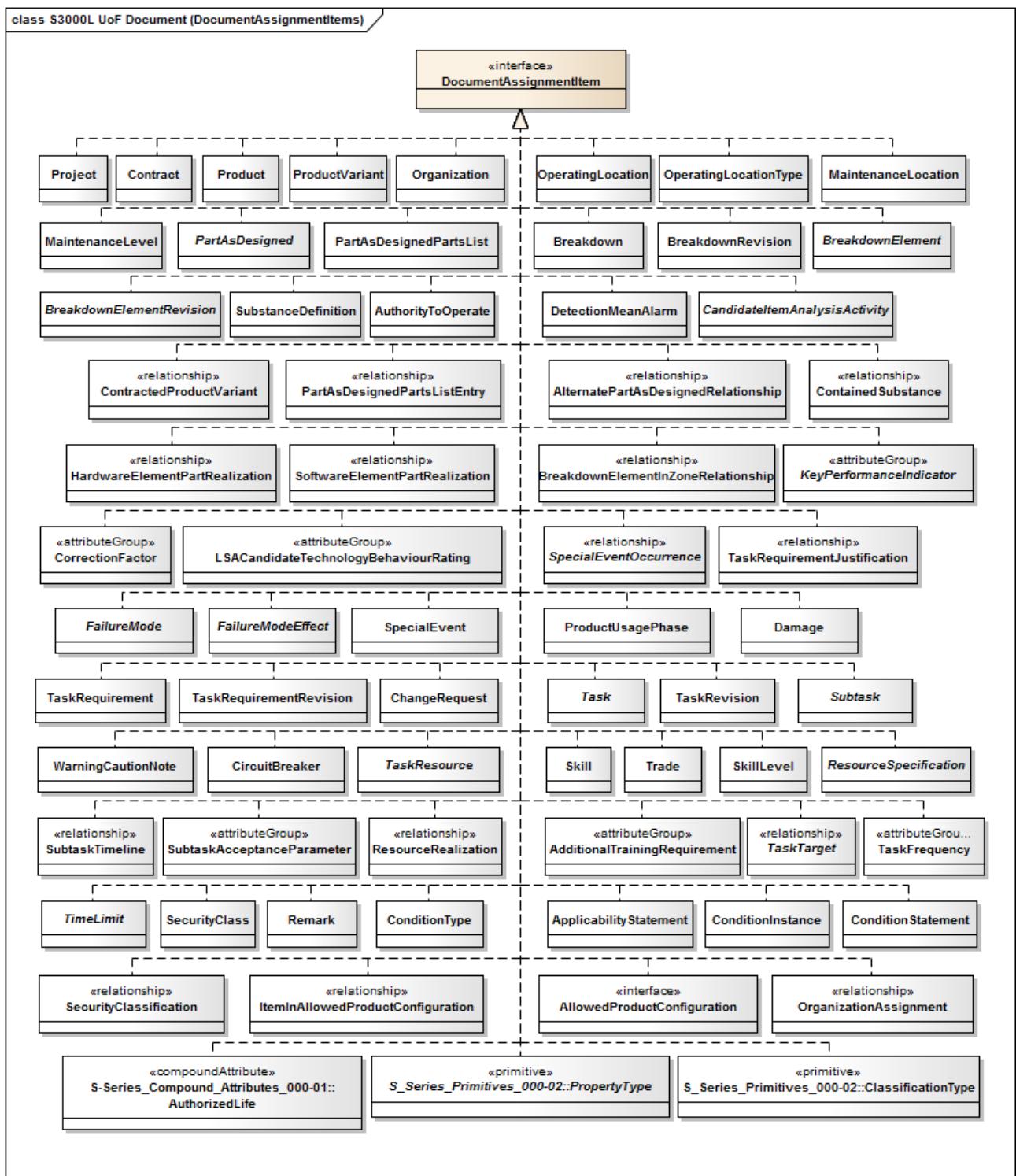
Note

The graphical representation of UoF Document is split into two illustrations. The first illustration illustrates classes that can be used to define documents and document issues. The second illustration illustrates the classes to which documents can be assigned.



ICN-B6865-S3000L0233-003-00

Fig 37 S3000L UoF Document - class model



ICN-B6865-S3000L0264-001-00

Fig 38 S3000L UoF Document - document assignment items

4.20.3 UoF Document - New class and interface definitions

4.20.3.1 Document

The Document class represents documents which are of relevance for the LSA program.

Applicable to: All

S3000L-A-19-00-0000-00A-040A-A

Chap 19

Note

Document is an abstract class, ie, an instantiation of Document must be either an S1000DDataModule, an S1000DPublicationModule, or an ExternalDocument.

Document implements the following <>interface>>:

- DocumentItem

Document associations:

- An optional association with zero, one or many instances of any class that implements the DocumentAssignmentItem <>interface>> (via the DocumentAssignment <>relationship>> class)

4.20.3.2 DocumentIssue

The DocumentIssue class represents a specific document issue which is of relevance for the LSA program.

Note

DocumentIssue is an abstract class, ie, an instantiation of DocumentIssue must be either an S1000DDataModuleIssue, an S1000DPublicationModuleIssue, or an ExternalDocumentIssue.

DocumentIssue implements the following <>interface>>:

- DocumentItem

DocumentIssue associations:

- An optional association with zero, one or many instances of any class that implements the DocumentAssignmentItem <>interface>> (via the DocumentAssignment <>relationship>> class)

4.20.3.3 S1000DDataModule

The S1000DDataModule class is a specialization of class Document and is used to represent documents written in accordance with an S1000D schema.

The S1000DDataModule class enables the cross references in between Tasks defined in S3000L and its corresponding S1000D data modules (UoF Task). Refer to [Para 4.14](#). However, an S1000DDataModule can also be assigned to zero, one or many instances of classes that implement the DocumentAssignmentItem <>interface>>, for the same reason as any other type of document.

S1000DDataModule attributes:

- dataModuleCode
- dataModuleInfoname (zero or one)

S1000DDataModule implements the following <>interface>>:

- DocumentItem (inherited from class Document)

S1000DDataModule association:

- An optional association with its S1000DDataModuleIssues
- An optional association with zero, one or many instances of any class that implements the DocumentAssignmentItem <>interface>> (inherited from class Document)

4.20.3.4 S1000DDataModuleIssue

The S1000DDataModuleIssue class is a specialization of the DocumentIssue class, and is used to represent a specific issue of a data module written in accordance with an S1000D schema.

The S1000DDataModuleIssue class enables the cross references in between Tasks defined in S3000L, and its corresponding S1000D data module issues (UoF Task). However, an S1000DDataModuleIssue can also be assigned to zero, one or many instances of classes that implement the DocumentAssignmentItem <>interface>, for the same reason as any other type of document.

S1000DDataModuleIssue attributes:

- dataModuleIssueNumber

S1000DDataModuleIssue implements the following <>interface>:

- DocumentItem (inherited from DocumentIssue)

S1000DDataModuleIssue associations:

- An association with the S1000DDataModule of which the S1000DDataModuleIssue is a revision
- An optional association with zero, one or many instances of any class that implements the DocumentAssignmentItem <>interface> (inherited from class Document)

4.20.3.5 S1000DPublicationModule

The S1000DPublicationModule class is a specialization of class Document, and is used to represent a publications published in accordance with S1000D.

The S1000DPublicationModule class enables references from eg, Subtasks to S1000D publication modules, as defined in UoF Task. Refer to [Para 4.14](#). However, a document that is of type S1000DPublicationModule can also be assigned to zero, one or many instances of classes that implement the DocumentAssignmentItem <>interface>, for the same reason as any other type of document.

S1000DPublicationModule attributes:

- publicationModuleCode
- publicationModuleTitle (zero or one)

S1000DPublicationModule implements the following <>interface>:

- DocumentItem (inherited from Document)

S1000DPublicationModule association:

- An optional association with its S1000DPublicationModuleIssues
- An optional association with zero, one or many instances of any class that implements the DocumentAssignmentItem <>interface> (inherited from Document)

4.20.3.6 S1000DPublicationModuleIssue

The S1000DPublicationModuleIssue class is a specialization of the DocumentIssue class, and is used to represent a specific issue of a publication module published in accordance with S1000D.

The S1000DPublicationModuleIssue class enables references from eg, Tasks or Subtasks to S1000D publication modules, as defined in UoF Task. Refer to [Para 4.14](#). However, a document that is of type S1000DPublicationModuleIssue can also be assigned to zero, one or many instances of classes that implement the DocumentAssignmentItem <>interface>, for the same reason as any other type of document.

S1000DPublicationModuleIssue attributes:

- publicationModuleIssueNumber

S1000DPublicationModuleIssue implements the following <>interface>:

- DocumentItem (inherited from DocumentIssue)

S1000D PublicationModuleIssue associations:

- An association with the S1000D PublicationModule of which the S1000D PublicationModuleIssue is a revision
- An optional association with zero, one or many instances of any class that implements the DocumentAssignmentItem <> (inherited from Document)

4.20.3.7 ExternalDocument

The ExternalDocument class is a specialization of class Document, and represents all documents that are not identified and produced in accordance with S1000D.

ExternalDocument attributes:

- documentIdentifier (one or many)
- documentTitle
- documentType (zero or one)
- documentLocation (zero or one)

ExternalDocument implements the following <>:

- DocumentItem (inherited from Document)

ExternalDocument associations:

- An optional association with its ExternalDocumentIssues
- An optional association with zero, one or many instances of any class that implements the DocumentAssignmentItem <> (inherited from Document)

4.20.3.8 ExternalDocumentIssue

The ExternalDocumentIssue class is a specialization of the DocumentIssue class, and represents specific issues of documents that are not identified and produced in accordance with S1000D.

ExternalDocumentIssue attributes:

- documentIssueIdentifier
- documentIssueDate (zero or one)

ExternalDocumentIssue implements the following <>:

- DocumentItem (inherited from DocumentIssue)

ExternalDocumentIssue associations:

- An association with the ExternalDocument of which the ExternalDocumentIssue is a revision
- An optional association with zero, one or many instances of any class that implements the DocumentAssignmentItem <> (inherited from Document)

4.20.3.9 DocumentAssignmentItem

The DocumentAssignmentItem <> is implemented by classes that can be associated with additional document information.

Classes and <> that implement the DocumentAssignmentItem <> are:

- UoF Product and Project
 - Project
 - Contract
 - Product

- ProductVariant
- ContractedProductVariant
- Organization
- UoF Product Usage Context
 - OperatingLocation
 - OperatingLocationType
 - MaintenanceLocation
 - MaintenanceLevel
- UoF Breakdown Structure
 - Breakdown
 - BreakdownRevision
 - BreakdownElement
 - BreakdownElementRevision
- UoF Part Definition
 - PartAsDesigned
 - PartAsDesignedPartsList
 - PartAsDesignedPartsListEntry
 - AlternatePartAsDesignedRelationship
 - SubstanceDefinition
 - ContainedSubstance
- UoF Breakdown Element Realization
 - HardwareElementPartRealization
 - SoftwareElementPartRealization
- UoF Breakdown Zone Element
 - BreakdownElementInZoneRelationship
- UoF Product Design Configuration
 - AllowedProductConfiguration
 - ItemInAllowedProductConfiguration
 - AuthorityToOperate
- UoF LSA Candidate
 - KeyPerformanceIndicator
 - CorrectionFactor
- UoF LSA Candidate Analysis Activity
 - CandidateItemAnalysisActivity
- UoF LSA FMEA
 - FailureMode
 - FailureModeEffect
 - DetectionMeanAlarm
- UoF Special Events and Damage
 - LSACandidateTechnologyBehaviourRating
 - SpecialEvent
 - ProductUsagePhase
 - SpecialEventOccurrence
 - Damage
- UoF LSA Candidate Task Requirement
 - TaskRequirement
 - TaskRequirementRevision
 - TaskRequirementJustification
 - ChangeRequest
- UoF Task
 - Task
 - TaskRevision
 - WarningCautionNote

- Subtask
- SubtaskTimeline
- SubtaskAcceptanceParameter
- CircuitBreaker
- UoF Task Resources
 - TaskResource
 - Skill
 - SkillLevel
 - Trade
 - AdditionalTrainingRequirement
 - ResourceSpecification
 - ResourceRealization
- UoF Task Usage (Part 1)
 - TaskTarget
 - TaskFrequency
 - TimeLimit
- UoF Security Classification
 - SecurityClass
 - SecurityClassification
- UoF Organization Assignment
 - OrganizationAssignment
- UoF Remark
 - Remark
- UoF Applicability Statement
 - ApplicabilityStatement
 - ConditionType
 - ConditionInstance
 - ConditionStatement
- S-Series Compound Attributes
 - AuthorizedLife
- S-Series Primitives
 - ClassificationType
 - PropertyType

Note

Documents can also be assigned to any class that is a subclass of the enumerated classes, eg, to a RectifyingTask which is a subclass of Task.

Classes that implement the DocumentAssignmentItem <> must implement the following association:

- An optional association with zero, one or many instances of Document or DocumentIssue (via the DocumentAssignment <> class)

4.20.3.10 DocumentItem

The DocumentItem <> is implemented by classes that represent documents.

Classes that implements the DocumentItem <> are:

- S1000DDataModule
- S1000DDataModuleIssue
- S1000DPublicationModule
- S1000DPublicationModuleIssue
- ExternalDocument
- ExternalDocumentIssue

Classes that implement the DocumentItem <<interface>> must implement the following association:

- An optional association with zero, one or many instances of classes that implements the DocumentAssignmentItem <<interface>> (via the DocumentAssignment <<relationship>> class)

4.20.3.11 DocumentAssignment

The DocumentAssignment <<relationship>> class defines a relationship between a specific Document or DocumentIssue and an instance of any class that implements the DocumentAssignmentItem <<interface>>.

DocumentAssignment attributes:

- documentAssignmentRole
- documentPortion (zero or one)

Note

The role of the document being assigned is determined by the documentAssignmentRole attribute.

Note

The documentPortion attribute identifies a defined portion of a document which is of interest in a specific usage.

DocumentAssignment associations:

- (subject) The specific Document or DocumentIssue being assigned to an object (instance of any class that implements the DocumentAssignmentItem <<interface>>)
- (object) The object to which the Document or DocumentIssue is assigned

Note

There is one instance of DocumentAssignment per relevant combination of Document or DocumentIssue and instance to which the Document or DocumentIssue is assigned.

DocumentAssignment special recommendations:

- Where an instance of DocumentAssignment is dependent upon eg, customer, it must be distinguished by the assignment of an ApplicabilityStatement to the DocumentAssignment instance. Refer to [Para 4.22](#).

4.20.4 UoF Document - Additions to referenced class and interface definitions

4.20.4.1 Project

The Project class defined in UoF Product and Project must also implement the following additional <<interface>>:

- DocumentAssignmentItem

4.20.4.2 ContractedProductVariant

The ContractedProductVariant <<relationship>> class defined in UoF Product and Project must also implement the following additional <<interface>>:

- DocumentAssignmentItem

4.20.4.3 Contract

The Contract class defined in UoF Product and Project must also implement the following additional <<interface>>:

- DocumentAssignmentItem

-
- 4.20.4.4 **Product**
The Product class defined in UoF Product and Project must also implement the following additional <>interface>>:
- DocumentAssignmentItem
- 4.20.4.5 **ProductVariant**
The ProductVariant class defined in UoF Product and Project must also implement the following additional <>interface>>:
- DocumentAssignmentItem
- 4.20.4.6 **Organization**
The Organization class defined in UoF Product and Project must also implement the following additional <>interface>>:
- DocumentAssignmentItem
- 4.20.4.7 **OperatingLocation**
The OperatingLocation class defined in UoF Product Usage Context must also implement the following additional <>interface>>:
- DocumentAssignmentItem
- 4.20.4.8 **MaintenanceLocation**
The MaintenanceLocation class defined in UoF Product Usage Context must also implement the following additional <>interface>>:
- DocumentAssignmentItem
- 4.20.4.9 **MaintenanceLevel**
The MaintenanceLevel class defined in UoF Product Usage Context must also implement the following additional <>interface>>:
- DocumentAssignmentItem
- 4.20.4.10 **OperatingLocationType**
The OperatingLocationType class defined in UoF Product Usage Context must also implement the following additional <>interface>>:
- DocumentAssignmentItem
- 4.20.4.11 **Breakdown**
The Breakdown class defined in UoF Breakdown Structure must also implement the following additional <>interface>>:
- DocumentAssignmentItem
- 4.20.4.12 **BreakdownRevision**
The BreakdownRevision class defined in UoF Breakdown Structure must also implement the following additional <>interface>>:
- DocumentAssignmentItem
- 4.20.4.13 **BreakdownElement**
The BreakdownElement class defined in UoF Breakdown Structure must also implement the following additional <>interface>>:
- DocumentAssignmentItem

4.20.4.14 BreakdownElementRevision

The BreakdownElementRevision class defined in UoF Breakdown Structure must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.15 PartAsDesigned

The PartAsDesigned class defined in UoF Part Definition must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.16 PartAsDesignedPartsList

The PartAsDesignedPartsList class defined in UoF Part Definition must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.17 PartAsDesignedPartsListEntry

The PartAsDesignedPartsListEntry <>relationship>> class defined in UoF Part Definition must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.18 AlternatePartAsDesignedRelationship

The AlternatePartAsDesignedRelationship <>relationship>> class defined in UoF Part Definition must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.19 SubstanceDefinition

The SubstanceDefinition class defined in UoF Part Definition must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.20 ContainedSubstance

The ContainedSubstance <>relationship>> class defined in UoF Part Definition must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.21 HardwareElementPartRealization

The HardwareElementPartRealization <>relationship>> class defined in UoF Breakdown Element Realization must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.22 SoftwareElementPartRealization

The SoftwareElementPartRealization <>relationship>> class defined in UoF Breakdown Element Realization must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.23 BreakdownElementInZoneRelationship

The BreakdownElementInZoneRelationship <>relationship>> class defined in UoF Breakdown Zone Element must also implement the following additional <>interface>>:

- DocumentAssignmentItem

-
- 4.20.4.24 AllowedProductConfiguration
Any class that implement the AllowedProductConfiguration <<interface>> defined in UoF Product Design Configuration must also implement the following additional <<interface>>:
- DocumentAssignmentItem
- 4.20.4.25 ItemInAllowedProductConfiguration
The ItemInAllowedProductConfiguration <<relationship>> class defined in UoF Product Design Configuration must also implement the following additional <<interface>>:
- DocumentAssignmentItem
- 4.20.4.26 AuthorityToOperate
The AuthorityToOperate class defined in UoF Product Design Configuration must also implement the following additional <<interface>>:
- DocumentAssignmentItem
- 4.20.4.27 KeyPerformanceIndicator
The KeyPerformanceIndicator <<attributeGroup>> defined in UoF LSA Candidate must also implement the following additional <<interface>>:
- DocumentAssignmentItem
- 4.20.4.28 CorrectionFactor
The CorrectionFactor <<attributeGroup>> defined in UoF LSA Candidate must also implement the following additional <<interface>>:
- DocumentAssignmentItem
- 4.20.4.29 CandidateItemAnalysisActivity
The CandidateItemAnalysisActivity class defined in UoF LSA Candidate Analysis Activity must also implement the following additional <<interface>>:
- DocumentAssignmentItem
- 4.20.4.30 FailureMode
The FailureMode class defined in UoF LSA FMEA must also implement the following additional <<interface>>:
- DocumentAssignmentItem
- 4.20.4.31 DetectionMeanAlarm
The DetectionMeanAlarm class defined in UoF LSA FMEA must also implement the following additional <<interface>>:
- DocumentAssignmentItem
- 4.20.4.32 FailureModeEffect
The FailureModeEffect class defined in UoF LSA FMEA must also implement the following additional <<interface>>:
- DocumentAssignmentItem
- 4.20.4.33 LSACandidateTechnologyBehaviourRating
The LSACandidateTechnologyBehaviourRating <<attributeGroup>> defined in UoF Special Events and Damage must also implement the following additional <<interface>>:
- DocumentAssignmentItem

4.20.4.34 SpecialEvent

The SpecialEvent class defined in UoF Special Events and Damage must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.35 ProductUsagePhase

The ProductUsagePhase class defined in UoF Special Events and Damage must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.36 SpecialEventOccurrence

The SpecialEventOccurrence <>relationship>> class defined in UoF Special Events and Damage must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.37 Damage

The Damage class defined in UoF Special Events and Damage must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.38 TaskRequirement

The TaskRequirement class defined in UoF LSA Candidate Task Requirement must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.39 TaskRequirementRevision

The TaskRequirementRevision class defined in UoF LSA Candidate Task Requirement must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.40 TaskRequirementJustification

The TaskRequirementJustification <>relationship>> class defined in UoF LSA Candidate Task Requirement must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.41 ChangeRequest

The ChangeRequest class defined in UoF LSA Candidate Task Requirement must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.42 Task

The Task class defined in UoF Task must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.43 TaskRevision

The TaskRevision class defined in UoF Task must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.44 WarningCautionNote

The WarningCautionNote class defined in UoF Task must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.45 Subtask

The Subtask class defined in UoF Task must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.46 SubtaskTimeline

The SubtaskTimeline <>relationship>>class defined in UoF Task must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.47 SubtaskAcceptanceParameter

The SubtaskAcceptanceParameter <>attributeGroup>> defined in UoF Task must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.48 CircuitBreaker

The CircuitBreaker class defined in UoF Task must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.49 TaskResource

The TaskResource class defined in UoF Task Resources must also implement the following additional <>interface>>:

- DocumentAssignmentItem.

4.20.4.50 Skill

The Skill class defined in UoF Task Resources must also implement the following additional <>interface>>:

- DocumentAssignmentItem.

4.20.4.51 SkillLevel

The SkillLevel class defined in UoF Task Resources must also implement the following additional <>interface>>:

- DocumentAssignmentItem.

4.20.4.52 Trade

The Trade class defined in UoF Task Resources must also implement the following additional <>interface>>:

- DocumentAssignmentItem.

4.20.4.53 AdditionalTrainingRequirement

The AdditionalTrainingRequirement <>attributeGroup>> defined in UoF Task Resources must also implement the following additional <>interface>>:

- DocumentAssignmentItem.

4.20.4.54 ResourceSpecification

The ResourceSpecification class defined in UoF Task Resources must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.55 ResourceRealization

The ResourceRealization <>relationship>> class defined in UoF Task Resources must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.56 TaskTarget

The TaskTarget <>relationship>> class defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.57 TaskFrequency

The TaskFrequency <>attributeGroup>> defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.58 TimeLimit

The TimeLimit class defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.59 SecurityClass

The SecurityClass class defined in UoF Security Classification must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.60 SecurityClassification

The SecurityClassification <>relationship>> class defined in UoF Security Classification must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.61 OrganizationAssignment

The OrganizationAssignment <>relationship>> class defined in UoF Organization Assignment must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.62 Remark

The Remark class defined in UoF Remark must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.63 ApplicabilityStatement

The ApplicabilityStatement class defined in UoF Applicability Statement must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.64 ConditionInstance

The ConditionInstance class defined in UoF Applicability Statement must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.65 ConditionType

The ConditionType class defined in UoF Applicability Statement must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.66 ConditionStatement

The ConditionStatement <>relationship>> class defined in UoF Applicability Statement must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.67 AuthorizedLife

The AuthorizedLife <>compoundAttribute>> defined in the S-Series Compound attributes must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.68 ClassificationType

The ClassificationType <>primitive>> defined in S-Series Primitives (Data Types) must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.20.4.69 PropertyType

The PropertyType <>primitive>> defined in S-Series Primitives (Data Types) must also implement the following additional <>interface>>:

- DocumentAssignmentItem

4.21 UoF Remark

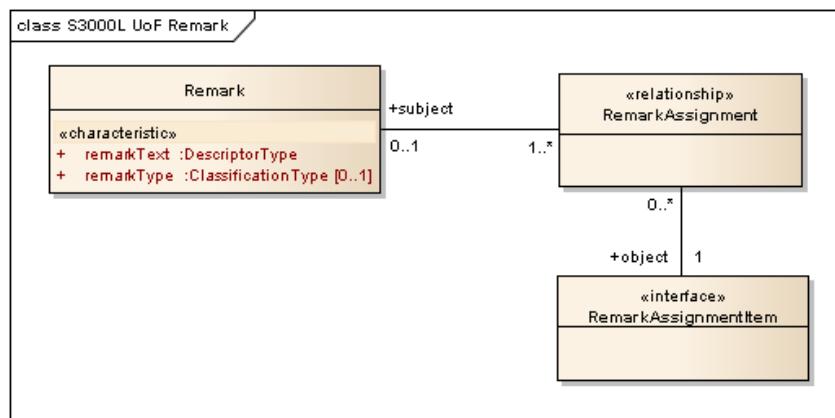
4.21.1 Overall description

The Remark UoF, [Fig 39](#) and [Fig 40](#), supports a flexible way of assigning remarks to instances of almost any class in the data model.

4.21.2 Graphical representation

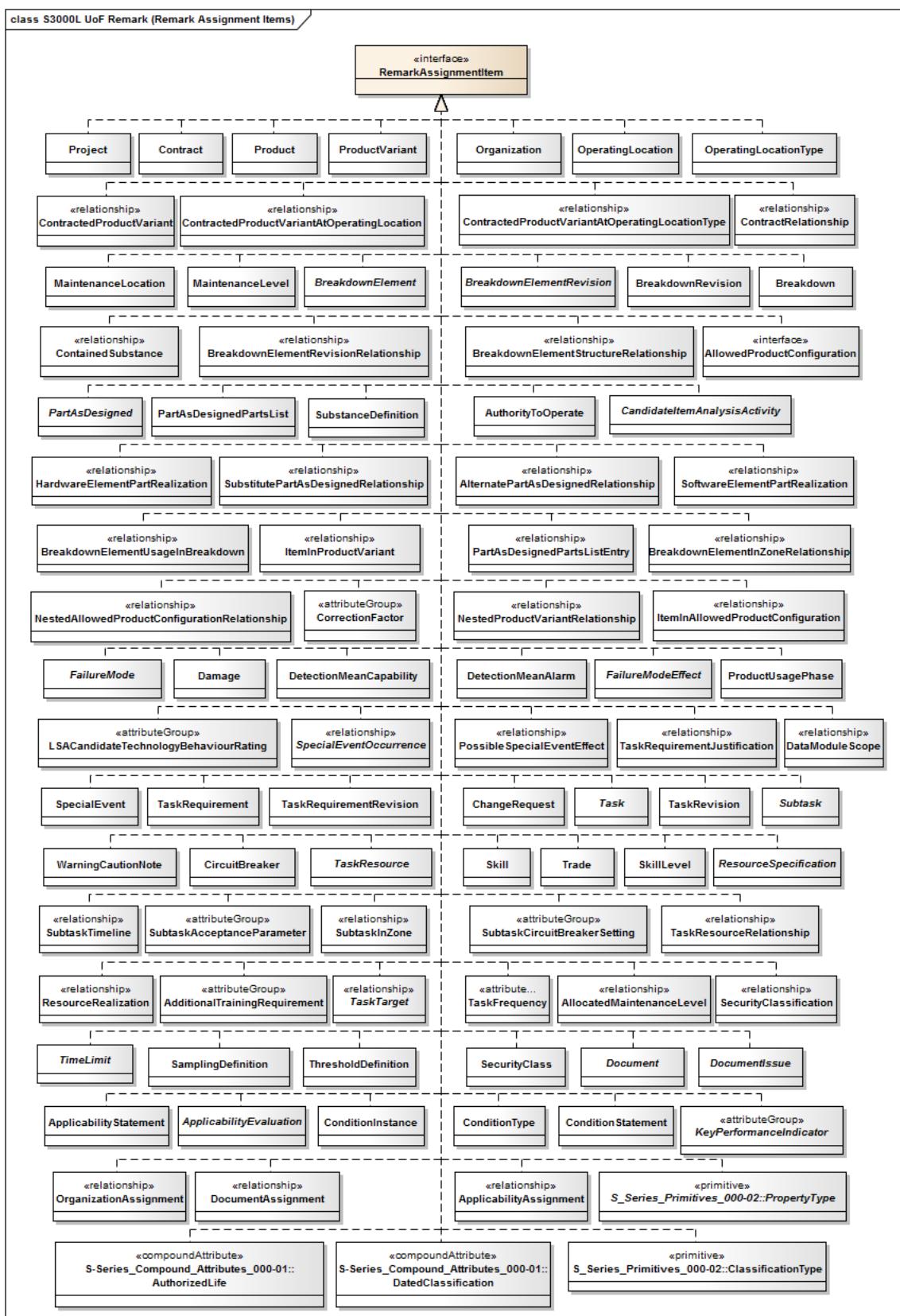
Note

The graphical representation of UoF Remark is split into two illustrations. The first illustration shows classes that can be used to define remarks. The second illustration shows the classes to which remarks can be assigned.



ICN-B6865-S3000L0234-003-00

Fig 39 S3000L UoF Remark - class model



ICN-B6865-S3000L0265-001-00

Fig 40 S3000L UoF Remark - remark assignment items

Applicable to: All

S3000L-A-19-00-0000-00A-040A-A
Chap 19

4.21.3 UoF Remark - New class and interface definitions

4.21.3.1 Remark

The Remark class represents remarks of any kind, which anyone working within the LSA program would like to put against an object of interest.

Remark attributes:

- remarkText
- remarkType (zero or one)

Note

Since the remarkText is of data type DescriptorType, all remarks can also identify the organization that provided the remark, and the date when it was recorded.

Remark associations:

- An association with one or many instances of any class that implements the RemarkAssignmentItem <> (via the RemarkAssignment <> class)

4.21.3.2 RemarkAssignmentItem

The RemarkAssignmentItem <> is implemented by all classes that can be associated with one or many remarks.

Classes and <> that implement the RemarkAssignmentItem <> are:

- UoF Product and Project
 - Project
 - Contract
 - ContractRelationship
 - Product
 - ProductVariant
 - ContractedProductVariant
 - Organization
- UoF Product Usage Context
 - ContractedProductVariantAtOperatingLocation
 - ContractedProductVariantAtOperatingLocationType
 - OperatingLocation
 - OperatingLocationType
 - MaintenanceLevel
 - MaintenanceLocation
- UoF Breakdown Structure
 - Breakdown
 - BreakdownRevision
 - BreakdownElement
 - BreakdownElementRevision
 - BreakdownElementRevisionRelationship
 - BreakdownElementUsageInBreakdown
 - BreakdownElementStructureRelationship
- UoF Part Definition
 - PartAsDesigned
 - PartAsDesignedPartsList
 - PartAsDesignedPartsListEntry
 - SubstanceDefinition
 - ContainedSubstance
 - AlternatePartAsDesignedRelationship
 - SubstitutePartAsDesignedRelationship
- UoF Breakdown Element Realization

- HardwareElementPartRealization
- SoftwareElementPartRealization
- UoF Breakdown Zone Element
 - BreakdownElementInZoneRelationship
- UoF Product Design Configuration
 - AllowedProductConfiguration
 - AuthorityToOperate
 - ItemInAllowedProductConfiguration
 - ItemInProductVariant
 - NestedProductVariantRelationship
 - NestedAllowedProductConfigurationRelationship
- UoF LSA Candidate
 - KeyPerformanceIndicator
 - CorrectionFactor
- UoF LSA Candidate Analysis Activity
 - CandidateItemAnalysisActivity
- UoF LSA FMEA
 - FailureMode
 - FailureModeEffect
 - DetectionMeanCapability
 - DetectionMeanAlarm
- UoF Special Events and Damage
 - ProductUsagePhase
 - SpecialEvent
 - SpecialEventOccurrence
 - Damage
 - LSACandidateTechnologyBehaviourRating
 - PossibleSpecialEventEffect
- UoF LSA Candidate Task Requirement
 - TaskRequirement
 - TaskRequirementRevision
 - TaskRequirementJustification
 - ChangeRequest
- UoF Task
 - Task
 - TaskRevision
 - DataModuleScope
 - WarningCautionNote
 - Subtask
 - SubtaskTimeline
 - SubtaskInZone
 - SubtaskAcceptanceParameter
 - SubtaskCircuitBreakerSetting
 - CircuitBreaker
- UoF Task Resources
 - TaskResource
 - TaskResourceRelationship
 - ResourceSpecification
 - ResourceRealization
 - AdditionalTrainingRequirement
 - Skill
 - Trade
 - SkillLevel

- UoF Task Usage (Part 1)
 - TaskTarget
 - TaskFrequency
 - AllocatedMaintenanceLevel
 - TimeLimit
 - SamplingDefinition
 - ThresholdDefinition
- UoF Security Classification
 - SecurityClass
 - SecurityClassification
- UoF Document
 - Document
 - DocumentIssue
 - DocumentAssignment
- UoF Organization Assignment
 - OrganizationAssignment
- UoF Applicability Statement
 - ApplicabilityStatement
 - ApplicabilityAssignment
 - ApplicabilityEvaluation
 - ConditionInstance
 - ConditionType
 - ConditionStatement
- S-Series Compound Attributes
 - AuthorizedLife
 - DatedClassification
- S-Series Primitives
 - ClassificationType
 - PropertyType

Note

Remarks can also be assigned to any class that is a subclass of the enumerated classes, eg, to a RectifyingTask which is a subclass of Task (the rule of substitutability).

Classes that implement the RemarkAssignmentItem <> must also implement the following association:

- An optional association with zero, one or many remarks (via the RemarkAssignment <> class)

4.21.3.3 RemarkAssignment

The RemarkAssignment <> class defines a relationship between a Remark and an instance of a class that implements the RemarkAssignmentItem <>.

RemarkAssignment associations:

- (subject) The specific Remark being associated with an object (instance of a class that implements the RemarkAssignmentItem <>)
- (object) The object for which the Remark is written

Note

There is one instance of RemarkAssignment per relevant combination of Remark and instance for which the Remark is written (ie, instance of a class that implements the RemarkAssignmentItem <>).

RemarkAssignment special recommendations:

- Where an instance of RemarkAssignment is dependent upon eg, external conditions or customer, it must be distinguished by the assignment of an ApplicabilityStatement to the RemarkAssignment instance. Refer to [Para 4.22](#).

4.21.4 UoF Remark - Additions to referenced class and interface definitions

4.21.4.1 Project

The Project class defined in UoF Product and Project must also implement the following additional <>:

- RemarkAssignmentItem

4.21.4.2 Contract

The Contract class defined in UoF Product and Project must also implement the following additional <>:

- RemarkAssignmentItem

4.21.4.3 ContractRelationship

The ContractRelationship <> class defined in UoF Product and Project must also implement the following additional <>:

- RemarkAssignmentItem

4.21.4.4 Product

The Product class defined in UoF Product and Project must also implement the following additional <>:

- RemarkAssignmentItem

4.21.4.5 ProductVariant

The ProductVariant class defined in UoF Product and Project must also implement the following additional <>:

- RemarkAssignmentItem

4.21.4.6 ContractedProductVariant

The ContractedProductVariant <> class defined in UoF Product and Project must also implement the following additional <>:

- RemarkAssignmentItem

4.21.4.7 Organization

The Organization class defined in UoF Product and Project must also implement the following additional <>:

- RemarkAssignmentItem

4.21.4.8 ContractedProductVariantAtOperatingLocation

The ContractedProductVariantAtOperatingLocation <> class defined in UoF Product Usage Context must also implement the following additional <>:

- RemarkAssignmentItem

4.21.4.9 ContractedProductVariantAtOperatingLocationType

The ContractedProductVariantAtOperatingLocationType <> class defined in UoF Product Usage Context must also implement the following additional <>:

- RemarkAssignmentItem

4.21.4.10 MaintenanceLevel

The MaintenanceLevel class defined in UoF Product Usage Context must also implement the following additional <>interface><:

- RemarkAssignmentItem

4.21.4.11 OperatingLocation

The OperatingLocation class defined in UoF Product Usage Context must also implement the following additional <>interface><:

- RemarkAssignmentItem

4.21.4.12 OperatingLocationType

The OperatingLocationType class defined in UoF Product Usage Context must also implement the following additional <>interface><:

- RemarkAssignmentItem

4.21.4.13 MaintenanceLocation

The MaintenanceLocation class defined in UoF Product Usage Context must also implement the following additional <>interface><:

- RemarkAssignmentItem

4.21.4.14 Breakdown

The Breakdown class defined in UoF Breakdown Structure must also implement the following additional <>interface><:

- RemarkAssignmentItem

4.21.4.15 BreakdownRevision

The BreakdownRevision class defined in UoF Breakdown Structure must also implement the following additional <>interface><:

- RemarkAssignmentItem

4.21.4.16 BreakdownElement

The BreakdownElement class defined in UoF Breakdown Structure must also implement the following additional <>interface><:

- RemarkAssignmentItem

4.21.4.17 BreakdownElementRevision

The BreakdownElementRevision class defined in UoF Breakdown Structure must also implement the following additional <>interface><:

- RemarkAssignmentItem

4.21.4.18 BreakdownElementRevisionRelationship

The BreakdownElementRevisionRelationship <>relationship>< class defined in UoF Breakdown Structure must also implement the following additional <>interface><:

- RemarkAssignmentItem

4.21.4.19 BreakdownElementUsageInBreakdown

The BreakdownElementUsageInBreakdown <>relationship>< class defined in UoF Breakdown Structure must also implement the following additional <>interface><:

- RemarkAssignmentItem

-
- 4.21.4.20 **BreakdownElementStructureRelationship**
The BreakdownElementStructureRelationship <> relationship class defined in UoF Breakdown Structure must also implement the following additional <> interface:
– RemarkAssignmentItem
- 4.21.4.21 **PartAsDesigned**
The PartAsDesigned class defined in UoF Part Definition must also implement the following additional <> interface:
– RemarkAssignmentItem
- 4.21.4.22 **PartAsDesignedPartsList**
The PartAsDesignedPartsList class defined in UoF Part Definition must also implement the following additional <> interface:
– RemarkAssignmentItem
- 4.21.4.23 **PartAsDesignedPartsListEntry**
The PartAsDesignedPartsListEntry <> relationship class defined in UoF Part Definition must also implement the following additional <> interface:
– RemarkAssignmentItem
- 4.21.4.24 **SubstanceDefinition**
The SubstanceDefinition class defined in UoF Part Definition must also implement the following additional <> interface:
– RemarkAssignmentItem
- 4.21.4.25 **ContainedSubstance**
The ContainedSubstance <> relationship class defined in UoF Part Definition must also implement the following additional <> interface:
– RemarkAssignmentItem
- 4.21.4.26 **AlternatePartAsDesignedRelationship**
The AlternatePartAsDesignedRelationship <> relationship class defined in UoF Part Definition must also implement the following additional <> interface:
– RemarkAssignmentItem
- 4.21.4.27 **SubstitutePartAsDesignedRelationship**
The SubstitutePartAsDesignedRelationship <> relationship class defined in UoF Part Definition must also implement the following additional <> interface:
– RemarkAssignmentItem
- 4.21.4.28 **HardwareElementPartRealization**
The HardwareElementPartRealization <> relationship class defined in UoF Breakdown Element Realization must also implement the following additional <> interface:
– RemarkAssignmentItem
- 4.21.4.29 **SoftwareElementPartRealization**
The SoftwareElementPartRealization <> relationship class defined in UoF Breakdown Element Realization must also implement the following additional <> interface:
– RemarkAssignmentItem

-
- 4.21.4.30 BreakdownElementInZoneRelationship
The BreakdownElementInZoneRelationship <> relationship class defined in UoF Breakdown Zone Element must also implement the following additional <> interface:
- RemarkAssignmentItem
- 4.21.4.31 AllowedProductConfiguration
Classes that implement the AllowedProductConfiguration <> interface defined in UoF Product Design Configuration must also implement the following additional <> interface:
- RemarkAssignmentItem
- 4.21.4.32 AuthorityToOperate
The AuthorityToOperate class defined in UoF Product Design Configuration must also implement the following additional <> interface:
- RemarkAssignmentItem
- 4.21.4.33 ItemInAllowedProductConfiguration
The ItemInAllowedProductConfiguration <> relationship class defined in UoF Product Design Configuration must also implement the following additional <> interface:
- RemarkAssignmentItem
- 4.21.4.34 ItemInProductVariant
The ItemInProductVariant <> relationship class defined in UoF Product Design Configuration must also implement the following additional <> interface:
- RemarkAssignmentItem
- 4.21.4.35 NestedProductVariantRelationship
The NestedProductVariantRelationship <> relationship class defined in UoF Product Design Configuration must also implement the following additional <> interface:
- RemarkAssignmentItem
- 4.21.4.36 NestedAllowedProductConfigurationRelationship
The NestedAllowedProductConfigurationRelationship <> relationship class defined in UoF Product Design Configuration must also implement the following additional <> interface:
- RemarkAssignmentItem
- 4.21.4.37 KeyPerformanceIndicator
The KeyPerformanceIndicator <> attribute group defined in UoF LSA Candidate must also implement the following additional <> interface:
- RemarkAssignmentItem
- 4.21.4.38 CorrectionFactor
The CorrectionFactor <> attribute group defined in UoF LSA Candidate must also implement the following additional <> interface:
- RemarkAssignmentItem
- 4.21.4.39 CandidateItemAnalysisActivity
The CandidateItemAnalysisActivity class defined in UoF LSA Candidate Analysis Activity must also implement the following additional <> interface:
- RemarkAssignmentItem

4.21.4.40 FailureMode

The FailureMode class defined in UoF LSA FMEA must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.41 FailureModeEffect

The FailureModeEffect class defined in UoF LSA FMEA must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.42 DetectionMeanCapability

The DetectionMeanCapability class defined in UoF LSA FMEA must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.43 DetectionMeanAlarm

The DetectionMeanAlarm class defined in UoF LSA FMEA must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.44 ProductUsagePhase

The ProductUsagePhase class defined in UoF Special Events and Damage must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.45 SpecialEvent

The SpecialEvent class defined in UoF Special Events and Damage must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.46 SpecialEventOccurrence

The SpecialEventOccurrence <>relationship>> class defined in UoF Special Events and Damage must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.47 Damage

The Damage class defined in UoF Special Events and Damage must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.48 LSACandidateTechnologyBehaviourRating

The LSACandidateTechnologyBehaviourRating <>attributeGroup>> defined in UoF Special Events and Damage must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.49 PossibleSpecialEventEffect

The PossibleSpecialEventEffect <>relationship>> class defined in UoF Special Events and Damage must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.50 TaskRequirement

The TaskRequirement class defined in UoF LSA Candidate Task Requirement must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.51 TaskRequirementRevision

The TaskRequirementRevision class defined in UoF LSA Candidate Task Requirement must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.52 TaskRequirementJustification

The TaskRequirementJustification <>relationship>> class defined in UoF LSA Candidate Task Requirement must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.53 ChangeRequest

The ChangeRequest class defined in UoF LSA Candidate Task Requirement must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.54 Task

The Task class defined in UoF Task must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.55 TaskRevision

The TaskRevision class defined in UoF Task must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.56 DataModuleScope

The DataModuleScope <>relationship>> class defined in UoF Task must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.57 WarningCautionNote

The WarningCautionNote class defined in UoF Task must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.58 Subtask

The Subtask class defined in UoF Task must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.59 SubtaskTimeline

The SubtaskTimeline <>relationship>> class defined in UoF Task must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.60 SubtaskInZone

The SubtaskInZone <<relationship>> class defined in UoF Task must also implement the following additional <<interface>>:

- RemarkAssignmentItem

4.21.4.61 SubtaskAcceptanceParameter

The SubtaskAcceptanceParameter <<attributeGroup>> defined in UoF Task must also implement the following additional <<interface>>:

- RemarkAssignmentItem

4.21.4.62 SubtaskCircuitBreakerSetting

The SubtaskCircuitBreakerSetting <<attributeGroup>> defined in UoF Task must also implement the following additional <<interface>>:

- RemarkAssignmentItem

4.21.4.63 CircuitBreaker

The CircuitBreaker class defined in UoF Task must also implement the following additional <<interface>>:

- RemarkAssignmentItem

4.21.4.64 TaskResource

The TaskResource class defined in UoF Task Resources must also implement the following additional <<interface>>:

- RemarkAssignmentItem

4.21.4.65 TaskResourceRelationship

The TaskResourceRelationship <<relationship>> class defined in UoF Task Resources must also implement the following additional <<interface>>:

- RemarkAssignmentItem

4.21.4.66 ResourceSpecification

The ResourceSpecification class defined in UoF Task Resources must also implement the following additional <<interface>>:

- RemarkAssignmentItem

4.21.4.67 ResourceRealization

The ResourceRealization <<relationship>> class defined in UoF Task Resources must also implement the following additional <<interface>>:

- RemarkAssignmentItem

4.21.4.68 AdditionalTrainingRequirement

The AdditionalTrainingRequirement <<attributeGroup>> defined in UoF Task Resources must also implement the following additional <<interface>>:

- RemarkAssignmentItem

4.21.4.69 Skill

The Skill class defined in UoF Task Resources must also implement the following additional <<interface>>:

- RemarkAssignmentItem

4.21.4.70 Trade

The Trade class defined in UoF Task Resources must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.71 SkillLevel

The SkillLevel class defined in UoF Task Resources must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.72 TaskTarget

The TaskTarget <>relationship>>class defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.73 TaskFrequency

The TaskFrequency <>attributeGroup>> defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.74 AllocatedMaintenanceLevel

The AllocatedMaintenanceLevel <>relationship>>class defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.75 TimeLimit

The TimeLimit class defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.76 SamplingDefinition

The SamplingDefinition class defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.77 ThresholdDefinition

The ThresholdDefinition class defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.78 SecurityClass

The SecurityClass class defined in UoF Security Classification must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.79 SecurityClassification

The SecurityClassification <>relationship>> class defined in UoF Security Classification must also implement the following additional <>interface>>:

- RemarkAssignmentItem

-
- 4.21.4.80 Document
The Document class defined in UoF Document must also implement the following additional <>interface>>:
– RemarkAssignmentItem
- 4.21.4.81 DocumentIssue
The DocumentIssue class defined in UoF Document must also implement the following additional <>interface>>:
– RemarkAssignmentItem
- 4.21.4.82 DocumentAssignment
The DocumentAssignment <>relationship>> class defined in UoF Document must also implement the following additional <>interface>>:
– RemarkAssignmentItem
- 4.21.4.83 OrganizationAssignment
The OrganizationAssignment <>relationship>> class defined in UoF Organization Assignment must also implement the following additional <>interface>>:
– RemarkAssignmentItem
- 4.21.4.84 ApplicabilityStatement
The ApplicabilityStatement class defined in UoF Applicability Statement must also implement the following additional <>interface>>:
– RemarkAssignmentItem
- 4.21.4.85 ApplicabilityAssignment
The ApplicabilityAssignment <>relationship>> class defined in UoF Applicability Statement must also implement the following additional <>interface>>:
– RemarkAssignmentItem
- 4.21.4.86 ApplicabilityEvaluation
The ApplicabilityEvaluation class defined in UoF Applicability Statement must also implement the following additional <>interface>>:
– RemarkAssignmentItem
- 4.21.4.87 ConditionInstance
The ConditionInstance class defined in UoF Applicability Statement must also implement the following additional <>interface>>:
– RemarkAssignmentItem
- 4.21.4.88 ConditionType
The ConditionType class defined in UoF Applicability Statement must also implement the following additional <>interface>>:
– RemarkAssignmentItem
- 4.21.4.89 ConditionStatement
The ConditionStatement <>relationship>> class defined in UoF Applicability Statement must also implement the following additional <>interface>>:
– RemarkAssignmentItem

4.21.4.90 AuthorizedLife

The AuthorizedLife <>compoundAttribute>> defined in the S-Series Compound attributes must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.91 DatedClassification

The DatedClassification <>compoundAttribute>> defined in the S-Series Compound attributes must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.92 ClassificationType

The ClassificationType <>primitive>> defined in S-Series Primitives (Data Types) must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.21.4.93 PropertyType

The PropertyType <>primitive>> defined in S-Series Primitives (Data Types) must also implement the following additional <>interface>>:

- RemarkAssignmentItem

4.22 UoF Applicability Statement

4.22.1 Overall description

The Applicability Statement UoF, [Fig 41](#) and [Fig 42](#), defines a flexible way of assigning applicability statements to instances of relevant classes in the data model. This UoF supports simple as well as highly complex definitions of applicability statements.

Note

The S3000L applicability statement class model is based on the applicability model defined in S1000D issue 4.0.

The Applicability Statement class model supports:

- Identification of parameters that can affect the applicability of LSA data (eg, parameters such as 'model A', 'tail number 15', 'arctic', 'desert', 'land-based', 'sea-based')
- Creation of computable logical expressions which can be evaluated, using the identified parameters
- Enumeration of the S3000L classes that can have an associated applicability statement

The definition of an applicability statement is based upon the following constructs:

- Applicability assignment, which identifies classes to which an applicability statement can be assigned
- Applicability evaluation, which defines the expression to be evaluated in order for the target item to be applicable

The applicability expression to be evaluated can be either a logical expression, or a value to be asserted. Any use of a logical expression will in turn relate to one or many values which must be asserted.

Note

S1000D defines two separate types of parameters which can affect the applicability of tech data. Firstly, product attributes and secondly, conditions. Product attributes are properties of the product that are typically set at the time of manufacture of a product instance and will not change throughout the service life of a product instance. Examples of product attributes given in S1000D are 'serial number' or 'model'. Conditions can be technical, operational,

environmental or any other type that can change throughout the service life of a product instance. Examples of conditions given in S1000D are location of maintenance, temperature, wind speed and sandy conditions. However, there is no hard and fast rule in S1000D for distinguishing between product attributes and conditions, which is why S3000L does not implement this distinction in the data model. Mapping in between S3000L parameters and S1000D product attributes and conditions needs to be done on a per project basis.

Note

Applicability statements in S1000D (<applic>) can be generated from Applicability statements defined in S3000L. Basic rules are:

- The S1000D <applic> element can be derived from the applicability statement paragraph of the S3000L data model, which includes ApplicabilityStatement, DatedApplicabilityStatement and the ApplicabilityAssignmentItem <>
- The content of the S1000D <evaluate> element within the <applic> element can be derived from the Applicability evaluation paragraph of the S3000L data model, which includes:
 - ApplicabilityEvaluationByLogicalOperator
 - LogicalOperator specializations AND, OR and NOT
- The content of the S1000D <assert> element within the <applic>/<evaluate> elements can be derived from:
 - ApplicabilityEvaluationByAssertionOfClassInstance together with instances of classes that implements the ApplicabilityAssertItem <>
 - ApplicabilityEvaluationByAssertionOfCondition together with ConditionStatement which relates to a defined ConditionDefinitionItem and a ConditionTypeAssertValue
 - ApplicabilityEvaluationByApplicabilityStatementReference which relates to a previously defined ApplicabilityStatement

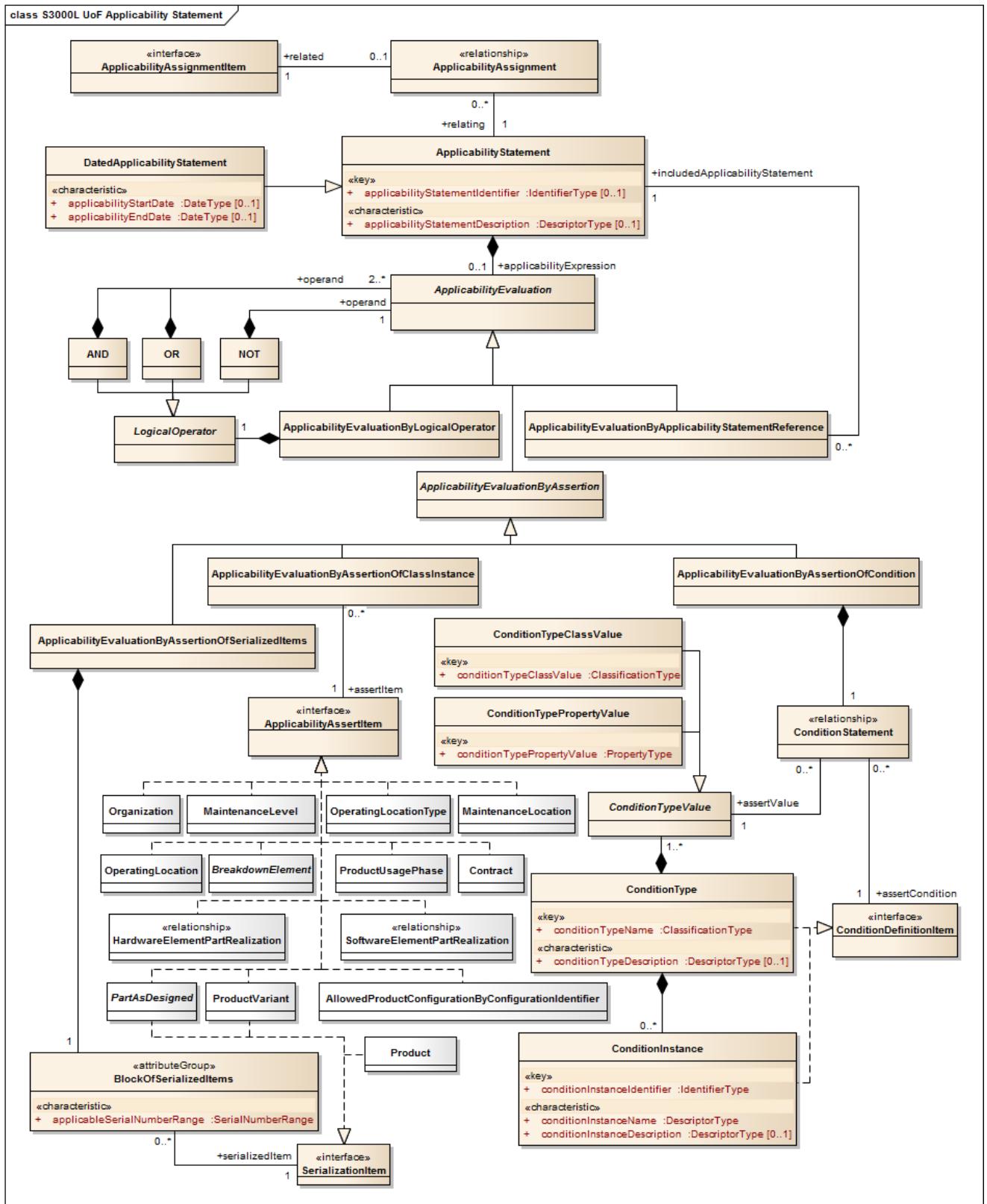
Note

ApplicabilityStatement is used to define usage related restrictions, whereas ItemInProductVariant and ItemInAllowedProductConfiguration (UoF Product Design Configuration) are used to define restrictions for a given product design (product definition). Refer to [Para 4.8](#).

4.22.2 Graphical representation

Note

The graphical representation of UoF Applicability Statement is split into two illustrations. The first illustration shows classes that can be used to define applicability statements. The second illustration shows the classes to which applicability statements can be assigned.



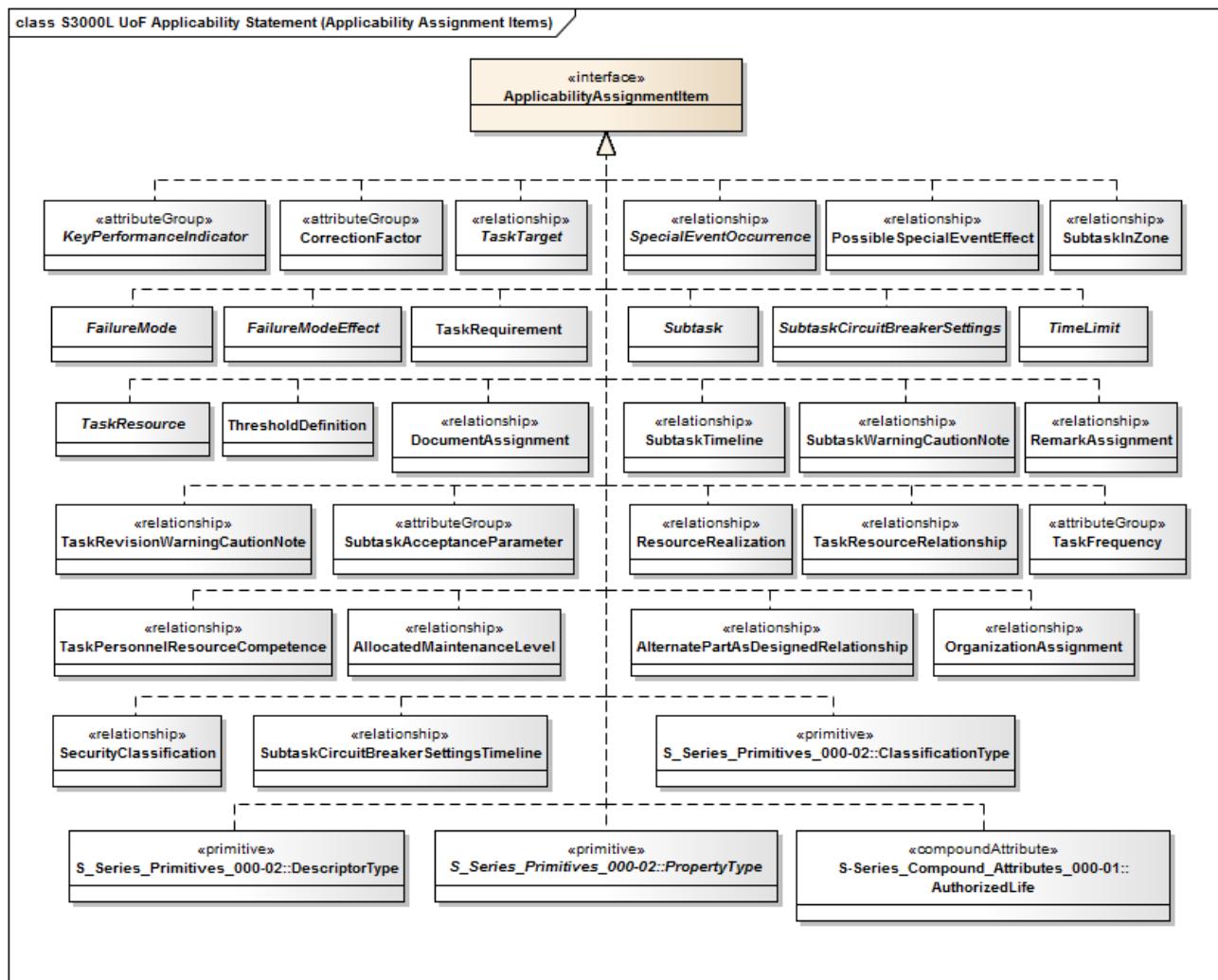
ICN-B6865-S3000L0235-003-00

Fig 41 S3000L UoF Applicability Statement - class model

Applicable to: All

S3000L-A-19-00-0000-00A-040A-A

Chap 19



ICN-B6865-S3000L0266-001-00

Fig 42 S3000L UoF Applicability Statement - applicability assignment items

4.22.3 UoF Applicability Statement - New class and interface definitions

4.22.3.1 ConditionType

The ConditionType class defines types of conditions which cannot be derived from any of the other classes that are defined within the S3000L data model.

Note

Examples of condition types that cannot be derived from any other class in the S3000L data model, and therefore needs to be defined as a ConditionType are:

- Service bulletin
- Operational environment
- Wind speed
- Ashore/Afloat
- Operational scenario
- Operational state

Note

An example of a class in the data model which can be used to assert an applicability expression is Customer. Individual customers (ie, instances of class Organization) can be

used in the ApplicabilityEvaluationByAssertionOfClassInstance. These values will therefore already exist in the LSA data and must not be defined as a ConditionType.

ConditionType attributes:

- conditionTypeName
- conditionTypeDescription (zero or one)

ConditionType class implements the following <>interface>>:

- ConditionDefinitionItem

ConditionType associations:

- An association with one or many instances of ConditionTypeValue, ie, values defined for the condition type
- An optional association with zero, one or many instances of ConditionInstance, where the ConditionInstance belongs to the ConditionType

Note

An example on ConditionType and ConditionInstance is 'Service Bulletin' (which is an instance of class ConditionType) and 'Service bulletin S001 - Chain guard' (which is an instance of ConditionInstance).

Note

The following rules can be applied to populate an S1000D Condition cross-reference table (CCT):

- conditionTypeName can populate the S1000D CCT <conditiontype id> attribute and <conditiontype><name> element, respectively
- conditionTypeDescription can populate the S1000D CCT <conditiontype><description> element

4.22.3.2 ConditionTypeValue

The ConditionTypeValue class defines valid values for a specific ConditionType.

Note

The ConditionTypeValue class is an abstract class, ie, any of its subclasses must be used for the actual representation of the expression to be evaluated or asserted.

- ConditionTypeValue subclasses:
 - ConditionTypeClassValue
 - ConditionTypePropertyValue

Note

Instances of the ConditionTypeValue have no meaning by itself, but only in the context of a specific ConditionType.

ConditionTypeValue associations:

- An association with ConditionType for which the ConditionTypeValue is defined
- An optional association with one or many ConditionStatements in which the ConditionTypeValue is used

4.22.3.3 ConditionTypeClassValue

The ConditionTypeClassValue class is a specialization of class ConditionTypeValue and represents condition values which can be enumerated (instances of ClassificationType).

Note

Examples on valid values (instances of ClassificationType) that can be used are:

- Pre/Post defined as valid values for eg, the Service bulletin ConditionType
- Deser'/Arctic defined as valid values for an Operational environment ConditionType

Note

Instances of the ConditionTypeClassValue have no meaning by itself, but only in the context of a specific ConditionType.

ConditionTypeClassValue attributes:

- conditionTypeClassValue

ConditionTypeClassValue associations:

- An association with ConditionType for which the ConditionTypeClassValue is defined (inherited from class ConditionTypeValue)
- An optional association with one or many ConditionStatements in which the ConditionTypeClassValue is used (inherited from class ConditionTypeValue)

Note

The following rules can be applied to populate an S1000D CCT:

- conditionTypeClassValue can populate the S1000D CCT <conditiontype><enum> element for the associated ConditionType

4.22.3.4 ConditionTypePropertyValue

The ConditionTypePropertyValue class is a specialization of class ConditionTypeValue and represents, eg, valid values and value ranges, which can be used for a specific ConditionType.

Note

Valid values can be of any kind of PropertyType, ie, it can be a SingleValuePropertyType, ValueWithTolerancesPropertyType, or a ValueRangePropertyType.

Note

Examples on valid property values are '0-15', '15-30' and '30-45' defined for ConditionType 'Wind speed'.

Note

Instances of the ConditionTypePropertyValue have no meaning by itself, but only in the context of a specific ConditionType.

ConditionTypePropertyValue attributes:

- conditionTypePropertyValue

ConditionTypePropertyValue associations:

- An association with ConditionType for which the ConditionTypePropertyValue is defined (inherited from class ConditionTypeValue)
- An optional association with one or many ConditionStatements in which the ConditionTypePropertyValue is used (inherited from class ConditionTypeValue)

Note

The following rules can be applied to populate an S1000D CCT:

- conditionTypePropertyValue can populate the S1000D CCT <conditiontype><enum> element for the associated ConditionType

4.22.3.5 ConditionDefinitionItem

The ConditionDefinitionItem <> is implemented by classes that can be asserted in order to determine the applicability of the data being defined during LSA.

Classes that implements the ConditionDefinitionItem <> are:

- ConditionType
- ConditionInstance

Classes that implement the ConditionDefinitionItem <> must implement the following associations:

- An optional association with one or many instances of ConditionStatement in which the ConditionDefinitionItem is referenced

4.22.3.6 ConditionInstance

The ConditionInstance class identifies instances that belongs to a specific ConditionType, and can be used in the definition of one or more applicability statements.

Note:

An example of a ConditionInstance is:

- ‘Service bulletin S001 - Chain guard’ (an identified instance that belongs to the ‘Service bulletin’ ConditionType)

ConditionInstance attributes:

- conditionInstanceIdentifier
- conditionInstanceName
- conditionInstanceDescription (zero or one)

ConditionInstance implements the following <>:

- ConditionDefinitionItem

ConditionInstance associations:

- An association with the ConditionType to which the ConditionInstance belongs.

Note

The following rules can be applied to populate an S1000D CCT:

- conditionInstanceIdentifier can populate the S1000D CCT <condition id> attribute
- conditionInstanceDescription can populate the S1000D CCT <condition><description> element
- conditionInstanceName can populate the S1000D CCT <condition><name> and <condition><displayname> elements, respectively

4.22.3.7 ApplicabilityStatement

The ApplicabilityStatement class defines explicit applicability statements to be applied against instances in a LSA database.

Note

Applicability statements can be evaluated in a maintenance execution environment, eg, using an Interactive Electronic Technical Publishing (IETP).

ApplicabilityStatement attributes:

- applicabilityStatementIdentifier (zero or one)
- applicabilityStatementDescription (zero or one)

ApplicabilityStatement associations:

- An optional association with zero or one computer interpretable expression, ie, an instance of ApplicabilityEvaluation
- An association with one or many objects to which the ApplicabilityStatement is applied, ie, instances of any class that implements the ApplicabilityAssignmentItem <> (via the ApplicabilityAssignment <> class)

- An optional association with zero, one or many instances of ApplicabilityEvaluationByApplicabilityStatementReference, in which the ApplicabilityStatement under consideration can be included as part of the evaluation of another ApplicabilityStatement

Note

ApplicabilityStatement can be specialized into a DatedApplicabilityStatement.

4.22.3.8 DatedApplicabilityStatement

The DatedApplicabilityStatement class is a specialization of class ApplicabilityStatement, and defines applicability statements where the applicability is not just defined by the expression to be evaluated, but is also limited in time.

DatedApplicabilityStatement attributes:

- applicabilityStatementIdentifier (inherited from class ApplicabilityStatement)
- applicabilityStatementDescription (inherited from class ApplicabilityStatement)
- applicabilityStartDate (zero or one)
- applicabilityEndDate (zero or one)

DatedApplicabilityStatement associations:

- An optional association with zero or one computer interpretable expression, ie, an instance of ApplicabilityEvaluation (inherited from class ApplicabilityStatement)
- An association with one or many objects to which the ApplicabilityStatement is applied, ie, instances of any class that implements the ApplicabilityAssignmentItem <> (inherited from class ApplicabilityStatement)
- An optional association with zero, one or many instances of ApplicabilityEvaluationByApplicabilityStatementReference, in which the ApplicabilityStatement under consideration can be included as part of the evaluation of another ApplicabilityStatement (inherited from class ApplicabilityStatement)

4.22.3.9 ApplicabilityAssignment

The ApplicabilityAssignment <> class defines a relationship between a specific ApplicabilityStatement and an instance of any class that implements the ApplicabilityAssignmentItem <>, ie, instance to which the ApplicabilityStatement applies.

ApplicabilityAssignment associations:

- (relating) The ApplicabilityStatement being applied to an object (instance of any class that implements the ApplicabilityAssignmentItem <>)
- (related) The object to which the ApplicabilityStatement is applied

Note

There is one instance of ApplicabilityAssignment per relevant combination of ApplicabilityStatement and instance to which the ApplicabilityStatement is being applied (ie, instance of any class that implements the ApplicabilityAssignmentItem <>).

4.22.3.10 ApplicabilityAssignmentItem

The ApplicabilityAssignmentItem <> is implemented by classes that can be associated with applicability statements.

Classes that implements the ApplicabilityAssignmentItem <> are:

- UoF Part Definition
 - AlternatePartAsDesignedRelationship
- UoF LSA Candidate
 - KeyPerformanceIndicator

- CorrectionFactor
- UoF LSA FMEA
 - FailureMode
 - FailureModeEffect
 - SpecialEventOccurrence
 - PossibleSpecialEventEffect
- UoF LSA Candidate Task Requirement
 - TaskRequirement
- UoF Task
 - Subtask
 - SubtaskTimeline
 - SubtaskInZone
 - TaskRevisionWarningCautionNote
 - SubtaskWarningCautionNote
 - SubtaskAcceptanceParameter
 - SubtaskCircuitBreakerSettings
 - SubtaskCircuitBreakerSettingsTimeline
- UoF Task Resources
 - TaskResource
 - TaskResourceRelationship
 - ResourceRealization
 - TaskPersonnelResourceCompetence
- UoF Task Usage (Part 1)
 - TaskTarget
 - TaskFrequency
 - AllocatedMaintenanceLevel
 - TimeLimit
 - ThresholdDefinition
- UoF Security Classification
 - SecurityClassification
- UoF Organization Assignment
 - OrganizationAssignment
- UoF Document
 - DocumentAssignment
- UoF Remark
 - RemarkAssignment
- S-Series Compound attributes
 - AuthorizedLife
- S-Series Primitives (Data Types)
 - ClassificationType
 - DescriptorType
 - PropertyType

Note

ApplicabilityStatements can also be assigned to any class that is a subclass of the enumerated classes, eg, RectifyingTask which is a subclass of Task (rule of substitutability).

Classes that implement the ApplicabilityAssignmentItem <> must also implement the following association:

- An optional association with one instance of ApplicabilityStatement per instance of the respective class

4.22.3.11 ApplicabilityEvaluation

The ApplicabilityEvaluation class identifies an expression against which the applicability statement can be asserted.

The expression to be evaluated is either a logical expression, or a value to be asserted. These are defined as separate subclasses of ApplicabilityEvaluation.

Note

ApplicabilityEvaluation is an abstract class, ie, an instantiation of ApplicabilityEvaluation must always be done using any of its specializations:

- ApplicabilityEvaluationByLogicalOperator
- ApplicabilityEvaluationByAssertionOfCondition
- ApplicabilityEvaluationByAssertionOfClassInstance
- ApplicabilityEvaluationByAssertionOfSerializedItems
- ApplicabilityEvaluationByApplicabilityStatementReference

ApplicabilityEvaluation associations:

- An instance of ApplicabilityEvaluation is always associated with a specific instance of ApplicabilityStatement, either directly or as a logical operator operand (ie, as part of an AND, OR or NOT expression)

4.22.3.12 ApplicabilityEvaluationByLogicalOperator

The ApplicabilityEvaluationByLogicalOperator class is a specialization of class ApplicabilityEvaluation, and defines a logical expression against which an applicability statement can be evaluated.

Note

An instance of LogicalOperator is always defined for a specific instance of ApplicabilityEvaluationByLogicalOperator. An instance of LogicalOperator does always refer to one or many other instances of ApplicabilityEvaluation, which in turn can be either a logical expression or a value to be asserted.

Note

An instance of ApplicabilityEvaluationByLogicalOperator does not have any meaning outside the context of the ApplicabilityStatement or LogicalOperator instance, in which it is being defined.

ApplicabilityEvaluationByLogicalOperator associations:

- An instance of ApplicabilityEvaluationByLogicalOperator is always associated with a specific ApplicabilityStatement, either directly or indirectly as a logical operator operand ie, as part of an AND, OR or NOT expression (inherited from class ApplicabilityEvaluation)
- An instance of ApplicabilityEvaluationByLogicalOperator must always be associated with an instance of a LogicalOperator which in turn refers to other instances of ApplicabilityEvaluation to be evaluated or asserted

4.22.3.13 LogicalOperator

The LogicalOperator class defines the logical expression to be evaluated.

Note

LogicalOperator is an abstract class, ie, an instantiation of LogicalOperator must always be done using any of its specializations AND, OR or NOT.

LogicalOperator associations:

- An instance of LogicalOperator is always associated with a specific instance of ApplicabilityEvaluationByLogicalOperator

4.22.3.14 AND

The AND class is a specialization of class LogicalOperator, and defines a list of ApplicabilityEvaluations that all need to be TRUE in order for the overall ApplicabilityStatement to be TRUE.

Note

S1000D uses AND statements in the definition of an <applic> element.

AND associations:

- An instance of AND is always associated with a specific instance of ApplicabilityEvaluationByLogicalOperator (inherited from class LogicalOperator)
- An instance of AND relates to two or more instances of ApplicabilityEvaluation as its operands

4.22.3.15 OR

The OR class is a specialization of class LogicalOperator, and defines a list of ApplicabilityEvaluations where at least one needs to be TRUE in order for the overall ApplicabilityStatement to be TRUE.

Note

S1000D does not use OR statements in the definition of an <applic> element.

OR associations:

- An instance of OR is always associated with a specific instance of ApplicabilityEvaluationByLogicalOperator (inherited from class LogicalOperator)
- An instance of OR relates to two or more instances of ApplicabilityEvaluation as its operands

4.22.3.16 NOT

The NOT class is a specialization of class LogicalOperator, and defines that the related ApplicabilityEvaluation needs to be FALSE in order for the overall ApplicabilityStatement to be TRUE.

Note

S1000D does not make explicit use of NOT statements in the definition of an <applic> element.

NOT associations:

- An instance of NOT is always associated with a specific instance of ApplicabilityEvaluationByLogicalOperator (inherited from class LogicalOperator)
- An instance of NOT relates to one instance of ApplicabilityEvaluation as its operand

4.22.3.17 ApplicabilityEvaluationByApplicabilityStatementReference

The ApplicabilityEvaluationByApplicabilityStatementReference class is a specialization of class ApplicabilityEvaluation, and makes a reference another ApplicabilityStatement that must be asserted as part of a parent ApplicabilityEvaluation expression. This class enables the definition of nested applicability statements.

Note

An instance of ApplicabilityEvaluationByApplicabilityStatementReference does not have any meaning outside the context of the ApplicabilityStatement or LogicalOperator, in which it is being defined.

ApplicabilityEvaluationByApplicabilityStatementReference associations:

- An instance of ApplicabilityEvaluationByApplicabilityStatementReference is always associated with a specific ApplicabilityStatement, either directly or indirectly as a logical operator operand, ie, as part of an AND, OR or NOT expression (inherited from class ApplicabilityEvaluation)
- An instance of ApplicabilityEvaluationByApplicabilityStatementReference is always associated with an instance of ApplicabilityStatement which must be asserted as part of the parent ApplicabilityStatement

4.22.3.18 ApplicabilityEvaluationByAssertion

The ApplicabilityEvaluationByAssertion class is a specialization of class ApplicabilityEvaluation, and identifies a value against which an applicability statement must be asserted.

Note

ApplicabilityEvaluationByAssertion is an abstract class, ie, an instantiation of ApplicabilityEvaluationByAssertion must always be done using any of its specializations.

ApplicabilityEvaluationByAssertion subclasses:

- ApplicabilityEvaluationByAssertionOfCondition
- ApplicabilityEvaluationByAssertionOfClassInstance
- ApplicabilityEvaluationByAssertionOfSerializedItems

ApplicabilityEvaluationByAssertion associations:

- An instance of ApplicabilityEvaluationByAssertion is always associated with a specific ApplicabilityStatement, either directly or indirectly as a logical operator operand ie, as part of an AND, OR or NOT expression (inherited from class ApplicabilityEvaluation)

4.22.3.19 ApplicabilityEvaluationByAssertionOfCondition

The ApplicabilityEvaluationByAssertionOfCondition class is a specialization of class ApplicabilityEvaluationByAssertion, and identifies a condition statement against which the applicability statement must be asserted.

Note

An instance of ApplicabilityEvaluationByAssertionOfCondition does not have any meaning outside the context of the ApplicabilityStatement or LogicalOperator in which it is being defined.

ApplicabilityEvaluationByAssertionOfCondition associations:

- An instance of ApplicabilityEvaluationByAssertionOfCondition is always associated with a specific instance of ApplicabilityStatement, either directly or indirectly as a logical operator operand ie, as part of an AND, OR or NOT expression (inherited from class ApplicabilityEvaluation)
- An instance of ApplicabilityEvaluationByAssertionOfCondition always refers to a ConditionStatement against which the applicability statement must be asserted

4.22.3.20 ConditionStatement

The ConditionStatement <> relationship class firstly defines a reference to the related condition definition which must be asserted, and secondly a reference to the value which needs to be true in order for the ApplicabilityEvaluationByAssertionOfCondition expression to be true.

Note

An instance of ConditionStatement does not have any meaning outside the context of the ApplicabilityEvaluationByAssertionOfCondition instance in which it is being defined.

ConditionStatement association:

- An instance of ConditionStatement is always related to a specific instance of ApplicabilityEvaluationByAssertionOfCondition
- An association with an instance of either ConditionType or ConditionInstance for which a value must be asserted (via the ConditionDefinitionItem <>interface>>)
- An association with an instance of ConditionTypeValue that represents the value that must be true, in order for the parent ApplicabilityEvaluationByAssertionOfCondition to be true

4.22.3.21 ApplicabilityEvaluationByAssertionOfClassInstance

The ApplicabilityEvaluationByAssertionOfClassInstance class is a specialization of class ApplicabilityEvaluationByAssertion and identifies a class instance against which an applicability statement must be asserted.

Note

An instance of ApplicabilityEvaluationByAssertionOfClassInstance does not have any meaning outside the context of the ApplicabilityStatement or LogicalOperator in which it is being defined.

ApplicabilityEvaluationByAssertionOfClassInstance associations:

- An instance of ApplicabilityEvaluationByAssertionOfClassInstance is always associated with a specific instance of ApplicabilityStatement, either directly or indirectly as a logical operator operand ie, as part of an AND, OR or NOT expression (inherited from class ApplicabilityEvaluation)
- An instance of ApplicabilityEvaluationByAssertionOfClassInstance is always associated with a class instance which represents the value that must be true, in order for the parent ApplicabilityEvaluationByAssertionOfClassInstance to be true (via the ApplicabilityAssertItem <>interface>>)

4.22.3.22 ApplicabilityAssertItem

The ApplicabilityAssertItem <>interface>> is implemented by classes that have instances (values) that can be used for defining applicability statements.

Classes and interfaces that implements the ApplicabilityAssertItem <>interface>> are:

- UoF Product and Project
 - Organization
 - Contract
 - ProductVariant
- UoF Product Usage Context
 - MaintenanceLevel
 - MaintenanceLocation
 - OperatingLocationType
 - OperatingLocation
- UoF Breakdown Structure
 - BreakdownElement
- UoF Part Definition
 - PartAsDesigned
- UoF Breakdown Element Realization
 - HardwareElementPartRealization
 - SoftwareElementPartRealization
- UoF Product Design Configuration
 - AllowedProductConfigurationByConfigurationIdentifier
- UoF Special Events and Damage
 - ProductUsagePhase

Classes that implement the ApplicabilityAssertItem <>interface>> must also implement the following association:

- An optional association with zero, one or many instances of ApplicabilityEvaluationByAssertionOfClassInstance

4.22.3.23 ApplicabilityEvaluationByAssertionOfSerializedItems

The ApplicabilityEvaluationByAssertionOfSerializedItems class is a specialization of class ApplicabilityEvaluationByAssertion, and identifies a range of serialized items against which the applicability statement must be asserted.

Note

An instance of ApplicabilityEvaluationByAssertionOfSerializedItems does not have any meaning outside the context of the ApplicabilityStatement or LogicalOperator in which it is being defined.

ApplicabilityEvaluationByAssertionOfSerializedItems associations:

- An instance of ApplicabilityEvaluationByAssertionOfSerializedItems is always associated with a specific instance of ApplicabilityStatement, either directly or indirectly as a logical operator operand ie, as part of an AND, OR or NOT expression (inherited from class ApplicabilityEvaluation)
- An association with a BlockOfSerializedItems against which the applicability statement must be asserted

4.22.3.24 BlockOfSerializedItems

The BlockOfSerializedItems <> defines a possibly open-ended serial number range of end items or specific hardware parts.

BlockOfSerializedItems attributes:

- applicableSerialNumberRange

BlockOfSerializedItems association:

- An instance of BlockOfSerializedItems is always related to a specific instance of ApplicabilityEvaluationByAssertionOfSerializedItems
- An association with the item for which the serial number range is defined (via the SerializationItem <>)

4.22.3.25 SerializationItem

The SerializationItem <> is implemented by classes that can be serialized.

Classes that implement the SerializationItem <> are:

- Product
- ProductVariant
- PartAsDesigned

Classes that implement the SerializationItem <> must also implement the following association:

- An optional association with zero, one or many instances of BlockOfSerializedItems identifying individual ranges of serialized items

4.22.4 UoF Applicability Statement - Additions to referenced class and interface definitions

4.22.4.1 Organization

The Organization class defined in UoF Product and Project must also implement the following <>:

- ApplicabilityAssertItem

4.22.4.2 Contract

The Contract class defined in UoF Product and Project must also implement the following additional <>interface><:

- ApplicabilityAssertItem

4.22.4.3 Product

The Product class defined in UoF Product and Project must also implement the following additional <>interface><:

- SerializationItem

4.22.4.4 ProductVariant

The ProductVariant class defined in UoF Product and Project must also implement the following additional <>interfaces><:

- ApplicabilityAssertItem
- SerializationItem

4.22.4.5 BreakdownElement

The BreakdownElement class defined in UoF Breakdown Structure must also implement the following additional <>interface><:

- ApplicabilityAssertItem

4.22.4.6 MaintenanceLevel

The MaintenanceLevel class defined in UoF Product Usage Context must also implement the following additional <>interface><:

- ApplicabilityAssertItem

4.22.4.7 MaintenanceLocation

The MaintenanceLocation class defined in UoF Product Usage Context must also implement the following additional <>interface><:

- ApplicabilityAssertItem

4.22.4.8 OperatingLocationType

The OperatingLocationType class defined in UoF Product Usage Context must also implement the following additional <>interface><:

- ApplicabilityAssertItem

4.22.4.9 OperatingLocation

The OperatingLocation class defined in UoF Product Usage Context must also implement the following additional <>interface><:

- ApplicabilityAssertItem

4.22.4.10 PartAsDesigned

The PartAsDesigned class defined in UoF Part Definition must also implement the following additional <>interfaces><:

- ApplicabilityAssertItem
- SerializationItem

-
- 4.22.4.11 AlternatePartAsDesignedRelationship
The AlternatePartAsDesignedRelationship class defined in UoF Part Definition must also implement the following additional <>interface>>:
– ApplicabilityAssignmentItem
- 4.22.4.12 HardwareElementPartRealization
The HardwareElementPartRealization <>relationship>> class defined in UoF Breakdown Element Realization must also implement the following additional <>interface>>:
– ApplicabilityAssertItem
- 4.22.4.13 SoftwareElementPartRealization
The SoftwareElementPartRealization <>relationship>> class defined in UoF Breakdown Element Realization must also implement the following additional <>interface>>:
– ApplicabilityAssertItem
- 4.22.4.14 AllowedProductConfigurationByConfigurationIdentifier
The AllowedProductConfigurationByConfigurationIdentifier class defined in UoF Product Design Configuration must also implement the following additional <>interface>>:
– ApplicabilityAssertItem
- 4.22.4.15 KeyPerformanceIndicator
The KeyPerformanceIndicator <>attributeGroup>> defined in UoF LSA Candidate must also implement the following additional <>interface>>:
– ApplicabilityAssignmentItem
- 4.22.4.16 CorrectionFactor
The CorrectionFactor <>attributeGroup>> defined in UoF LSA Candidate must also implement the following additional <>interface>>:
– ApplicabilityAssignmentItem
- 4.22.4.17 FailureMode
The FailureMode class defined in UoF LSA FMEA must also implement the following additional <>interface>>:
– ApplicabilityAssignmentItem
- 4.22.4.18 FailureModeEffect
The FailureModeEffect class defined in UoF LSA FMEA must also implement the following additional <>interface>>:
– ApplicabilityAssignmentItem
- 4.22.4.19 ProductUsagePhase
The ProductUsagePhase class defined in UoF Special Events and Damage must also implement the following additional <>interface>>:
– ApplicabilityAssertItem
- 4.22.4.20 SpecialEventOccurrence
The SpecialEventOccurrence <>relationship>> class defined in UoF Special Events and Damage must also implement the following additional <>interface>>:
– ApplicabilityAssignmentItem

4.22.4.21 SpecialEventEffect

The SpecialEventEffect <>relationship>> class defined in UoF Special Events and Damage must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.22 TaskRequirement

The TaskRequirement class defined in UoF LSA Candidate Task Requirement must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.23 Subtask

The Subtask class defined in UoF Task must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.24 SubtaskTimeline

The SubtaskTimeline <>relationship>> class defined in UoF Task must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.25 SubtaskInZone

The SubtaskInZone <>relationship>> class defined in UoF Task must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.26 TaskRevisionWarningCautionNote

The TaskRevisionWarningCautionNote <>relationship>> class defined in UoF Task must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.27 SubtaskWarningCautionNote

The SubtaskWarningCautionNote <>relationship>> class defined in UoF Task must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.28 SubtaskAcceptanceParameter

The SubtaskAcceptanceParameter <>attributeGroup>> defined in UoF Task must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.29 SubtaskCircuitBreakerSettings

The SubtaskCircuitBreakerSettings class defined in UoF Task must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.30 SubtaskCircuitBreakerSettingsTimeline

The SubtaskCircuitBreakerSettingsTimeline <>relationship>> class defined in UoF Task must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.31 TaskResource

The TaskResource class defined in UoF Task Resources must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.32 TaskResourceRelationship

The TaskResourceRelationship <>relationship>> class defined in UoF Task Resources must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.33 ResourceRealization

The ResourceRealization <>relationship>> class defined in UoF Task Resources must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.34 TaskPersonnelResourceCompetence

The TaskPersonnelResourceCompetence <>relationship>> class defined in UoF Task Resources must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.35 TaskTarget

The TaskTarget <>relationship>> class defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.36 TaskFrequency

The TaskFrequency <>attributeGroup>> defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.37 AllocatedMaintenanceLevel

The AllocatedMaintenanceLevel <>relationship>> class defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.38 TimeLimit

The TimeLimit class defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.39 ThresholdDefinition

The ThresholdDefinition class defined in UoF Task Usage (Part 1) must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.40 SecurityClassification

The SecurityClassification <>relationship>> class defined in UoF Security Classification must also implement the following additional <>interface>>:

- ApplicabilityAssignmentItem

4.22.4.41 DocumentAssignment

The DocumentAssignment <<relationship>> class defined in UoF Document must also implement the following additional <<interface>>:

- ApplicabilityAssignmentItem

4.22.4.42 OrganizationAssignment

The OrganizationAssignment <<relationship>> class defined in UoF Organization Assignment must also implement the following additional <<interface>>:

- ApplicabilityAssignmentItem

4.22.4.43 RemarkAssignment

The RemarkAssignment <<relationship>> class defined in UoF Remark must also implement the following additional <<interface>>:

- ApplicabilityAssignmentItem

4.22.4.44 AuthorizedLife

The AuthorizedLife <<compoundAttribute>> defined in the S-Series Compound attributes must also implement the following additional <<interface>>:

- ApplicabilityAssignmentItem

4.22.4.45 ClassificationType

The ClassificationType <<primitive>> defined in S-Series Primitives (Data Types) must also implement the following additional <<interface>>:

- ApplicabilityAssignmentItem

Note

Assignment of ApplicabilityStatements to instances of ClassificationType enables multiple classifications to be given for a single attribute in the data model, where multiple classifications are allowed, and still distinguish when a single classification is applicable.

4.22.4.46 DescriptorType

The DescriptorType <<primitive>> defined in S-Series Primitives (Data Types) must also implement the following additional <<interface>>:

- ApplicabilityAssignmentItem

Note

Assignment of ApplicabilityStatements to instances of DescriptorType enables multiple descriptive texts to be given for a single attribute in the data model, where multiple descriptions are allowed, and still distinguish when a single descriptive text is applicable.

4.22.4.47 PropertyType

The PropertyType <<primitive>> defined in S-Series Primitives (Data Types) must also implement the following additional <<interface>>:

- ApplicabilityAssignmentItem

Note

Assignment of ApplicabilityStatements to instances of PropertyType enables multiple values to be given for a single attribute in the data model, where multiple values are allowed, and still distinguish when a single value is applicable.

Chapter 20

Data exchange

Table of contents

	Page
Data exchange	1
References.....	1
1 General	1
1.1 Introduction	1
1.2 Objective.....	2
1.3 Scope.....	2
2 Overview.....	2
2.1 S3000L data exchanges	2
2.1.1 Data flow from other disciplines to LSA.....	2
2.1.2 Data flow from LSA to other disciplines or stakeholders.....	3
3 S3000L XML Schemas.....	3
4 PLCS (Product Life Cycle Support)	4

List of tables

1 References	1
-------------------------	---

List of figures

1 S3000L data exchanges	2
2 ASD XML Schema to PLCS implementation mapping.....	4

References

Table 1 References

Chap No./Document No.	Title
Chap 19	Data model
ISO 10303-239 PLCS	Product Life Cycle Support (PLCS)
PLCS PSM	PLCS Platform Specific Model, refer to OASIS PLCS TC www.oasis-open.org

1 General

1.1 Introduction

This chapter defines the standard technology used for the exchange of data that results from Logistics Support Analysis as defined in S3000L. The exchange of data for S3000L, issue 1.1 is defined using XML and XML Schema.

S3000L XML Schemas will be published separately on the S3000L website (www.s3000l.org).

1.2 Objective

The objective for this chapter is to describe how S3000L XML Schemas support the S3000L LSA process and its interaction with other business processes.

1.3 Scope

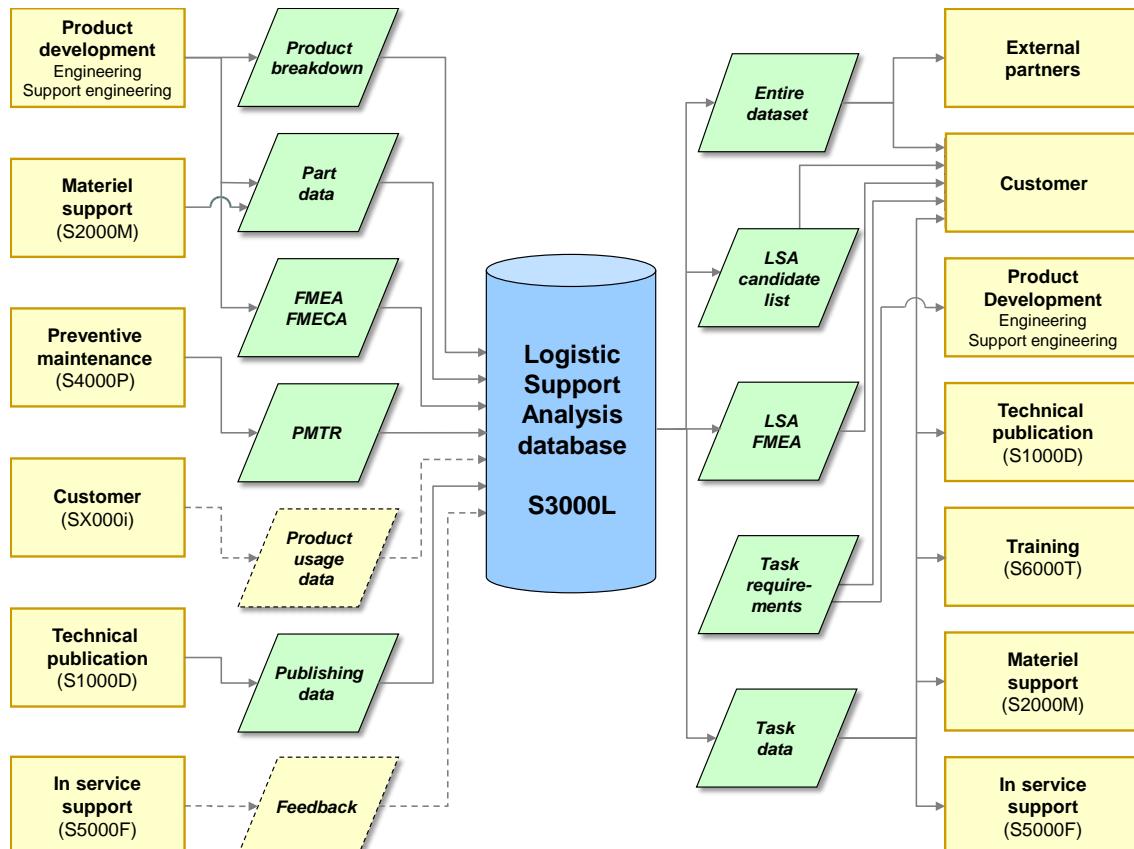
The following areas are within scope of the data exchange:

- Overview of S3000L data exchange using S3000L XML Schemas
- Overview of the defined S3000L XML Schemas
- Relationship between S3000L XML Schemas and ISO 10303-239 PLCS

2 Overview

2.1 S3000L data exchanges

An overview of the types of data exchanges that is supported by S3000L XML Schemas is summarized in [Fig 1](#):



ICN-B6865-S3000L0236-002-00

Fig 1 S3000L data exchanges

2.1.1 Data flow from other disciplines to LSA

The identified data exchanges basically contain the following information:

- *Product breakdown* - from product development (engineering and support engineering) to LSA, includes information on:
 - The Product to be supported
 - One or many Product variants

- One or many breakdowns of the Product, including its breakdown elements (eg, systems, functions, hardware, software, zones)
- Possible breakdown element realizations in terms of hardware or software parts
- Product variant applicability and specific Product variant realizations
- Key performance indicators for each LSA candidate
- *Part data* - from product development (engineering and support engineering) and material management to LSA, includes information on, eg:
 - Part identifiers (eg, part numbers)
 - Part name
 - Operational authorized life
- *FMEA/FMECA* - from product development (engineering and support engineering) to LSA includes identified failure modes on parts that will be used in the Product to be supported can require the definition of corrective maintenance tasks
- *Preventive maintenance task requirements (PMTR)* - from Preventive Maintenance Analysis (PMA) to LSA identifies the need for preventive maintenance tasks including intervals and thresholds
- *Publishing data* - from technical publication to LSA, includes cross reference data between S3000L tasks and S1000D data module codes

2.1.2 Data flow from LSA to other disciplines or stakeholders

The identified data exchanges basically contain the following information:

- *Entire dataset* - from LSA to external partners contain the complete LSA dataset
- *LSA candidate list* - from LSA to the customer includes information on selected LSA candidates and recommended (decided) analysis activities
- *LSA FMEA* - from LSA to the customer includes information to justify the recommended corrective maintenance tasks
- *Task requirement* - from LSA to customers and product development (engineering and support engineering), includes information on, eg:
 - Task requirement specification
 - Task requirement authority
 - Task limit requirements
- *Task data* from LSA to the customer, technical publication, training, materiel support and in-service support includes information on, eg:
 - Task identification
 - Task justification
 - Subtasks
 - Task resources

3 S3000L XML Schemas

The S3000L XML Schemas are derived from the S3000L data model defined in [Chap 19](#). The method of mapping the S3000L data model to the S3000L XML Schemas is done in accordance with the XML Schema Authoring Rules defined by the DMEWG (Data Model and Exchange Working Group).

An added feature for the exchange of LSA data using the XML Schemas of S3000L, issue 1.1 is the option to only exchange updates. Update messages can be sent in between complete (baseline) messages and can accommodate minor as well as major changes to the LSA data.

This means that the receiver of LSA data does not need to analyze what actions to take with respect to update the target data set (eg, database).

The XML Schema used for complete (baseline) messages enforces all rules defined in the S3000L data model in order to guarantee consistency in the exchanged data set.

Another additional feature of the XML Schemas of S3000L, issue 1.1 is the support of all UoFs defined in [Chap 19](#).

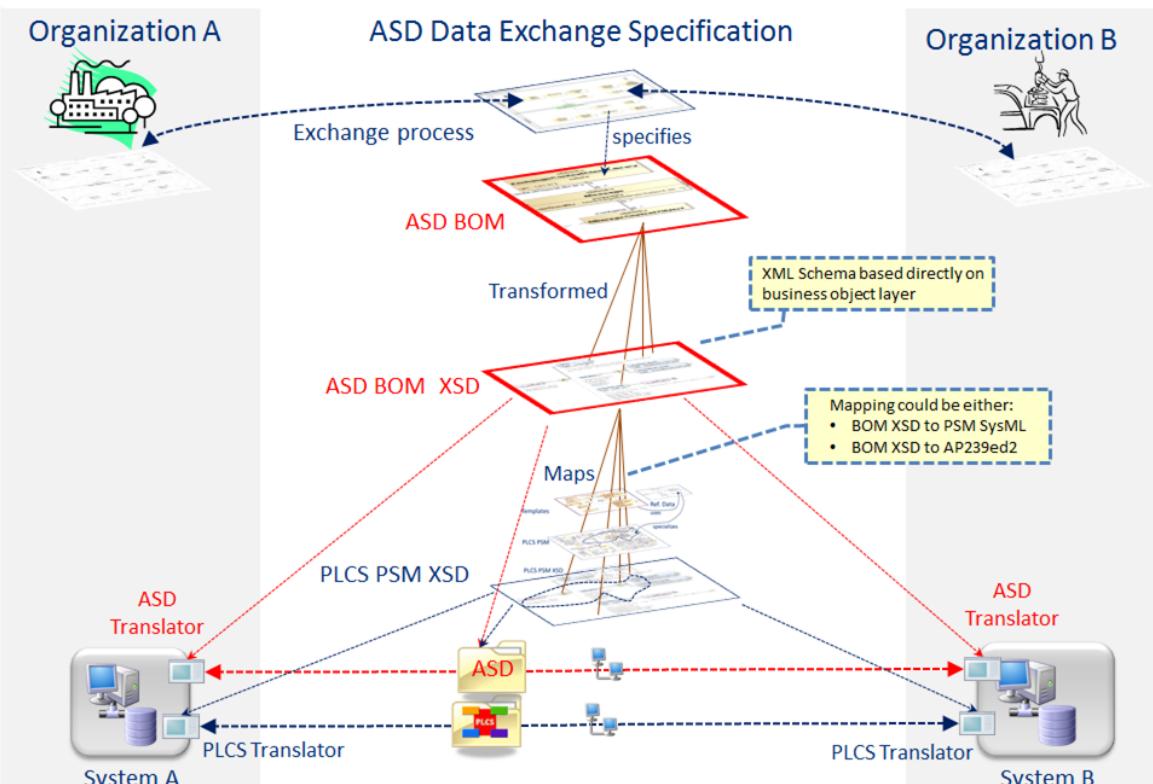
4 Product Life Cycle Support (PLCS)

S3000L XML Schemas will provide mappings to ISO 10303-239 PLCS ed2 or OASIS PLCS PSM, in order to support continued use of ISO 10303-239 PLCS. These mappings will be an integral part of the respective S3000L XML Schemas.

OASIS PLCS PSM is an XML Schema which is published as an alternative (complement) to ISO 10303-239 PLCS (Product Life Cycle Support) and its related exchange formats ISO 10303:21 and ISO 10303:28, respectively.

The rationale for introducing S3000L XML Schemas as the basis for supporting S3000L data exchanges is to allow for organizations that do not have the required PLCS skills to still support the S3000L specified data exchanges.

In awaiting ISO 10303-239 PLCS ed3 and its associated data exchange development environment, all future S-Series ILS specifications will follow the same XML Schema approach as described for S3000L. [Fig 2](#) illustrates how S-Series ILS specification XML Schemas is to be viewed in general in respect of ISO 10303-239 PLCS and OASIS PLCS PSM.



B6865-S3000L0237-001-00

Fig 2 ASD XML Schema to PLCS implementation mapping



The S-Series ILS specifications XML Schemas are targeted to support data exchange at the Business Object Model (BOM) layer. However, each XML Schema will also include the mapping details required for an unambiguous mapping of each element and attribute to PLCS in order to enable PLCS-based data exchanges and/or PLCS-based data consolidation.

Chapter 21

Terms, abbreviations and acronyms

Table of contents

Document

Chap 21	Terms, abbreviations and acronyms	S3000L-A-01-00-0000-00A-040A-A
Chap 21.1	Terms, abbreviations and acronyms - Introduction	S3000L-A-01-01-0000-00A-040A-A
Chap 21.2	Terms, abbreviations and acronyms - Glossary of terms	S3000L-A-01-02-0000-00A-040A-A
Chap 21.3	Terms, abbreviations and acronyms - Abbreviations and acronyms	S3000L-A-01-03-0000-00A-040A-A

Chapter 21.1

Terms, abbreviations and acronyms - Introduction

Table of contents

	Page
Terms, abbreviations and acronyms - Introduction.....	1
References.....	1
1 General	1
2 Content	1

List of tables

1 References.....	1
----------------------------	---

References

Table 1 References

Chap No./Document No.	Title
Chap 21.2	Terms, abbreviations and acronyms - Glossary of terms
Chap 21.3	Terms, abbreviations and acronyms - Abbreviations and acronyms

1 General

[Chap 21](#) provides a comprehensive terminology dictionary for the terms, abbreviations and acronyms used throughout S3000L.

2 Content

A comprehensive terminology dictionary for terms used in this specification is given in [Chap 21.2](#). A dictionary of abbreviations and acronyms used in this specification is presented in [Chap 21.3](#). These include all specific S3000L definitions, abbreviations and some basic abbreviations to be used when completing Logistic Support Analysis.

Chapter 21.2

Terms, abbreviations and acronyms - Glossary of terms

Table of contents

	Page
Terms, abbreviations and acronyms - Glossary of terms	1
References.....	1
1 General	1
2 Glossary of terms.....	1

List of tables

1 References	1
2 Glossary of terms.....	1

References

Table 1 References

Chap No./Document No.	Title
None	

1 General

[Table 2](#) provides the definitions of terms used throughout this specification.

2 Glossary of terms

Table 2 Glossary of terms

Term	Definition
Availability	Availability is the measure of the degree to which an item is in an operable and ready-for-use state at the start of a mission or operation, when the mission or operation is called for at an unknown time. This is sometimes called operational readiness.
Breakdown Element Identifier	The Breakdown Element Identifier (BEI) is a code that identifies the breakdown structure of all system items within the LSA. The breakdown can include software items.
Breakdown Element Revision	The Breakdown Element Revision (BER) is a code which indicates that an alternative equipment or component occupies the same position in the breakdown and, therefore, the same installation position.

Term	Definition
Built In Test	<p>Built-In Test (BIT) are implemented on items to enable them to carry out some self-testing up to a given degree.</p> <p>Usually, three types of BIT are implemented:</p> <ul style="list-style-type: none"> - power-on BIT (P-BIT) executed at start-up of the item - continuous BIT (C-BIT) periodically and automatically executed during the operation of the item, without any intervention from the operating crew - initiated BIT (I-BIT) executed upon order from the operator or from the maintenance team <p>Each of these types of tests detects specific categories of failures.</p> <p>They are characterized by:</p> <ul style="list-style-type: none"> - a detection rate that gives the percentage of functions or items whose failure is detected - a localization or isolation rate that gives the percentage of detected failures that are associated without ambiguity to the failure of an identified replaceable unit - a false alarm rate <p>Built-in tests are likely to:</p> <ul style="list-style-type: none"> - detect failures - localize failed replaceable units <p>Localization can be:</p> <ul style="list-style-type: none"> - ambiguous, which means that the failure has been localized within a group of replaceable parts, not on a single replaceable part - non-ambiguous, which means that the failed item has been identified and localized with an acceptable degree of certainty <p>Removing the ambiguity is the goal of troubleshooting analysis.</p> <p>In terms of warning and signals, BIT results can be signaled to the operator through a warning or alarm (which is usually the case for critical failures).</p>
Candidate Item	A Candidate Item (CI) is an item that has been identified that requires analysis within the LSA process.
Candidate Item List	A Candidate Item List (CIL) documents the CI's that are to be analyzed (documentation of the selection of analysis tasks).
chemical abstract substance identification number	Unique international identifier for a given substance.
Commercial Off-The-Shelf (COTS)	Software or hardware, generally technology products, that is ready-made and available for sale, lease, or license to the general public.
common cause	Some failures can lead to several malfunctions. For instance, the failure of a power supply leads to malfunction of all its supplied items. This type of failure with multiple impacts is called a common cause.

Term	Definition
corrective maintenance	All maintenance activities, which are carried out to reset a faulty item to full functionality.
Customer Requirements Document	The Customer Requirements Document (CRD) contains all the customer logistic requirements. This document must be available for the Guidance Conference.
damage	A loss or reduction of functionality, excluding inherent failure (intrinsic reliabilities). Normally a maintenance task will be required. Damages can be grouped into "damage families" (eg, concerning structures, typical damage can be identified like scratches, dents or cracks). These damage families are typical candidates for a standard repair procedure.
dangerous/toxic substance	Substance which harms human beings and environment (animals, plants, etc).
Data Element List (DEL)	List of selected data elements or an output of a data element tailoring process. This list can contain additional data elements required for a project, which are not predefined in any standard.
defect	Any non-conformance of an item with its specified requirements is a defect. Note that a defect does not necessarily result in a failure of the item.
definition files	Definition files, such as technical descriptions, interface definitions, wiring diagrams, description of sockets bounding, etc. This can be useful if the localization remains ambiguous after simple troubleshooting has been carried out and when it becomes necessary to carry out functional or electrical tests to help isolate the failed item.
demilitarization	Operation which makes a defense Product unavailable for its intended military use.
detection	Failure detection warns the operator that a failure has occurred
direct replaceable	Item that is directly replaceable on the Product.
deviation	Authorized approval to depart from a particular requirement of a Product/equipment approved configuration documentation for a specified period of time. This allows the acceptance of an Product/equipment which departs from the particular requirement, but is considered as suitable for use 'as is' or after repair by an approved method.
disposal phase	Last phase of a Product Life Cycle, where the Product is phased out of use.
disposal process	Set of tasks to be performed on a Product at the end of its intended use
external cause	A cause is said to be external when an event independent of Product usage occurs eg, bird-strike.
failure	Unacceptable reduction of functionality of an item where the item cannot continue its intended use. The failure occurs during proper usage of the item.

Term	Definition
Failure Cause	A Failure Cause (FC) is any circumstance during design, manufacture or use of the Product, which led to the failure.
failure mode	A failure mode is a predicted or observed physical, mechanical, thermal or other process which leads to a failure. The result of this process is stated in relation to the operating conditions at the time of the failure.
failure rate	The failure rate is the probability of failure per unit of time of items in operation
Failure Modes and Effects Analysis	A Failure Modes and Effects Analysis (FMEA) is a procedure for analysis of potential failure modes within a system for classification by severity or determination of the effect of failures on the system.
Failure Mode, Effects, and Criticality Analysis	Failure Mode, Effects, and Criticality Analysis (FMECA) is an extension of Failure Mode and Effects Analysis (FMEA). In addition to the basic FMEA, it includes a criticality analysis, which is used to chart the probability of failure modes against the severity of their consequences. The probability that a failure mode is caused by a fault item is calculated during the analysis. The severity of the consequences of this fault determines the remedial action required to correct the fault and return the item to service.
Field Loadable Software (FLS)	Software that can be installed to one or several equipments on a Product without need to remove the equipment from its installation location.
firmware	Software that can be loaded into a Line Replaceable Unit (LRU) or Shop Replaceable Unit (SRU) but requires the target to be dismounted from its installation on the operational system and requiring the replacement of a component.
fit form and function	Where the physical interface, physical parameters that uniquely characterize or the function of a Product/equipment remain unchanged.
function breakdown	A simple functional breakdown can start on top level of the Product as the root of the breakdown tree. The different functions of the Product must be documented from the main functions down to the sub-functions, etc, to the required depth. A functional breakdown must not contain any physical items.

Term	Definition
functionality analysis	<p>Functional analysis describes and functions performed by the Product, systems, subsystems and sub-subsystems.</p> <p>Each function of the system can be characterized by the flows or parameters that:</p> <ul style="list-style-type: none"> - it takes as inputs - it requires as part of operating context - it provides as outputs <p>The outputs provided by the characterization justify the inclusion of the items in the design. The results of the functionality analysis are used to identify the impact of a failed physical item or items on other associated functions. The results also indicate the detection and corrective action required preventing the failure and ensuring continuity and/or restoration of the items intended functionality.</p>
functional symptom	A functional symptom can be observed during a functional check and/or the failure of an item associated with the function.
Guidance Conference (GC)	A conference at which the supplier and the customer agree the scope and depth of the LSA business processes and procedures.
Guidance Conference Document (GCD)	A signed, contractual record of the agreements reached by the supplier and the customer. This document is an output of the Guidance Conference.
Integrated Logistic Support	Integrated Logistics Support (ILS) is an integrated and iterative process for developing materiel and a support strategy that optimizes functional support, leverages existing resources, and guides the system engineering process to quantify and lower life cycle cost and decrease the logistics footprint (demand for logistics), making the system easier to support.
internal cause	A cause is said to be internal when it comes from normal Product usage by itself (eg, excessive vibration).
labor time	The sum of the man-hours taken to complete a task. If more than one person is working on a task at the same time, the time taken by each person must be summarized by skill level.
Level Of Repair Analysis	Level Of Repair Analysis (LORA) is a prescribed procedure for determining the most economical and efficient level at which maintenance is performed.
Line Replaceable Unit (LRU)	An item, which can be removed and installed at the organizational maintenance area.
localization	Failure isolation that indicates an item or group of items that have failed. This isolation is usually associated with a failure detection.
Logistic Support Analysis (LSA)	A structured approach to increase in maintenance efficiency and reduction of the cost of providing support by preplanning all aspects of Integrated Logistics Support.
LSA database	A database that records the logistic data gathered as a result of the ILS processes.

Term	Definition
life cycle	Total Product lifespan since its initial feasibility phase, until its service withdrawal and disposal.
Life Cycle Costs	Life Cycle Cost (LCC) is the cumulative cost of a Product over its life cycle from entry into service to disposal as determined by a process of economic analysis that allows for the assessment of the total cost of acquisition, ownership and disposal of the Product.
maintainability	The measure of the ability of an item to be retained in, or restored to a specified condition, when maintenance is performed by personnel having specified skill levels, using prescribed procedures and resources, at each prescribed level of maintenance and the level at which maintenance can be carried out for the continuance of the Product's mission as limited by the design
Maintenance Concept (MC)	A strategy for providing maintenance support to ensure a Product meets its mission performance requirements
Maintenance Plan	A plan for conducting maintenance in the most efficient and cost effective manner such that the Product performs as intended, to meet its mission requirements.
	The Maintenance plan specifies when, where, and how maintenance tasks are performed on the Product/equipment, including both: <ul style="list-style-type: none"> – Preventive maintenance – Corrective maintenance
Maintenance Task Analysis	The plan ensures that all required maintenance resources are in place to support deployment of the Product. The Maintenance Task Analysis (MTA) is a detailed analysis process that results in the identification of the procedures, spares and materials, tools, support equipment, personnel skill levels, estimated and elapsed times as well as any facility issues that must be considered for a corrective, repair or preventive maintenance task
Maintenance Relevant Item	A Maintenance Relevant Item (MRI) is an item that can be repaired or replaced in case of a failure or damage. Normally this item is automatically a potential LSA candidate, but not automatically an MSI.
Maintenance Significant Item	A Maintenance Significant Item (MSI) is an item that was identified by any selection process as the result of a Scheduled Maintenance Analysis (SMA) procedure such as S4000M. An MSI can become a LSA candidate if the SMA identifies a scheduled task which is documented in the LSA database.

Term	Definition
malfunction analysis	<p>Malfunctioning analysis aims to define actions to be taken in the event of a failure. It can cover two types of corrective actions:</p> <ul style="list-style-type: none"> – actions to be taken by the operating crew when a failure occurs during operation – actions to be taken as part of the corrective maintenance <p>Only the second type of actions falls into the scope of this specification.</p>
Mean Elapsed Time (MET)	The average time expended, irrespective of the number of personnel, required to complete a task. The duration of the whole task can be calculated as the sum of times coming from the subtasks and from the referenced supporting tasks.
Mean Time Between Failures	Mean Time Between Failures (MTBF) is the predicted elapsed time between inherent failures of a system during operation. MTBF can be calculated as the arithmetic mean (average) time between failures of a system. The MTBF is typically part of a model that assumes the failed system is immediately repaired (zero elapsed time), as a part of a renewal process. This is in contrast to the Mean Time To Failure (MTTF), which measures average time between failures with the modeling assumption that the failed system is not repaired.
Model Identifier	The Model Identifier is a code, which uniquely identifies the Product. It is recommended to use the model identifier in conjunction with other identifiers within the entire ILS process.
Operational Requirements Document	The Operational Requirements Document (ORD) defines the operational requirements in a qualitative way for the Product. This document must be available for the Guidance Conference.
physical breakdown	A top down representation of hardware and software of a Product based on the engineering design model/drawings.
physical symptom	A physical symptom characterizes a failure detected by visual inspection, measurement of a wear-out parameter, material degradation. It is detectable or measurable when the system is currently operated or if it is undergoing inspection or maintenance.
preventive maintenance	Maintenance activities to prevent the occurrence of critical failures or damages in conjunction with safety, economical or ecological aspects. The Preventive Maintenance also includes activities after special events where these events, chronological intervals or other regular threshold cannot be defined.
Product	Any platform, system or equipment (air, sea, land vehicle, equipment or facilities, civil or military). The term the Product is used in this specification to mean platforms, systems, subsystems, sub-subsystems, assemblies, equipment, LRU, components, parts etc.
realization	The completion of a Product breakdown from a design breakdown, including different system/subsystem or components for the same installed location.

Term	Definition
rectifying task	A maintenance activity, which resolves an event such as failures, damages, special events or thresholds. A rectifying task contains subtasks in terms of referenced supporting tasks and/or definite working steps.
reliability	The probability of failure free performance of a Product/equipment under stated conditions, or the probability that an item can perform its intended function for a specified interval under stated conditions, is a prime driver of support resources.
Reliability Centered Maintenance	Reliability Centered Maintenance (RCM) is a disciplined logic or methodology used to identify scheduled maintenance tasks to maintain the inherent reliability of equipment at a minimum expenditure of resources.
scheduled maintenance	Maintenance activities to prevent the occurrence of critical failures or damages in conjunction with safety, economical or ecological aspects. These maintenance tasks are defined with a corresponding interval or threshold (eg, after certain time, cycles, rounds, distance). Scheduled maintenance is a subset of preventive maintenance.
Scheduled Maintenance Analysis Candidate	Element of the Product breakdown, which is identified as a potential candidate for which a scheduled or preventive maintenance task could be necessary.
Scheduled Maintenance Analysis (SMA)	The analysis task for the identification of preventive and scheduled maintenance. Different methods are available to perform this analysis task such as S4000M.
Shop Loadable Software (SLS)	Software that can be loaded into a LRU but that requires the target LRU to be dismounted from its installation in the system where it is located.
Shop Replaceable Unit (SRU)	An item, which can only be changed at shop/rear level (cannot be removed and installed at the organizational maintenance area).
signature	A list of physical symptoms, functional symptoms and BIT results that help detect and localize the failure.
Software Support Analysis	Software Support Analysis (SSA) is the methodology to analyze software with the purpose to supply the customer with all necessary information to establish a cost effective Software Support Concept. This includes all equipment, tools, personnel, documentation, infrastructure and required skill and training respectively.
special event	A special event is something that can occur during the life of the Product but cannot be considered as a normal way of operation. It can be due either to external causes, (eg, meteorological phenomenon or to out of bound use such as over G maneuver).

Term	Definition
specific test equipment	<p>Whenever BIT is unable to detect or localize failures, some specific test equipment can be required. This specific test equipment can include:</p> <ul style="list-style-type: none"> – external measurement or test items – external means used to simulate a function of the operating system
structural detail	A specific area of the Structure Significant Item (SSI) which was identified by any selection process from an SMA procedure such as S4000M. For this detail, an SMA will be performed.
Structural Item	A Structural Item (SI) is a part of the Product bodywork. It can be an LSA candidate as well as a Structure Significant Item.
Structure Significant Item	An item which is identified by the selection process from a Scheduled Maintenance Analysis procedure such as S4000M.
substances cartography	Characterization (identification, risk, quantity, etc) and location of all the tracked substances within a Product.
supply chain	A supply chain consists of all organizations involved, directly or indirectly, in fulfilling a customer support requirements.
supply chain management	The process of planning, implementing and controlling the operations of the supply chain as efficiently as possible.
supporting task	A supporting task is a part of a complete maintenance activity. As a standalone task it cannot rectify an event such as failures, damages, special events or thresholds. A supporting task contains subtasks in terms of working steps.
testability	<p>Performances of BIT developed on some items of the system are often assessed using FMEA terms and results, but this is not mandatory, particularly when the item has been previously developed for another program. BIT can be characterized by:</p> <ul style="list-style-type: none"> – a list of functions or sub-functions that are tested – a detection rate that gives the percentage of functions or items whose failure is detected – a localization or isolation rate that gives the percentage of detected failures that are associated without ambiguity to the failure of an identified item or component – a “No Fault Found” rate <p>These analyses and their justification are used for malfunctioning analysis and for LSA FMEA, especially because they provide a list of failure reports (from BIT) giving failure codes or any other message that indicates which item or group of items is liable to have failed, therefore restraining greatly the number of items to be investigated as part of the troubleshooting .</p> <p>Warnings are sent to the operating crew or to the maintenance team to indicate detected failures. These warnings are used as initial symptoms and not necessarily sent from the faulty item causing the failure.</p>
Training Needs Analysis (TNA)	Analysis task to identify the requirements for training for operational and maintenance activities.

Term	Definition
troubleshooting	Troubleshooting consists of localizing failed replaceable units when their failure is not obvious or previously achieved by other means (eg, built-in test). Troubleshooting is carried out after a failure has been detected.
unified identifier	Designation for an identifier (ID), which can be used across all logistic disciplines in the ILS process for a definite identification of any item.
Usable on Code (UoC)	A Product, system or equipment identifier that indicates the configuration of a Product on which the item under analysis is used.

Chapter 21.3

Terms, abbreviations and acronyms - Abbreviations and acronyms

Table of contents

	Page
1 General	1
2 Word combination - Acronym	1
3 Tense and number	1
4 Abbreviation and acronym list	2

List of tables

1 References	1
2 Abbreviations and acronyms	2

References

Table 1 References

Chap No./Document No.	Title
None	

1 General

When there is doubt that an abbreviation or acronym will be understood or whenever there is ample space to write in full, the term must be written out rather than abbreviated.

[Table 2](#) provides the definitions of abbreviations and acronyms used throughout this specification.

2 Word combination - Acronym

Abbreviations for word combinations, acronyms, must be used as such and not separated for use singly, unless authorized singly.

Single abbreviations can be combined when necessary if there is no abbreviation listed for the combination.

3 Tense and number

Abbreviations are used in a consistent manner throughout this specification to represent all tenses, singular and plural of their definition.

4 Abbreviation and acronym list

Table 2 Abbreviations and acronyms

Abbreviation / Acronym	Definition
AIA	Aerospace Industries Association of America
AD	Accidental Damage
AOR	Annual Operating Rate
ASD	AeroSpace and Defense Industries Association of Europe
BE	Breakdown Element
BEI	Breakdown Element Identifier
BER	Breakdown Element Revision
BIT	Built in Test
BITE	Built in Test Equipment
CBS	Cost Breakdown Structure
CE	Communauté Européenne
CEE	Communauté Economique Européenne
Chap	Chapter
CI	Candidate Item
CIL	Candidate Item List
CM	Configuration Management
CMR	Certification Maintenance Requirements
COTS	Commercial Off The Shelf
CP	Change Proposal
CRD	Customer Requirements Document
DA	Design Authority
DADD	Data and Definition Document
DEL	Data Element List
DEX	Data Exchange Specification
DLM	Depot Level Maintenance
DMC	Data Module Code
DoD	Department of Defense
DRACAS	Data Reporting and Corrective Action System
DSE	Damage and Special Event

Abbreviation / Acronym	Definition
ED	Environmental Deterioration
ELORA	Economical Level of Repair Analysis
FC	Failure Cause
FFE	Functional Failure Effect
FFEC	Functional Failure Effect Code
Fig	Figure
FLS	Field Loadable Software
FMA	Failure Mode Analysis
FMEA	Failure Mode and Effects Analysis
FMECA	Failure Mode and Effects Criticality Analysis
FMR	Failure Mode Ratio
FTA	Fault Tree Analysis
GC	Guidance Conference
GCD	Guidance Conference Document
GEIA	Government Electronic and Information Technology Association
GVI	General Visual Inspection
ICN	Information Control Number
ID	Identifier
ILC	Item Location Code
ILS	Integrated Logistic Support
ISMO	In Service Maintenance Optimization
ISO	International Standards Organization
IT	Information Technology
IUA	Item Under Analysis
kg	kilogram
LAN	Local Area Network
lb	pound
LCC	Life Cycle Costs
LCMP	LSA Configuration Management Plan
LCN	Logistic Control Number
LORA	Level of Repair Analysis
LRU	Line Replaceable Unit

Abbreviation / Acronym	Definition
LSA	Logistic Support Analysis
LSA RC	Logistic Support Analysis Review Conference
\overline{M}	Mean active maintenance action time
MC	Maintenance Concept
MDT	Maintenance Down Time
MET	Mean Elapsed Time
MIL-STD	Military Standard (US DoD)
ML	Maintenance Level
MoD	Ministry of Defence
MRI	Maintenance Relevant Item
MRO	Maintenance, Repair & Overhaul
MSDS	Material Safety Data Sheet
MSI	Maintenance Significant Item
MSN	Manufacturer Serial Number
MTA	Maintenance Task Analysis
MTBF	Mean Time Between Failures
MTBM	Mean Time Between Maintenance
MTTR	Mean Time to Repair
NATO	North Atlantic Treaty Organization
O&S	Operating and Support
OASIS	Organisation for the Advancement of Structural Information Standards
OEM	Original Equipment Manufacturer
OM	Obsolescence Management
OMP	Operator Maintenance Plan
ORD	Operational Requirements Document
para	Paragraph
PBS	Product Breakdown Structure
PDF	Product Disposal File
PDM	Product Data Management
PE	Periodical
PHST	Packaging, Handling, Storage and Transportation

Abbreviation / Acronym	Definition
PLCS	Product Life Cycle Support
P/N	Part Number
PMTR	Preventative Maintenance Task Requirement
PNR	Part number
PO	Perform once
PP	Program Plan
PPH	Policy and Procedure Handbook
RC	Review Conference
RCM	Reliability Centered Maintenance
RDL	Reference Data Library
RFC	Request For Clarification
SB	Service Bulletin
SD	Significant Detail
SI	Structural Item
SLS	Shop Loadable Software
SMA	Scheduled Maintenance Analysis
SNS	Standard Numbering System
SOW	Statement of Work
SRU	Shop Replaceable Unit
SSA	Software Support Analysis
SSC	Software Support Concept
SSI	Structural Significant Item
ST	Sub-Task
SUA	System Under Analysis
SW	Software
TNA	Training Needs Analysis
UML	Unified Modeling Language™
UoF	Unit of Functionality
WAN	Wide Area Network
WBS	Work Breakdown Structure
ZAM	Zonal Analysis Module

Chapter 22

Data element list

Table of contents

	Page
Data element list	1
References.....	1
1 General	1

List of tables

1 References	1
2 Data elements in alphabetical order	2
3 Attributes of the S3000L data types	49

References

Table 1 References

Chap No./Document No.	Title
Chap 19	Data elements
Chap 20	Data exchange
S1000D	International specification for technical publications using a common source database
S2000M	International specification for material management
S4000P	International specification for developing and continuously improving preventive maintenance

1 General

This chapter defines all the data elements that are used as attributes in the S3000L data model, refer to [Chap 19](#).

The data element list in [Table 2](#) is organized alphabetically by the data element name, and contains:

- Data element name
- Data element data type (refer to [Chap 19](#) on more details on data types used in S3000L)
- Data element definition, contains a textual definition and a list of valid values
- Class/Interface name, identifies Classes/Interfaces in the S3000L data model where the data element is used as an attribute ([Chap 19](#))
- Unit of Functionality, identifies the section in [Chap 19](#) where the Class is defined

The attribute list in [Table 3](#) defines all the data elements that are used to define the S3000L data types.

Table 2 Data elements in alphabetical order

Data element	Type	Definition	Class/Interface	UoF
additionalTrainingRequirementsDescription	DescriptorType	additionalTrainingRequirementsDescription is a narrative statement which describes the additional training which is necessary to achieve a specific personnel competence.	AdditionalTrainingRequirement	Task Resources
aggregatedElementDescription	DescriptorType	aggregatedElementDescription is a phrase that gives more information on the aggregated element under consideration.	AggregatedElementRevision	Breakdown Aggregated Element
aggregatedElementType	ClassificationType	aggregatedElementType is a classification that identifies further specialization of an aggregated element. Examples: - family - function - slot - system	AggregatedElement	Breakdown Aggregated Element
allowedProductConfigurationIdentifier	IdentifierType	allowedProductConfigurationIdentifier is a string of characters that are unique to the issuing organization which is used to designate an AllowedProductConfiguration and to differentiate it from other AllowedProductConfigurations.	AllowedProductConfiguration ByConfigurationIdentifier	Product Design Configuration
applicabilityEndDate	DateType	applicabilityEndDate is a date that defines the upper bound of the interval for a dated applicability statement.	DatedApplicabilityStatement	Applicability Statement
applicabilityStartDate	DateType	applicabilityStartDate is a date that defines the lower bound of the interval for a dated applicability statement.	DatedApplicabilityStatement	Applicability Statement
applicabilityStatementDescription	DescriptorType	applicabilityStatementDescription is a narrative statement that provides a human readable description of the rule expressed in the applicability statement.	ApplicabilityStatement	Applicability Statement
applicabilityStatementIdentifier	IdentifierType	applicabilityStatementIdentifier is a string of characters used to uniquely identify an ApplicabilityStatement and to differentiate it from other ApplicabilityStatements.	ApplicabilityStatement	Applicability Statement

Applicable to: All

S3000L-A-22-00-0000-00A-040A-A
Chap 22

Data element	Type	Definition	Class/Interface	UoF
applicableSerialNumberRange	SerialNumberRange	applicableSerialNumberRange is a design characteristic that identifies an interval of serialized items which may be an open-ended.	ApplicableBlockOfSerializedEndItems BlockOfSerializedItems	Product Design Configuration / Applicability Statement
authorityToOperateIdentifier	IdentifierType	authorityToOperateIdentifier is a string of characters that are unique to the issuing organization which is used to designate an AuthorityToOperate and to differentiate it from other AuthorityToOperate.	AuthorityToOperate	Product Design Configuration
breakdownElementEssentiality	ClassificationType	breakdownElementEssentiality is a design classification that identifies the operational impact of the breakdown element at the product level. <i>Note: Based on the criticality as defined during the FMECA.</i>	BreakdownElement	Breakdown Structure
breakdownElementIdentifier	IdentifierType	breakdownElementIdentifier is a string of characters used to uniquely identify a BreakdownElement and to differentiate it from other BreakdownElements that comprise a product. Examples: <ul style="list-style-type: none">- the combination of logistics support analysis control number and alternate logistics support analysis control number within GEIA-STD-0007- Standard Numbering System defined by S1000D <i>Note: Can be used to establish a hierarchical structure of the technical system.</i>	BreakdownElement	Breakdown Structure
breakdownElementName	DescriptorType	breakdownElementName is a word or phrase by which the breakdown element is known and can be easily referenced. <i>Note: It is recommended that breakdownElementName be the same as technical name (techname) in S1000D.</i>	BreakdownElement	Breakdown Structure



Data element	Type	Definition	Class/Interface	UoF
breakdownElementRelationshipType	ClassificationType	<p>breakdownElementRelationshipType is a classification that identifies the type of relationship established between two BreakdownElementRevisions.</p> <p>Examples:</p> <ul style="list-style-type: none">- alternateBreakdownElement- functionalAndPhysicalBreakdownElementRelationship- accessPoint- relatedContract <p><i>Note: The related breakdown elements do not need to be used in the same breakdown, ie it can be used to establish the relationship between a breakdown element in a functional breakdown and a breakdown element in a physical breakdown.</i></p>	BreakdownElementRelationship	Breakdown Structure
breakdownElementRevisionCreationDate	DateType	breakdownElementRevisionCreationDate defines the date when the breakdown element revision was created.	BreakdownElementRevision	Breakdown Structure
breakdownElementRevisionIdentifier	IdentifierType	breakdownElementRevisionIdentifier is a string of characters used to uniquely identify a BreakdownElementRevision and to differentiate it from other BreakdownElementRevisions.	BreakdownElementRevision	Breakdown Structure
breakdownElementRevisionStatus	ClassificationType	breakdownElementRevisionStatus is a classification that identifies the maturity of a BreakdownElementRevision.	BreakdownElementRevision	Breakdown Structure
breakdownElementStructureRelationshipType	ClassificationType	breakdownElementStructureRelationshipType is a classification that identifies the type of restriction or association between two BreakdownElementRevisions.	BreakdownElementStructureRelationship	Breakdown Structure
breakdownRevisionCreationDate	DateType	breakdownRevisionCreationDate defines the date when the breakdown revision was created.	BreakdownRevision	Breakdown Structure
breakdownRevisionIdentifier	IdentifierType	breakdownRevisionIdentifier is a string of characters used to uniquely identify a BreakdownRevision and to differentiate it from other BreakdownRevisions.	BreakdownRevision	Breakdown Structure



Data element	Type	Definition	Class/Interface	UoF
breakdownRevisionStatus	ClassificationType	breakdownRevisionStatus is a classification that identifies the maturity of a BreakdownRevision.	BreakdownRevision	Breakdown Structure
breakdownType	ClassificationType	breakdownType is a classification that identifies the type of breakdown for the product. Examples: <ul style="list-style-type: none">- physicalBreakdown- functionalBreakdown- systemBreakdown- ASDSystemHardwareBreakdown- zonalBreakdown- familyTreeBreakdown- aggregatedBreakdown	Breakdown	Breakdown Structure
candidateItemAnalysisActivityDate	DateType	candidateItemAnalysisActivityDate defines the date for the latest update of the candidate item analysis activity information.	CandidateItemAnalysisActivity	LSA Candidate Analysis Activity
candidateItemAnalysisActivityIndicator	ClassificationType	candidateItemAnalysisActivityIndicator is a classification that identifies whether the candidate item analysis activity is selected for a specific LSA candidate. Examples: <ul style="list-style-type: none">- selected- rejected- open (not yet decided)	CandidateItemAnalysisActivity	LSA Candidate Analysis Activity
candidateItemAnalysisActivityRationale	DescriptorType	candidateItemAnalysisActivityRationale is a phrase that gives more information on the reason for the selection/non selection of the candidate item analysis activity.	CandidateItemAnalysisActivity	LSA Candidate Analysis Activity

Applicable to: All

S3000L-A-22-00-0000-00A-040A-A

Chap 22

Data element	Type	Definition	Class/Interface	UoF
candidateItemAnalysisActivityStatus	ClassificationType	candidateItemAnalysisActivityStatus a classification that identifies the progression of the candidate item analysis activity. Examples: <ul style="list-style-type: none">- notStarted- ongoing- completed	CandidateItemAnalysisActivity	LSA Candidate Analysis Activity
changeRequestDescription	DescriptorType	changeRequestDescription is a phrase that gives more information on a desired product design change.	ChangeRequest	LSA Candidate Task Requirement
changeRequestIdentifier	IdentifierType	changeRequestIdentifier is a string of characters used to uniquely identify a ChangeRequest and to differentiate it from other ChangeRequests.	ChangeRequest	LSA Candidate Task Requirement
circuitBreakerIdentifier	IdentifierType	circuitBreakerIdentifier is a string of characters used to uniquely identify a CircuitBreaker and to differentiate it from other CircuitBreakers.	CircuitBreaker	Task
circuitBreakerName	DescriptorType	circuitBreakerName is a word or phrase by which the circuit breaker is known and can be easily referenced.	CircuitBreaker	Task
circuitBreakerType	ClassificationType	circuitBreakerType is a classification that defines type of circuit breaker. Examples: <ul style="list-style-type: none">- electronicCircuitBreaker ('eltro' in S1000D)- electroMechanicCircuitBreaker ('elmec' in S1000D)- dummyCircuitBreaker ('clip' in S1000D)	CircuitBreaker	Task

Data element	Type	Definition	Class/Interface	UoF
circuitBreakerState	ClassificationType	<p>circuitBreakerState is a classification which defines the state that the circuit breaker must be in after the accomplishment of the subtask.</p> <p>Examples:</p> <ul style="list-style-type: none"> - open - close - verifyOpen - verifyClose 	SubtaskCircuitBreakerSetting	Task
conditionInstanceDescription	DescriptorType	conditionInstanceDescription is a narrative statement of the meaning of the condition instance.	ConditionInstance	Applicability Statement
conditionInstanceIdentifier	IdentifierType	conditionInstanceIdentifier is a string of characters used to uniquely identify a ConditionInstance and to differentiate it from other ConditionInstances.	ConditionInstance	Applicability Statement
conditionInstanceName	DescriptorType	conditionInstanceName is a word or phrase by which the condition instance is known and can be easily referenced.	ConditionInstance	Applicability Statement
conditionTypeClassValue	ClassificationType	<p>conditionTypeClassValue is a classification that is valid for a specific ConditionType.</p> <p>Examples:</p> <ul style="list-style-type: none"> - pre/post (serviceBulletinConditionType) - ashore/afloat (ashoreOrAfloatConditionType) - arctic/desert (operationalEnvironmentConditionType) - docked/indoor/outdoor (maintenanceEnvironmentConditionType) 	ConditionTypeClassValue	Applicability Statement
conditionTypeDescription	DescriptorType	conditionTypeDescription is a narrative statement of the meaning of the condition type.	ConditionType	Applicability Statement

Data element	Type	Definition	Class/Interface	UoF
conditionTypeName	ClassificationType	conditionTypeName is a word by which the condition type is known and can be easily referenced. Examples: - serviceBulletinConditionType - ashoreOrAfloatConditionType - operationalEnvironmentConditionType - maintenanceEnvironmentConditionType	ConditionType	Applicability Statement
conditionTypePropertyValue	PropertyType	conditionTypePropertyValue is a property that is valid for a specific ConditionType.	ConditionTypePropertyValue	Applicability Statement
containedSubstanceJustificationDescription	DescriptorType	containedSubstanceJustificationDescription is a phrase that gives more information on the most important property for the function of the hardware part as designed, that the included substance has.	ContainedSubstance	Part Definition
contractedBlockOfSerializedItems	SerialNumberRange	contractedBlockOfSerializedItems identifies an interval of serialized items as known by the customer.	ContractedProductVariant	Project
contractIdentifier	IdentifierType	contractIdentifier is a string of characters used to uniquely identify a Contract and to differentiate it from other Contracts.	Contract	Project
contractRelationshipType	ClassificationType	contractRelationshipType is a classification that defines the type of relationship that is established between two contracts. Examples: - subcontract - relatedContract	ContractRelationship	Project

Data element	Type	Definition	Class/Interface	UoF
correctionFactor	double	correctionFactor is a value by which the original key performance indicator value must be corrected to meet a specific design or usage of the system related to specific conditions, eg. environment which causes additional stress to the system (sand, extreme temperatures, salty environment).	CorrectionFactor	LSA Candidate
correctionFactorDate	DateType	correctionFactorDate defines the date when the correction factor was identified.	CorrectionFactor	LSA Candidate
correctionFactorJustification	DescriptorType	correctionFactorJustification is a phrase that gives more information on the reason for the introduction of the correction factor.	CorrectionFactor	LSA Candidate
damageDescription	DescriptorType	damageDescription is a phrase that gives more information on the expected damage.	Damage	Special Event and Damage
damageFamily	ClassificationType	damageFamily is a classification that defines type of damage.	Damage	Special Event and Damage
dataModuleCode	IdentifierType	dataModuleCode is a string of characters used to uniquely identify an S1000DDataModule and to differentiate it from other S1000DDataModules. <i>Note: A dataModuleCode must be created in accordance with the rules defined in S1000D.</i>	S1000DDataModule	Document
dataModuleInfoname	DescriptorType	dataModuleInfoname is a word or phrase by which the data module is known and can be easily referenced. <i>Note: Equals S1000D element "Infoname".</i>	S1000DDataModule	Document
dataModuleIssueNumber	IdentifierType	dataModuleIssueNumber is a string of characters used to uniquely identify a data module issue (revision) and to differentiate it from other data module issues.	S1000DDataModuleIssue	Document
detectionMeanAlarmDescription	DescriptorType	detectionMeanAlarmDescription is a phrase that gives more information on the alarm as such.	DetectionMeanAlarm	LSA FMEA

Applicable to: All

S3000L-A-22-00-0000-00A-040A-A
Chap 22



Data element	Type	Definition	Class/Interface	UoF
detectionMeanDescription	DescriptorType	detectionMeanDescription is a phrase that gives more information on the mean that warns the user or maintainer that a failure has occurred.	DetectionMeanCapability	LSA FMEA
detectionMeanFalseAlarmPresentation	DescriptorType	detectionMeanFalseAlarmPresentation is a phrase that gives more information on how the alarm is presented to the user/maintainer.	DetectionMeanAlarm	LSA FMEA
detectionMeanFalseAlarmRate	PropertyType	detectionMeanFalseAlarmRate identifies the percentage (expressed as a decimal value) of all alarms that indicate a malfunction that cannot be verified by maintenance personnel performing follow-on tests.	DetectionMeanAlarm	LSA FMEA
detectionMeanType	ClassificationType	<p>detectionMeanType is a classification that defines the type of test that generates the symptom.</p> <p>Examples:</p> <ul style="list-style-type: none">- powerOnBuiltInTest (P-BIT), executed at start-up of the item- continuousBuiltInTest (C-BIT), executed periodically and automatically during the operation of the item, without any intervention from the operating crew- initiatedBuiltInTest (I-BIT), executed upon order from the operator or from the maintenance team- groundSupportEquipment	DetectionMeanCapability	LSA FMEA
directMaintenanceCost	PropertyType	directMaintenanceCost include shop maintenance man-hours, shop test man-hours, repair cost (incl. required material). Statistical values like MTBF (Mean Time Between Failure) and MTBUR (Mean Time Between Unscheduled Removal) are taken into consideration.	DirectMaintenanceCost	LSA Candidate

Data element	Type	Definition	Class/Interface	UoF
documentAssignmentRole	ClassificationType	documentAssignmentRole is a classification that identifies the role of a document being assigned to something. Examples: <ul style="list-style-type: none">- documentReference<ul style="list-style-type: none">- drawingReference- designDocumentReference- directive- source- verification	DocumentAssignment	Document
documentIdentifier	IdentifierType	documentIdentifier is a string of characters used to uniquely identify an ExternalDocument and to differentiate it from other ExternalDocuments.	ExternalDocument	Document
documentIssueDate	DateType	documentIssueDate defines the date when a specific issue (revision) of a document was issued.	ExternalDocumentIssue	Document
documentIssueldentifier	IdentifierType	documentIssueldentifier is a string of characters used to uniquely identify an ExternalDocumentIssue and to differentiate it from other ExternalDocumentIssues.	ExternalDocumentIssue	Document
documentLocation	DescriptorType	documentLocation is a phrase that gives more information on where to find a specific document (issue). <i>Note: May be a string that contains a hyperlink to the document (or specific issue of the document).</i>	ExternalDocument	Document
documentPortion	DescriptorType	documentPortion is a phrase that gives a reference to the portion of a document which is of interest in a specific usage.	DocumentAssignment	Document
documentTitle	DescriptorType	documentTitle is a word or phrase by which the document is known and can be easily referenced.	ExternalDocument	Document

Data element	Type	Definition	Class/Interface	UoF
documentType	ClassificationType	documentType is a classification that identifies categories of documents to which individual documents can belong. Examples: - technicalReport - standard - drawing	ExternalDocument	Document
downTime	PropertyType	downTime is the acceptable (maximum) down time (MDT), where down time is the duration where an item is non-operational.	DownTime	LSA Candidate
eventThresholdNumberOfEventOccurrences	int	eventThresholdNumberOfEventOccurrences is the number of event occurrences required to initiate the related threshold or trigger.	EventThresholdDefinition	Task Usage (Part 1)
failureModeDescription	DescriptorType	failureModeDescription is a phrase that gives more information on the manner in which a failure occurs.	FailureMode	LSA FMEA
failureModeDetectionAbilityDescription	DescriptorType	failureModeDetectionAbilityDescription is a phrase that gives more information on the ability to detect an identified failure mode. The description can include eg, a summary of the available detection means.	FailureMode	LSA FMEA
failureModeDetectionAbilityRating	DatedClassification	failureModeDetectionAbilityRating is a support classification defining the ability to detect an identified failure mode. Examples: - highLikelihood - moderatelyHighLikelihood - mediumLikelihood - moderatelyLowLikelihood - lowLikelihood <i>Note: Qualitative judgment made by the analyst.</i>	FailureMode	LSA FMEA
failureModeDetectionRate	PropertyType	failureModeDetectionRate defines the percentage of failure modes detected by the defined detection mean.	FailureModeDetection	LSA FMEA

Applicable to: All

S3000L-A-22-00-0000-00A-040A-A
Chap 22

Data element	Type	Definition	Class/Interface	UoF
failureModeEffect	DescriptorType	<p>failureModeEffect is a phrase that gives more information on the consequences that a failure mode can have on the local/next higher/end item operation, function, or status.</p> <p><i>Note: Local effect - The consequences of each postulated failure/damage mode affecting the LSA candidate must be described along with any second order effects that result. Potential conditions where the failure/damage of one item results in a change of the conditional failure probability or effect of a second item must be identified. It is possible for the "local effect" to be the failure/damage mode itself.</i></p> <p><i>Next higher effect - The consequences of each failure/damage mode affecting the next higher indenture level must be described.</i></p> <p><i>End effect - The effect of each failure/damage mode upon the essential functions(s) affecting system/equipment operating capability and mission completion capability must be determined. The end effect described may be the result of a double failure. For example, failure of a safety device may result in a catastrophic end effect only in the event that both the prime function goes beyond the limit for which the safety device is set, and the safety device fails.</i></p>	FailureModeEffect	LSA FMEA

Data element	Type	Definition	Class/Interface	UoF
failureModeEffectLevel	ClassificationType	<p>failureModeEffectLevel is classification that identifies the higher indenture level that will be affected by the failure mode under consideration.</p> <p>Examples:</p> <ul style="list-style-type: none"> - nextHigherEffect: These effects concentrate on the impact a failure/damage mode has on the operation and function of the items in the next higher indenture level above the LSA candidate under consideration. - endEffect: End effects evaluate and define the total effect a failure/damage mode has on the operation, function, or status of the uppermost system. 	HigherFailureModeEffect	LSA FMEA
failureModeFailureRate	PropertyType	<p>failureModeFailureRate is the total number of failures within a population of the item under analysis divided by the total functional life of the population during the measurement interval.</p> <p><i>Note: The definition holds for time, rounds, miles, cycles or other measures of life units.</i></p>	FailureMode	LSA FMEA
failureModelIdentifier	IdentifierType	failureModelIdentifier is a string of characters used to uniquely identify a FailureMode and to differentiate it from other FailureModes.	FailureMode	LSA FMEA
failureModelIsolationRate	PropertyType	failureModelIsolationRate defines the percentage of detected failures that are associated without ambiguity to the failure of an identified item or component.	FailureMode	LSA FMEA

Data element	Type	Definition	Class/Interface	UoF
failureModeLocalizationAbilityDescription	DescriptorType	<p>failureModeLocalizationAbilityDescription is a phrase that gives more information on the ability to localize an identified failure mode. The description can include a summary of localization means available or required.</p> <p><i>Note: failureModeLocalizationAbilityDescription is generally complementary to failureModeDetectionAbilityRating.</i></p>	FailureMode	LSA FMEA
failureModeLocalizationAbilityRating	DatedClassification	<p>failureModeLocalizationAbilityRating is a support classification defining the ability to localize an identified failure mode.</p> <p>Examples:</p> <ul style="list-style-type: none"> - highLikelihood - moderatelyHighLikelihood - mediumLikelihood - moderatelyLowLikelihood - lowLikelihood <p><i>Note: Qualitative judgment made by the analyst.</i></p>	FailureMode	LSA FMEA
failureRate	.PropertyType	<p>failureRate is the total number of failures within a population of an LSA candidate item divided by the total functional life of the population during the measurement interval.</p> <p><i>Note: The definition holds for time, rounds, miles, cycles or other measures of life units.</i></p> <p><i>Note: Ref GEIA-0007, failure_rate_type</i></p>	FailureRate	LSA Candidate
failuresPerOperatingHour	PropertyParams	failuresPerOperatingHour is the failure rate expressed in failures per operating hour.	FailuresPerOperatingHour	LSA Candidate
fixedResourceMarker	boolean	fixedResourceMarker is a classification that identifies that no other resource than the specified is allowed to be used in the task/subtask.	TaskResource	Task Resources

Data element	Type	Definition	Class/Interface	UoF
functionalFailureDescription	DescriptorType	functionalFailureDescription is a phrase that gives more information on a possible functional failure identified during preventive maintenance analysis (PMA).	FunctionalFailure	LSA Candidate Task Requirement
functionalFailureEffectCriticality	ClassificationType	functionalFailureEffectCriticality is a classification that identifies the most serious impact that the functional failure will have on the end item.	FunctionalFailure	LSA Candidate Task Requirement
hardwareElementRepairability	ClassificationType	hardwareElementRepairability is a design classification that indicates whether a part is repairable from a technical standpoint, independent of customer maintenance concepts. Examples: - Repairable - partiallyRepairable - expendable - notApplicable	HardwareElementRevision	Breakdown Element Realization
hardwareElementRepairabilityStrategy	ClassificationType	hardwareElementRepairabilityStrategy is a support classification that defines the repairability strategy chosen for a specific customer and maintenance concept. <i>Note: Repairability strategy can include information on which maintenance level the repair is to be performed.</i>	HardwareElementRevision	Breakdown Element Realization
hardwareElementFunctionalReplaceability	ClassificationType	hardwareElementFunctionalReplaceability is a design classification that indicates whether a part is replaceable at its functional location defined from a technical standpoint, independent of customer maintenance concepts. Examples: - Replaceable - notReplaceable - notApplicable	HardwareElementRevision	Breakdown Element Realization

Data element	Type	Definition	Class/Interface	UoF
hardwareElementFunctionalReplaceabilityStrategy	ClassificationType	<p>hardwareElementFunctionalReplaceabilityStrategy is a support classification that defines the replaceability strategy chosen for a specific customer and maintenance concept. Replaceability strategy can include information about the maintenance level, at which the replacement task is to be performed.</p> <p>Examples:</p> <ul style="list-style-type: none"> - lineReplaceable - shopReplaceable 	HardwareElementRevision	Breakdown Element Realization
hardwareElementStructuralIndicator	ClassificationType	<p>hardwareElementStructuralIndicator is a design classification that defines how important the breakdown element is to structural integrity.</p> <p>Examples:</p> <ul style="list-style-type: none"> - structuralSignificantItem (SSI) - structuralItem (SI) - structuralDetail (SD) - notApplicable (N/A) <p><i>Note 1: A Structure Significant Item (SSI) is an item which was identified by any selection process coming from a Scheduled Maintenance Analysis Procedure like MSG-3 or S4000P. For this item, a Scheduled Maintenance Analysis (SMA) will be performed.</i></p> <p><i>Note 2: A Structural Item (SI) is part of the systems bodywork.</i></p> <p><i>Note 3: A Structural Detail (SD), also named Significant Detail in the S4000P, is a limited area of an SSI or a local spot being also part of the whole SSI. In contrast to an SSI, the SD is not identified by an own identifier (eg an own part number). Therefore these SD's must be defined and documented on a respective SSI drawing. For identification within any breakdown structure, an additional artificial breakdown element identifier can be addressed.</i></p>	HardwareElementRevision	Breakdown Element Realization

Applicable to: All

S3000L-A-22-00-0000-00A-040A-A

Chap 22

Data element	Type	Definition	Class/Interface	UoF
hardwareElementType	ClassificationType	hardwareElementType is a design classification that identifies further specialization of a hardware element. Examples: <ul style="list-style-type: none">- equipment- accessPoint<ul style="list-style-type: none">- door- panel- electricalPanel	HardwareElement	Breakdown Element Realization
hardwarePartConsumptionRate	PropertyType	hardwarePartConsumptionRate is a support characteristic which defines the number of times that the hardware part is replaced in 100 repairs of the next higher assembly.	HardwarePartAsDesigned SupportData	Part Definition
hardwarePartElectromagneticIncompatible	boolean	hardwarePartElectromagneticIncompatible is a design classification which identifies the ability of an electrical equipment to function satisfactorily in its electromagnetic environment without inadmissibly influencing this environment to which also other equipment belongs.	HardwarePartAsDesigned DesignData	Part Definition
hardwarePartElectromagneticSensitive	boolean	hardwarePartElectromagneticSensitive is a design classification which identifies electronic components subject to catastrophic failure, major characteristic change or performance degradation from influences of electromagnetic fields.	HardwarePartAsDesigned DesignData	Part Definition
hardwarePartElectrostaticSensitive	boolean	hardwarePartElectrostaticSensitive is a design classification which identifies electronic components subject to catastrophic failure, major characteristic change or performance degradation from influences of electrostatic fields and/or electrical charges.	HardwarePartAsDesigned DesignData	Part Definition



Data element	Type	Definition	Class/Interface	UoF
hardwarePartEnvironmentalAspectInUseClass	ClassificationType	<p>hardwarePartEnvironmentalAspectInUseClass is a support classification which identifies environment aspects that must to be considered during use of the hardware part.</p> <p>Examples:</p> <ul style="list-style-type: none">- harmfulToEnvironment- acidification- dangerousForOzoneLayer- greenhouseEffect- materialWaste- energyRegainingByBurning- materialRecycling	HardwarePartAsDesigned SupportData	Part Definition
hardwarePartEnvironmentalAspectPlannedDisposalClass	ClassificationType	<p>hardwarePartEnvironmentalAspectPlannedDisposalClass is a support classification which identifies environment aspects that must to be considered during planned disposal of the hardware part.</p> <p>Examples:</p> <ul style="list-style-type: none">- harmfulToEnvironment- acidification- dangerousForOzoneLayer- greenhouseEffect- materialWaste- energyRegainingByBurning- materialRecycling	HardwarePartAsDesigned SupportData	Part Definition
hardwarePartFitmentRequirement	ClassificationType	<p>hardwarePartFitmentRequirement is a design classification which identifies that an item cannot be fitted in its 'as supplied' state but must undergo some operation before, or during, installation.</p> <p>Examples:</p> <ul style="list-style-type: none">- needs drilling, reaming or trimming during fitting, normally carried out at Organizational or Intermediate Level- needs major repair facilities for fitment, normally carried out at Depot Level or Industrial Maintenance Organization	HardwarePartAsDesigned DesignData	Part Definition

Applicable to: All

S3000L-A-22-00-0000-00A-040A-A

Chap 22

Data element	Type	Definition	Class/Interface	UoF
hardwarePartHazardousClass	ClassificationType	<p>hardwarePartHazardousClass is a design classification which identifies a design part as capable of posing a significant risk to health, safety or property during transportation, handing or storage.</p> <p><i>Note: The Substance Identification Numbers listed in Chapter 2 of the United Nations Recommendations on the Transport of Dangerous Goods ST/SG/AC.10/1/Rev5, must be used to enable integration with other ASD specifications.</i></p>	HardwarePartAsDesigned DesignData	Part Definition
hardwarePartLogisticsCategory	ClassificationType	<p>hardwarePartLogisticsCategory is a support classification that defines the role of a hardware part as designed in the context of product support.</p> <p>Examples:</p> <ul style="list-style-type: none"> - expendable - repairable - consumable - disposable - material - spare - supply - supportEquipment 	HardwarePartAsDesigned SupportData	Part Definition
hardwarePartMagneticSensitive	boolean	hardwarePartMagneticSensitive is a design classification which identifies electronic components subject to catastrophic failure, major characteristic change or performance degradation from influences of magnetic fields.	HardwarePartAsDesigned DesignData	Part Definition



Data element	Type	Definition	Class/Interface	UoF
hardwarePartMaintenanceStart	ClassificationType	hardwarePartMaintenanceStart is a support classification which defines when maintenance scheduling must be initiated for a realized part. Examples: <ul style="list-style-type: none">- maintenanceStartAtProduction- maintenanceStartAtDelivery- maintenanceStartAtInstallationInAssembly- maintenanceStartAtInstallationInEndItem	HardwarePartAsDesigned SupportData	Part Definition
hardwarePartOperationalAuthorizedLife	AuthorizedLife	hardwarePartOperationalAuthorizedLife is a design characteristic which identifies the maximum usage limit for which an item may be operated, and upon reaching this limit, any further usage of the item must be re-authorized.	HardwarePartAsDesigned DesignData	Part Definition
hardwarePartRadiationSensitive	boolean	hardwarePartRadiationSensitive is a design classification which identifies electronic components subject to catastrophic failure, major characteristic change or performance degradation from influences of radioactive radiation.	HardwarePartAsDesigned DesignData	Part Definition
hardwarePartRepairability	ClassificationType	hardwarePartRepairability is a support classification which defines whether the HardwarePartAsDesigned is repairable from a technical perspective (eg, a vendor/supplier standpoint) independent of customer maintenance concepts. Examples: <ul style="list-style-type: none">- repairable- partiallyRepairable- discardable- notApplicable	HardwarePartAsDesigned SupportData	Part Definition
hardwarePartRepairabilityStrategy	ClassificationType	hardwarePartRepairabilityStrategy is a support classification which defines the repairability strategy chosen for a specific customer and maintenance concept. Repairability strategy can include information on which maintenance level the repair is to be performed.	HardwarePartAsDesigned SupportData	Part Definition

Applicable to: All

S3000L-A-22-00-0000-00A-040A-A

Chap 22

Data element	Type	Definition	Class/Interface	UoF
hardwarePartScrapRate	PropertyType	hardwarePartScrapRate is a support characteristic of a hardware part which defines the percentage of repairable units which, when removed from service, will be found to be beyond economic repair and therefore have to be scrapped.	HardwarePartAsDesigned SupportData	Part Definition
hardwarePartWasteProductsInUseDisposalDescription	DescriptorType	hardwarePartWasteProductsInUseDisposalDescription is a phrase that gives more information on how waste products for an individual part need to be managed when the part is disposed of according to the procedure during use or after being used.	HardwarePartAsDesigned SupportData	Part Definition
hardwarePartWasteProductsPlannedDisposalDescription	DescriptorType	hardwarePartWasteProductsPlannedDisposalDescription is a phrase that gives more information on how waste products need to be managed when the entire population of a part is disposed of according to the procedure for planned disposal.	HardwarePartAsDesigned SupportData	Part Definition
informationCode	ClassificationType	informationCode is a classification which identifies type of task according to S1000D. <i>Note: S1000D information codes must be used to enable integration with other ASD specifications.</i>	TaskRevision SubtaskByDefinition	Task Task
keyPerformanceIndicatorMethod	DescriptorType	keyPerformanceIndicatorMethod is a phrase that gives more information on the method by which the key performance indicator value has been derived.	KeyPerformanceIndicator	LSA Candidate
keyPerformanceIndicatorPercentile	PropertyType	keyPerformanceIndicatorPercentile is the percentage of all occurrences related to a specified key performance indicator that must be within the limit of the value defined for the key performance indicator. <i>Note: A customer requirement is that 98% of all replacement tasks must be performed below a specified value of two hours (= maximum replacement time).</i>	KeyPerformanceIndicator	LSA Candidate



Data element	Type	Definition	Class/Interface	UoF
keyPerformanceIndicatorStatus	ClassificationType	keyPerformanceIndicatorStatus is a classification defining the status of acceptance of the recorded value for the key performance indicator. Examples: <ul style="list-style-type: none">- preliminary- accepted- released	KeyPerformanceIndicator	LSA Candidate
lifeCycleCost	PropertyType	lifeCycleCost is the sum of all recurring and non-recurring (one-time) costs over the full life cycle of the LSA candidate. <i>Note: Includes purchase price, installation cost, operating costs, maintenance and upgrade costs, and remaining (residual or salvage) value at the end of ownership or its useful life.</i>	LifeCycleCost	LSA Candidate
LSACandidateIndicator	ClassificationType	LSACandidateIndicator is a support classification which defines the depth of analysis that will be performed on the candidate item. Examples: <ul style="list-style-type: none">- fullCandidate, provide full scale of selected LSA information applicable to the related item- partialCandidate, provide a partial scale of selected LSA information applicable to the related item, eg only remove and install information because of the need to gain access to other items, but no repair information required, because the item is a discardable item- nonCandidate, no LSA required- open, not yet decided	LSACandidate	LSA Candidate
LSACandidateMaintenanceConcept	DescriptorType	LSACandidateMaintenanceConcept is a statement of maintenance considerations, constraints and strategy for the operational support that governs the maintenance levels and type of maintenance activities to be carried out for the LSA candidate under analysis.	LSACandidate	LSA Candidate

Applicable to: All

S3000L-A-22-00-0000-00A-040A-A

Chap 22



Data element	Type	Definition	Class/Interface	UoF
LSACandidateMaintenanceSolution	DescriptorType	LSACandidateMaintenanceSolution is a statement of maintenance activities and maintenance levels that has been decided for the LSA candidate under analysis.	LSACandidate	LSA Candidate
LSACandidateRationale	DescriptorType	LSACandidateRationale is a phrase that gives more information on the reason for 'full', 'partial' or 'non' LSA Candidate indicator selection.	LSACandidate	LSA Candidate
LSAFailureModeDistributionRating	ClassificationType	LSAFailureModeDistributionRating is a support classification that identifies the rating for the occurrence of an individual LSA failure mode in comparison to the entire population of LSA failure modes identified for the item under analysis. Examples: - moderatelyHighLikelihood - mediumLikelihood - moderatelyLowLikelihood - lowLikelihood - veryLowLikelihood	LSAFailureModeWith DistributionRating	LSA FMEA
LSAFailureModeDistributionRatio	PropertyType	LSAFailureModeDistributionRatio identifies the fraction of an individual LSA failure mode in relation to the entire population of LSA failure modes identified for the item under analysis.	LSAFailureModeWith DistributionRatio	LSA FMEA
maintenanceFreeOperatingPeriod	PropertyType	maintenanceFreeOperatingPeriod is the acceptable (minimum) maintenance free operating period, where maintenance free operating period is the interval in which no maintenance actions occur.	MaintenanceFreeOperating Period	LSA Candidate
maintenanceLevelCapabilityDescription	DescriptorType	maintenanceLevelCapabilityDescription is a phrase that gives more information on eg. defined capabilities in terms of personnel, availability of special facilities, time limits and environmental conditions to be assumed. These capabilities are the basis in determining the functions to be accomplished at the respective maintenance level.	MaintenanceLevel	Product Usage Context

Applicable to: All

S3000L-A-22-00-0000-00A-040A-A

Chap 22



Data element	Type	Definition	Class/Interface	UoF
maintenanceLevelIdentifier	IdentifierType	maintenanceLevelIdentifier is a string of characters used to uniquely identify a MaintenanceLevel and to differentiate it from other MaintenanceLevels.	MaintenanceLevel	Product Usage Context
maintenanceLevelName	DescriptorType	maintenanceLevelName is a word or phrase by which the maintenance level is known and can be easily referenced.	MaintenanceLevel	Product Usage Context
maintenanceLocationDescription	DescriptorType	maintenanceLocationDescription is a phrase that gives more information on eg. actual capabilities in terms of personnel, availability of special facilities, time limits and environmental conditions.	MaintenanceLocation	Product Usage Context
maintenanceLocationIdentifier	IdentifierType	maintenanceLocationIdentifier is a string of characters used to uniquely identify a MaintenanceLocation and to differentiate it from other MaintenanceLocations.	MaintenanceLocation	Product Usage Context
maintenanceLocationName	DescriptorType	maintenanceLocationName is a word or phrase by which the maintenance location is known and can be easily referenced.	MaintenanceLocation	Product Usage Context
maintenanceManHoursPerOperatingHour	PropertyType	maintenanceManHoursPerOperatingHour is the (maximum) acceptable maintenance man hours per operating hour, where maintenance man hours per operating hour is the ratio of maintenance man hours expended to the operating interval (as defined by the measurement base) of the system/equipment.	MaintenanceManHours PerOperatingHour	LSA Candidate



Data element	Type	Definition	Class/Interface	UoF
maintenanceSignificantOrRelevantIndicator	ClassificationType	<p>maintenanceSignificantOrRelevantIndicator is a support classification of a BreakdownElement as a candidate for maintenance as a result of scheduled maintenance analyses or failure modes and effects analyses.</p> <p><i>Note: A maintenance significant item is an item which was identified by any selection process coming from a scheduled maintenance analysis like MSG-3 or S4000P. For this type of item a scheduled maintenance task will be documented. A maintenance relevant item is an item which can be repaired or replaced as a result of failure or damage.</i></p>	BreakdownElementRevision	Breakdown Structure
meanTimeBetweenFailure	PropertyType	<p>meanTimeBetweenFailure is the (minimum) mean time between failure (MTBF), where the MTBF is the total operational life of a population of an LSA candidate divided by the total number of failures within the population during a particular measurement interval.</p> <p><i>Note: The definition holds for time, rounds, miles, events, or other measure of life units.</i></p>	MeanTimeBetweenFailure	LSA Candidate
meanTimeBetweenUnscheduledRemoval	PropertyType	<p>meanTimeBetweenUnscheduledRemoval is the minimum mean time between unscheduled removal, where mean time between unscheduled removal is the total number of operational units (eg, miles, rounds, hours) divided by the total number of items removed from that system during a stated period of time.</p> <p><i>Note: This term is defined to exclude removals either being scheduled or performed to facilitate other maintenance and removals for product improvement.</i></p>	MeanTimeBetweenUnscheduledRemoval	LSA Candidate
meanTimeToRepair	PropertyType	meanTimeToRepair is the (maximum) mean time to repair (MTTR), where MTTR is the total elapsed time for corrective maintenance divided by the total number of corrective maintenance actions during a given period of time.	MeanTimeToRepair	LSA Candidate



Data element	Type	Definition	Class/Interface	UoF
nonConformanceDescription	DescriptorType	nonConformanceDescription is a phrase that gives more information on how the related item does not comply with its requirements.	NonConformanceData	Product Design Configuration
nonConformanceRestriction	DescriptorType	nonConformanceRestriction is a phrase that gives more information on how the use of the related item restricts the specified capabilities of the item in which it is contained.	NonConformanceData	Product Design Configuration
nonConformanceType	ClassificationType	nonConformanceType is a classification that identifies in which way the related item does not comply with its requirements. Examples: - concession (unintentional) - waiver (intentional)	NonConformanceData	Product Design Configuration
numberOfOperatingLocations	PropertyType	numberOfOperatingLocations is the number of locations which will receive and operate the contracted product variant.	OperatingLocationType	Product Usage Context
operatingLocationDescription	DescriptorType	operatingLocationDescription is a phrase that gives more information on eg, actual operating and environmental conditions.	OperatingLocation	Product Usage Context
operatingLocationIdentifier	IdentifierType	operatingLocationIdentifier is a string of characters used to uniquely identify an OperatingLocation and to differentiate it from other OperatingLocations.	OperatingLocation	Product Usage Context
operatingLocationName	DescriptorType	operatingLocationName is a word or phrase by which the operating location is known and can be easily referenced.	OperatingLocation	Product Usage Context
operatingLocationTypeDescription	DescriptorType	operatingLocationTypeDescription is a phrase that gives more information on eg, operating and environmental conditions to be assumed.	OperatingLocationType	Product Usage Context



Data element	Type	Definition	Class/Interface	UoF
operatingLocationTypeIdentifier	IdentifierType	operatingLocationTypeIdentifier is a string of characters used to uniquely identify an OperatingLocationType and to differentiate it from other OperatingLocationTypes.	OperatingLocationType	Product Usage Context
operatingLocationTypeName	DescriptorType	operatingLocationTypeName is a word or phrase by which the operating location type is known and can be easily referenced.	OperatingLocationType	Product Usage Context
operatingRequirementAtOperatingLocation	PropertyType	operatingRequirementAtOperatingLocation is the (annual) operating requirement per operating location and contracted product. <i>Note: Annual operating requirements may be measurement based, generally operating hours, but also other measurement bases can be possible.</i>	ContractedProductVariantAt OperatingLocation	Product Usage Context
operatingRequirementAtOperatingLocationType	PropertyType	operatingRequirementAtOperatingLocationType is the (annual) operating requirement per operating location type and contracted product. <i>Note: Annual operating requirements may be measurement based, generally operating hours, but also other measurement bases can be possible.</i>	ContractedProductVariantAt OperatingLocationType	Product Usage Context
organizationAssignmentRole	ClassificationType	organizationAssignmentRole is a classification that identifies the role of the organization being assigned to an object in the LSA dataset. Examples: - authorizingOrganization - designResponsibleOrganization - publishedByOrganization	OrganizationAssignment	Organization Assignment

Data element	Type	Definition	Class/Interface	UoF
organizationIdentifier	IdentifierType	organizationIdentifier is a string of characters used to uniquely identify an organization and to differentiate it from other organizations. Examples: - CAGECode - VATNumber	Organization	Project
organizationName	DescriptorType	organizationName is a word or phrase by which the organization is known and can be easily referenced.	Organization	Project
packagedTask	boolean	packagedTask identifies if the task is created in order to group a set of previously defined tasks.	RectifyingTask	Task
partDemilitarizationClass	DatedClassification	partDemilitarizationClass is a controlled item classification defining special measures to be taken when a part is being disposed of, eg. render them useless for military purposes or destroy any indications of military purposes or performance characteristics. Examples (S2000M): - demilitarization not required - trade security controls (TSC) required at disposal - remove and/or demilitarize installed key point(s) - demilitarize by mutilation - demilitarization to be furnished by the MoD or national demilitarization program office - demilitarization instructions to be furnished by item/technical manager - demilitarization required prior to transfer of item to national reutilization and disposition offices - security classified Item	PartAsDesignedControlledItemData	Part Definition

Data element	Type	Definition	Class/Interface	UoF
partIdentifier	IdentifierType	<p>partIdentifier is a string of characters that are unique to the issuing organization which is used to designate a PartAsDesigned and to differentiate it from other designed parts.</p> <p><i>Note: Includes manufacturer's part, drawing, model, type, or source controlling numbers.</i></p>	PartAsDesigned	Part Definition
partMaturityClass	DatedClassification	<p>partMaturityClass is a support classification defining the maturity of the part in order to determine the certainty by which the parts characteristics can be valued.</p> <p>Examples:</p> <ul style="list-style-type: none"> - newDeveloped - majorModificationOfExistingItem - moderateModificationOfExistingItem - COTSItem - customerItem 	PartAsDesignedSupportData	Part Definition
partName	DescriptorType	partName is a word or phrase by which the designed part is known and can be easily referenced.	PartAsDesigned	Part Definition
partObsolescenceRiskAssessment	DescriptorType	partObsolescenceRiskAssessment is a support characteristic describing the risk associated with loss of the part in the supply chain due to factors such as termination of manufacturing, replacement with a different model, etc.	PartAsDesignedSupportData	Part Definition
partsListEntryIdentifier	IdentifierType	partsListEntryIdentifier is a string of characters providing a relative position within the parts list.	PartAsDesignedPartsListEntry	Part Definition
partsListRevisionIdentifier	IdentifierType	partsListRevisionIdentifier is a string of characters used to uniquely identify a revision of a PartAsDesignedPartsList partsListType and to differentiate it from other PartAsDesignedPartsList revisions of the same partsListType.	PartAsDesignedPartsList	Part Definition

Data element	Type	Definition	Class/Interface	UoF
partsListType	ClassificationType	partsListType is a classification which identifies the context and intended use of the list of parts.	PartAsDesignedPartsList	Part Definition
partSpecialHandlingRequirement	DescriptorType	partSpecialHandlingRequirement is a phrase that gives more information on requirements for special handling of the part.	PartAsDesignedDesignData	Part Definition
physicalReplaceability	ClassificationType	<p>physicalReplaceability is a classification which identifies whether a part is interchangeable in a specific assembly.</p> <p>Examples:</p> <ul style="list-style-type: none"> - interchangeable (Interchangeable items must be capable of being readily installed, removed, or replaced without alteration misalignment or damage to items being installed or to adjoining items or structure) - replaceable (Replaceable items require alterations of the items in addition to the normal application and methods of attachment. Such alterations may include drilling, reaming, cutting, filing, trimming, bending, shaping, etc). - notApplicable <p><i>Note: This classification is done from a technical standpoint (ie, a vendor/supplier standpoint) and is independent of customer maintenance concepts.</i></p>	PartAsDesignedPartsListEntry	Part Definition
physicalReplaceabilityStrategy	ClassificationType	<p>physicalReplaceabilityStrategy is a classification which identifies the replaceability strategy chosen for a specific customer and maintenance concept. Replaceability strategy can include information about the maintenance level, at which the replacement task is to be performed.</p> <p>Examples:</p> <ul style="list-style-type: none"> - lineReplaceable - shopReplaceable 	PartAsDesignedPartsListEntry	Part Definition



Data element	Type	Definition	Class/Interface	UoF
productIdentifier	IdentifierType	productIdentifier is a string of characters used to uniquely identify a Product and to differentiate it from other Products. Examples: <ul style="list-style-type: none">- endItemAcronymCode- modelIdentificationCode	Product	Project
productName	DescriptorType	productName is a word or phrase by which the product is known and can be easily referenced.	Product	Project
productServiceLife	PropertyType	productServiceLife defines the number of years that the LSA candidate is expected to be in service. <i>Note: Other measures than years can be used.</i>	ProductServiceLife	LSA Candidate
productUsagePhase	ClassificationType	productUsagePhase is a classification which identifies phases during the products in-service phase. Examples: <ul style="list-style-type: none">- operation<ul style="list-style-type: none">- takeOff- flight- landing- maintenance- storage- transportation	ProductUsagePhase	Special Event and Damage
productVariantIdentifier	IdentifierType	productVariantIdentifier is a string of characters used to uniquely identify a ProductVariant and to differentiate it from other ProductVariants. Examples: <ul style="list-style-type: none">- usableOnCode- modelIdentificationCode	ProductVariant	Project
productVariantName	DescriptorType	productVariantName is a word or phrase by which the product variant is known and can be easily referenced.	ProductVariant	Project



Data element	Type	Definition	Class/Interface	UoF
projectIdentifier	IdentifierType	projectIdentifier is a string of characters used to uniquely identify a Project and to differentiate it from other Projects.	Project	Project
projectName	DescriptorType	projectName is a word or phrase by which the project is known and can be easily referenced.	Project	Project
publicationModuleCode	IdentifierType	publicationModuleCode is a string of characters used to uniquely identify an S1000DPublicationModule and to differentiate it from other S1000DPublicationModules. <i>Note: A publicationModuleCode must be created in accordance with the rules defined in S1000D.</i>	S1000DPublicationModule	Document
publicationModuleTitle	DescriptorType	publicationModuleTitle is a word or phrase by which the publication module is known and can be easily referenced.	S1000DPublicationModule	Document
publicationModuleIssueNumber	IdentifierType	publicationModuleIssueNumber is a string of characters used to uniquely identify a publication module issue (revision) and to differentiate it from other publication module issues.	S1000DpublicationModule Issue	Document
quantityOfChildElement	PropertyType	quantityOfChildElement is the amount that a child element is used in its parent element within a parent/child relationship. <i>Note: If no value is given, it must be interpreted as value "1" with a unit of "each". For as required amounts, the text property is used with "As Required" or other text as appropriate.</i>	BreakdownElementStructure PartAsDesignedPartsListEntry	Breakdown Structure Part Definition
quantityOfContainedSubstance	PropertyType	quantityOfContainedSubstance is the amount of the substance included in a HardwarePartAsDesigned	ContainedSubstance	Part Definition
quantityOfContractedProductVariant	PropertyType	quantityOfContractedProductVariant is the amount of a product variant included in a contract.	ContractedProductVariant	Project
quantityOfProductVariantAtOperatingLocation	PropertyType	quantityOfProductVariantAtOperatingLocation is the number of serialized items, of a specific product variant, that is to be operated at a specific operating location.	ContractedProductVariantAt OperatingLocation	Product Usage Context

Applicable to: All

S3000L-A-22-00-0000-00A-040A-A

Chap 22

Data element	Type	Definition	Class/Interface	UoF
quantityOfProductVariantAtOperatingLocationType	.PropertyType	quantityOfProductVariantAtOperatingLocationType is the number of serialized items, of a specific product variant, that is to be operated at a specific operating location type.	ContractedProductVariantAtOperatingLocationType	Product Usage Context
referenceDesignator	IdentifierType	<p>referenceDesignator is a string of characters used to uniquely identify the location of a child element within a parent element.</p> <p><i>Note: referenceDesignator serves as a cross reference between parts contained in wiring diagrams, hydraulic systems etc. and the Illustrated Parts Catalogue (IPC). Letters, numbers or symbols are used to uniquely identify and locate discrete units, portions thereof and basic parts of a specific component.</i></p>	BreakdownElementStructure PartAsDesignedPartsListEntry	Breakdown Structure Part Definition
remarkText	DescriptorType	remarkText is a phrase that provides the user with information that is helpful but does not belong to the immediate subject.	Remark	Remark
remarkType	ClassificationType	<p>remarkType is a classification that defines the purpose of the remark.</p> <p>Examples:</p> <ul style="list-style-type: none"> - comment - remark 	Remark	Remark
replacementTime	PropertyParams	replacementTime is the duration of the replacement of a (eg faulty) component within any technical system by another (eg new) component.	ReplacementTime	LSA Candidate
resourceSpecificationDescription	DescriptorType	resourceSpecificationDescription is a phrase that gives more information on the specified resource needed to perform one or many tasks.	ResourceSpecification	Task Resources
resourceSpecificationIdentifier	IdentifierType	resourceSpecificationIdentifier is a string of characters used to uniquely identify a ResourceSpecification and to differentiate it from other ResourceSpecifications.	ResourceSpecification	Task Resources

Data element	Type	Definition	Class/Interface	UoF
resourceSpecificationName	DescriptorType	resourceSpecificationName is a word or phrase by which the resource specification is known and can be easily referenced.	ResourceSpecification	Task Resources
samplingMethodDescription	DescriptorType	samplingMethodDescription is a phrase that gives more information on the sampling method used.	SamplingDefinition	Task Usage (Part 1)
samplingRatio	double	samplingRatio is the number of samples over the total population expressed as a ratio, eg, 0,1 (equals 10%). <i>Note: The value must be represent as decimal, even though it should have been a fraction.</i>	SamplingDefinitionByRatio	Task Usage (Part 1)
samplingValue	PropertyType	samplingValue the number of samples over the total population expressed as a value, eg, 10 aircraft.	SamplingDefinitionByValue	Task Usage (Part 1)
scheduledMaintenanceInterval	PropertyType	scheduledMaintenanceInterval is the (minimum) number of operational units (eg rounds, miles, hours) between scheduled maintenance.	ScheduledMaintenanceInterval	LSA Candidate
securityClass	ClassificationType	securityClass is a classification that defines the level of confidentiality. Examples: - unclassified - restricted - confidential - secret - topSecret - companyConfidential	SecurityClass	Security Classification
shopProcessingTime	PropertyType	shopProcessingTime is the duration from the start of the repair activities in the repair shop until the closing of the repair procedure without considering any shipping and delay times.	ShopProcessingTime	LSA Candidate

Data element	Type	Definition	Class/Interface	UoF
skillCode	IdentifierType	skillCode is a string of characters used to uniquely identify a Skill and to differentiate it from other Skills.	Skill	Task Resources
skillLevelDescription	DescriptorType	skillLevelDescription is a phrase that gives more information on the identified skill level.	SkillLevel	Task Resources
skillLevelName	ClassificationType	skillLevelName is a classification which identifies competency level. Examples: - advanced - intermediate - basic	SkillLevel	Task Resources
softwareElementModificationFrequency	PropertyType	softwareElementModificationFrequency is a support characteristic defining the expected frequency that the SoftwarePart will be modified.	SoftwareElementRevision	Breakdown Element Realization
softwareElementSize	PropertyType	softwareElementSize is a design characteristic defining the expected size of the SoftwarePart for this particular SoftwareElementRevision. Examples: - kiloByte - megaByte - gigaByte - teraByte	SoftwareElementRevision	Breakdown Element Realization
softwareElementType	ClassificationType	softwareElementType is a design classification that identifies further specialization of a software element. Examples: - loadable - embedded - distributed	SoftwareElement	Breakdown Element Realization

Data element	Type	Definition	Class/Interface	UoF
softwarePartSize	PropertyType	softwarePartSize is the actual size of a software part. Examples: - kiloByte - megaByte - gigaByte - teraByte	SoftwarePartAsDesigned DesignData	Part Definition
softwareType	ClassificationType	softwareType is a design classification which identifies if the software and/or data can be loaded/unloaded to/from the related hardware part. Examples: - loadable - embedded - distributed	SoftwarePartAsDesigned DesignData	Part Definition
specialEventDescription	DescriptorType	specialEventDescription is a phrase that gives more information on the special event that can cause a related damage mode.	SpecialEvent	Special Event and Damage
specialEventGroup	ClassificationType	specialEventGroup is a classification used for the grouping of special events. Examples: - externalCause <ul style="list-style-type: none">- naturalPhenomenon<ul style="list-style-type: none">o meteorologicalo animal- humanImpact<ul style="list-style-type: none">o combato materialManeuver - internalCause <ul style="list-style-type: none">- internalDysfunction<ul style="list-style-type: none">o extensiveHeato extensiveVibration	SpecialEvent	Special Event and Damage



Data element	Type	Definition	Class/Interface	UoF
specialEventOccurrenceRate	PropertyType	specialEventOccurrenceRate is a quantification of how often a specific special event will occur.	QuantifiedSpecialEvent Occurrence	Special Event and Damage
specialEventOccurrenceRating	ClassificationType	specialEventOccurrenceRating is a classification that qualifies how often a specific special event will occur. Examples: <ul style="list-style-type: none">- extremelyUnlikely- remoteLikelihood- occasional- reasonablyProbable- frequent	RatedSpecialEvent Occurrence	Special Event and Damage
specialEventTitle	ClassificationType	specialEventTitle is a classification which identifies the title (name) by which a special event is known.	SpecialEvent	Special Event and Damage
substanceCharacteristicsRecordingDate	DateType	substanceCharacteristicsRecordingDate defines the date of the latest changes to the substance characteristics data.	SubstanceDefinition	Part Definition
substanceDescription	DescriptorType	substanceDescription is a phrase that gives more information on the substance used in one or many hardware parts.	SubstanceDefinition	Part Definition
substanceIdentifier	IdentifierType	substanceIdentifier is a string of characters, unique to the issuing organization used to designate the substance and to differentiate it from other similar substances. <i>Note: One source for substance identifiers can be Chemical Abstract Substance Registry Numbers (often referred to as a CAS Number). Refer to www.cas.org.</i>	SubstanceDefinition	Part Definition
substanceName	DescriptorType	substanceName is a word or phrase by which a substance is known and can be easily referenced.	SubstanceDefinition	Part Definition
substanceRiskDescription	DescriptorType	substanceRiskDescription is a phrase that describes risks associated with the substance.	SubstanceDefinition	Part Definition

Data element	Type	Definition	Class/Interface	UoF
substanceRiskFactor	ClassificationType	substanceRiskFactor is a classification which identifies possible risks associated with substance usage.	SubstanceDefinition	Part Definition
substanceUsageCategory	ClassificationType	substanceUsageCategory is a classification which identifies possible limitations in substance usage. Examples: - forbidden - authorizedWithLimitation - authorizedWithNoLimitation	SubstanceDefinition	Part Definition
subtaskAcceptanceParameterDescription	DescriptorType	subtaskAcceptanceParameterDescription is a phrase that describes the criteria that determines whether a subtask is completed.	SubtaskAcceptanceParameter	Task
subtaskAcceptanceParameterValue	PropertyType	subtaskAcceptanceParameterValue is the value (criteria) that must be fulfilled before the subtask can be ended.	SubtaskAcceptanceParameter	Task
subtaskDescription	DescriptorType	subtaskDescription is a phrase that gives more information on the subtask procedure.	SubtaskByDefinition	Task
subtaskDuration	PropertyType	subtaskDuration is the average time expended, regardless of the number of personnel working simultaneously, required for the performance of the subtask. <i>Note: subtaskDuration does not include time spent awaiting spares, support equipment, facilities or personnel (logistics delay time).</i>	SubtaskByDefinition	Task

Data element	Type	Definition	Class/Interface	UoF
subtaskEndItemObjectiveState	ClassificationType	<p>subtaskEndItemObjectiveState is a classification which defines the state that will exist after the accomplishment of the subtask.</p> <p>Examples:</p> <ul style="list-style-type: none"> - taskChecked - jacked/unjacked - safetyDeviceEstablished - electricalPowerEstablished <ul style="list-style-type: none"> - electricalPowerFromEngineEstablished - electricalPowerFromAPUEstablished - externalElectricalPowerEstablished - internalElectricalPowerEstablished - hydraulicPowerEstablished - airSupplyEstablished - fueled/defueled - waterSupplyEstablished - controlStatusEstablished 	SubtaskByDefinition	Task
subtaskIdentifier	IdentifierType	<p>subtaskIdentifier is a string of characters used to uniquely identify a Subtask and to differentiate it from other Subtasks.</p> <p><i>Note: A subtask is identified within the context of a task.</i></p>	Subtask	Task

Data element	Type	Definition	Class/Interface	UoF
subtaskMaintenanceLocation	ClassificationType	<p>subtaskMaintenanceLocation is a classification which indicates where the maintenance task will be carried out in terms of a product (end item or major assembly).</p> <p>Examples:</p> <ul style="list-style-type: none"> - "A" Information related to items installed on the Product - "B" Information related to items installed on a major assembly removed from the Product - "C" Information related to items on the bench. In this context, it does not matter, for example, whether an item has been removed from the Product. - "D" Information related to all locations A, B, and C <p><i>Source S1000D Item Location Code</i></p>	SubtaskByDefinition	Task
subtaskName	DescriptorType	subtaskName is a word or phrase by which a subtask is known and can be easily referenced.	SubtaskByDefinition	Task
subtaskRole	ClassificationType	<p>subtaskRole is a classification which identifies if the subtask is required for preparation, core, or close-up purposes.</p> <p>Examples:</p> <ul style="list-style-type: none"> - startup - core - coreNoRequiredConditions - closeup 	Subtask	Task
subtaskTimelineEvent	ClassificationType	<p>subtaskTimelineEvent is a classification which identifies the starting point for subtask under consideration in relation to the start or end point of the subtask playing the role of its predecessor.</p> <p>Examples:</p> <ul style="list-style-type: none"> - start - end 	SubtaskTimeline	Task



Data element	Type	Definition	Class/Interface	UoF
subtaskTimelineLag	PropertyType	subtaskTimelineLag is the time between the related subtask timeline event (start/end) and the start for the subtask under consideration. <i>Note: Time lining just using references to the start point for the entire task, can use the lag attribute to indicate the time relative to the starting point of the first subtask and set subtaskTimelineEvent = 'start'.</i>	SubtaskTimeline	Task
taskDuration	PropertyType	taskDuration is the average time expended, regardless of the number of personnel working simultaneously, required for the performance of a task, scheduled or unscheduled. <i>Note: taskDuration does not include time spent awaiting spares, support equipment, facilities or personnel (logistics delay time).</i> <i>Note: May be calculated from the durations of the subtasks.</i>	TaskRevision	Task
taskIdentifier	IdentifierType	taskIdentifier is a string of characters used to uniquely identify a Task and to differentiate it from other Tasks.	Task	Task
taskFacilityResourceQuantity	PropertyType	taskFacilityResourceQuantity is the quantity of the facility resource needed by a subtask, or aggregated per task.	TaskFacilityResource	Task Resources
taskFrequency	PropertyType	taskFrequency is the frequency of performance or occurrence of the task, expressed as the number of annual occurrences.	TaskFrequency	Task Usage (Part 1)
taskFrequencyCalculationMethod	DescriptorType	taskFrequencyCalculationMethod is a phrase that gives more information on the method used for calculating the task frequency.	TaskFrequency	Task Usage (Part 1)

Data element	Type	Definition	Class/Interface	UoF
taskMaterialResourceCategory	ClassificationType	<p>taskMaterialResourceCategory is a classification that defines the logistics role of the needed material resource in the context of the task/subtask.</p> <p>Examples:</p> <ul style="list-style-type: none"> - supportEquipment <ul style="list-style-type: none"> - safetyRelatedSupportEquipment - spare - supply 	TaskMaterialResource	Task Resources
taskMaterialResourceQuantity	PropertyType	taskMaterialResourceQuantity is the quantity of the material resource needed by a subtask, or aggregated per task.	TaskMaterialResource	Task Resources
taskName	DescriptorType	<p>taskName is a word or phrase by which a task is known and can be easily referenced.</p> <p><i>Note: The following rules must be followed to enable integration with other ASD specifications:</i></p> <p><i>taskName must use the information code name for the informationCode that is associated with the task.</i></p> <p><i>Additional qualifiers can be added to establish a unique taskName, if multiple tasks with the same informationCode is associated with one and the same breakdown element or part.</i></p>	TaskRevision	Task
taskNumberOfPersonnelResource	PropertyType	taskNumberOfPersonnelResource is the number of persons with the same role and skill code needed within one task/subtask.	TaskPersonnelResource	Task Resources



Data element	Type	Definition	Class/Interface	UoF
taskOperabilityImpact	ClassificationType	taskOperabilityImpact is a classification that indicates the operational status and mission readiness of the end item during the task. Examples: <ul style="list-style-type: none">- system inoperable during equipment maintenance- system operable during equipment maintenance- full mission capable- partial mission capable- not mission capable- turnaround	TaskRevision	Task
taskPersonnelResourceLabourTime	PropertyType	taskPersonnelResourceLabourTime is the time expended within a task/subtask per personnel resource.	TaskPersonnelResource	Task Resources
taskPersonnelResourceRole	ClassificationType	taskPersonnelResourceRole is a classification that defines the role of the personnel resource needed within a task/subtask. Examples: <ul style="list-style-type: none">- manA- manB- performer- supervisor- qualityAssurance	TaskPersonnelResource	Task Resources
taskPersonnelSafetyCriticality	ClassificationType	taskPersonnelSafetyCriticality is a classification that identifies the most serious health aspects that the performance of the task can pose on personnel performing the task.	TaskRevision	Task



Data element	Type	Definition	Class/Interface	UoF
taskProductIntegrityCriticality	ClassificationType	taskProductIntegrityCriticality is a classification that identifies whether the task is critical if failure to accomplish it would result in adverse effects on system reliability, efficiency, effectiveness, safety, or cost. A task will also be designated as critical whenever system design characteristics approach human limitations, and thereby, significantly increase the likelihood of degraded, delayed, or otherwise impaired mission performance.	TaskRevision	Task
taskRequirementAuthority	Organization	taskRequirementAuthority identifies the organization that is the authoritative source for the identified task requirement.	AuthorityDrivenTask Requirement	LSA Candidate Task Requirement
taskRequirementAuthoritySourceType	ClassificationType	taskRequirementAuthoritySourceType is a classification which indicates the source of an authority driven task requirement. Examples: - MSG3 - CMR - AD	AuthorityDrivenTask Requirement	LSA Candidate Task Requirement
taskRequirementRevisionChange Description	DescriptorType	taskRequirementRevisionChangeDescription is a phrase that gives more information on changes introduced between two revisions of a task requirement.	TaskRequirementRevision	LSA Candidate Task Requirement
taskRequirementDate	DateType	taskRequirementDate defines the date when the task requirement was recorded.	TaskRequirementRevision	LSA Candidate Task Requirement
taskRequirementDecision	ClassificationType	taskRequirementDecision is a classification which identifies the status for the task requirement. Examples: - accepted - rejected - deferred - realized	TaskRequirementRevision	LSA Candidate Task Requirement

Applicable to: All

S3000L-A-22-00-0000-00A-040A-A

Chap 22

Data element	Type	Definition	Class/Interface	UoF
taskRequirementDescription	DescriptorType	taskRequirementDescription is a phrase that gives more information on the maintenance requirements that have been determined as the outcome of eg. Reliability Centered Maintenance (RCM) analysis, LSA-FMEA or In-service feedback.	TaskRequirementRevision	LSA Candidate Task Requirement
taskRequirementIdentifier	IdentifierType	taskRequirementIdentifier is a string of characters used to uniquely identify a TaskRequirement and to differentiate it from other TaskRequirements.	TaskRequirement	LSA Candidate Task Requirement
taskRequirementRevisionIdentifier	IdentifierType	taskRequirementRevisionIdentifier is a string of characters used to uniquely identify a TaskRequirementRevision and to differentiate it from other TaskRequirementRevisions.	TaskRequirementRevision	LSA Candidate Task Requirement
taskRequirementSpecialResourceRequirement	DescriptorType	taskRequirementSpecialResourceRequirement is a phrase that gives more information on special (peculiar) resources needed for the performance of the required task.	TaskRequirementRevision	LSA Candidate Task Requirement
taskResourceDuration	PropertyType	taskResourceDuration is the average time that a resource is needed for the performance of a task, scheduled or unscheduled.	TaskResource	Task Resources
taskResourceRelationshipCategory	ClassificationType	taskResourceRelationshipCategory is a classification that identifies the type of relationship established between two resources needed within the same subtask. Examples: - uses (eg, maintainer uses a support equipment) - supervises (eg, manA supervises manB)	TaskResourceRelationship	Task Resources
taskRevisionChangeDescription	DescriptorType	taskRevisionChangeDescription is a phrase that gives more information on changes introduced between two revisions of a task.	TaskRevision	Task
taskRevisionIdentifier	IdentifierType	taskRevisionIdentifier is a string of characters used to uniquely identify a TaskRevision and to differentiate it from other TaskRevisions.	TaskRevision	Task

Applicable to: All

S3000L-A-22-00-0000-00A-040A-A
Chap 22



Data element	Type	Definition	Class/Interface	UoF
taskRevisionStatus	ClassificationType	taskRevisionStatus is a classification which identifies the progress on the definition of a task revision.	TaskRevision	Task
taskTotalLabourTime	PropertyType	taskTotalLabourTime is the total time expended within a task. Includes the labor time for all required personnel resources.	TaskRevision	Task
technologyBehaviourKnowledgeRating	DatedClassification	technologyBehaviourKnowledgeRating is a support classification that identifies the knowledge of behavior of the technology used, during normal use or caused by special event.	LSACandidateTechnologyBehaviourRating	Special Event and Damage
technologySensitivityRating	DatedClassification	technologySensitivityRating is a support classification that identifies possible damage sources during normal use or caused by special event.	LSACandidateTechnologyBehaviourRating	Special Event and Damage
thresholdValue	PropertyType	thresholdValue is the value that defines the limit (interval) for when a task is to be performed.	ParameterThresholdDefinition	Task Usage (Part 1)
timeLimitDescription	DescriptorType	timeLimitDescription is a phrase that gives more information on the limit (threshold and/or trigger) that initiates the performance of the associated task.	TimeLimit	Task Usage (Part 1)
timeLimitHarmonizationIndicator	boolean	timeLimitHarmonizationIndicator identifies if the time limit is the result from a task limit harmonization and/or packaging procedure. <i>Note: Such tasks are in general inspection or/and overhaul packages, documented against non-physical breakdown elements.</i>	TimeLimit	Task Usage (Part 1)
tradeName	ClassificationType	tradeName is a classification which identifies the type of occupation.	Trade	Task Resources



Data element	Type	Definition	Class/Interface	UoF
trainingMethod	ClassificationType	trainingMethod is a classification which identifies the method how an additional required training for a specific competence can be performed.. Examples: <ul style="list-style-type: none">- classroom training- computer based learning- further training with final test and certification- training on the job	AdditionalTrainingRequirement	Task Resources
warningCautionNoteDescription	DescriptorType	warningCautionNoteDescription is a phrase that gives more information on advices concerning safety, legal and health aspects.	WarningCautionNote	Task
warningCautionNoteIdentifier	IdentifierType	warningCautionNoteIdentifier is a string of characters used to uniquely identify a WarningCautionNote and to differentiate it from other WarningCautionNotes.	WarningCautionNote	Task
warningCautionNoteType	ClassificationType	warningCautionNoteType is a classification of the provided safety, legal and health aspects as a warning, caution or note. Examples: <ul style="list-style-type: none">- warning- caution- note	WarningCautionNote	Task
zoneElementDescription	DescriptorType	zoneElementDescription is a phrase that gives more information on the zone element under consideration.	ZoneElementRevision	Breakdown Zone Element
zoneElementType	ClassificationType	zoneElementType is a classification that identifies further specialization of a zone element. Examples: <ul style="list-style-type: none">- zone- workArea	ZoneElement	Breakdown Zone Element

Table 3 Attributes of the S3000L data types and compound attributes

Data element	Type	Definition	Class	UoF
authorizedLife	.PropertyType	authorizedLife is a characteristic which defines the maximum life limit for an item, and upon reaching this limit, any further usage of the item must be re-authorized.	AuthorizedLife	Compound Attributes
class	ClassificationType	class is a characteristic that represents the term that is used for classification.	DatedClassification	Compound Attributes
classificationDate	DateType	classificationDate is a characteristicMetadata which defines the date when the classification was determined.	DatedClassification	Compound Attributes
classifier	char	The word or code by which the term (class) is known. <i>Source: ISO 10303:239 ed.2, Product Life Cycle Support.</i>	ClassificationType	Data Types
dayComponent	int	The day element of the DateType expressed as a value between 1 and 31. <i>Source: ISO 10303:239 Product Life Cycle Support.</i>	DateType	Data Types
descriptorLanguage	ClassificationType	Classification that determines the language in which the descriptorText is written. <i>Source: OASIS PLCS DEXlib.</i>	DescriptorType	Data Types
descriptorProvidedBy	Organization	The organization that provided the descriptorText. <i>Source: OASIS PLCS DEXlib.</i>	DescriptorType	Data Types
descriptorProvidedDate	DateType	The date when the descriptorText was provided. <i>Source: OASIS PLCS DEXlib.</i>	DescriptorType	Data Types
descriptorText	char	Text that provides further information about the subject under consideration. <i>Source: OASIS PLCS DEXlib.</i>	DescriptorType	Data Types
identifier	char	The text that conveys the assigned identifier. <i>Source: ISO 10303:239 Product Life Cycle Support.</i>	IdentifierType	Data Types



Data element	Type	Definition	Class	UoF
identifierClassifier	ClassificationType	Classification that determines the type of identifier being defined. <i>Source: OASIS PLCS DEXlib.</i>	IdentifierType	Data Types
identifierSetBy	Organization	Defines the organization that "owns" the assigned identifier. <i>Source: OASIS PLCS DEXlib.</i>	IdentifierType	Data Types
lifeAuthorizingOrganization	Organization	lifeAuthorizingOrganization is a characteristicMetadata which defines the organization that authorized the maximum life limit.	AuthorizedLife	Compound Attributes
lowerBound	char	lowerBound is a characteristic that represents the first valid serial number.	SerialNumberRange	Compound Attributes
lowerLimitValue	double	The lower limit of a value range. <i>Source: ISO 10303:239 Product Life Cycle Support.</i>	ValueRangePropertyType	Data Types
lowerOffsetValue	double	The lower limit defined as the lower offset value from the nominal value (value + lower limit). <i>Source: ISO 10303:239 Product Life Cycle Support.</i>	ValueWithTolerances PropertyType	Data Types
monthComponent	int	The month element of the DateType expressed as a value between 1 and 12, where: <ul style="list-style-type: none">- 1 = January- 2 = February- 3 = March- 4 = April- 5 = May- 6 = June- 7 = July- 8 = August- 9 = September- 10 = October- 11 = November- 12 = December <i>Source: ISO 10303:239 Product Life Cycle Support.</i>	DateType	Data Types

Data element	Type	Definition	Class	UoF
nominalValue	double	Specifies the single value that is the base value for specifying the range. <i>Source: ISO 10303:239 Product Life Cycle Support.</i>	ValueWithTolerances PropertyType	Data Types
textValue	char	The string that is the element of representation. <i>Source: ISO 10303:239 Product Life Cycle Support.</i>	TextPropertyType	Data Types
unit	ClassificationType	The unit with which the quantity is expressed. <i>Source: ISO 10303:239 ed.2 Product Life Cycle Support.</i>	NumericalPropertyType	Data Types
upperBound	char	upperBound is a characteristic that represents the last valid serial number. <i>Note: If the value for upperBound is not specified, the range has no upper bound.</i>	SerialNumberRange	Compound Attributes
upperOffsetValue	double	The upper limit defined as the upper offset value from the nominal value (value + upper offset value). <i>Source: ISO 10303:239 Product Life Cycle Support.</i>	ValueWithTolerances PropertyType	Data Types
upperLimitValue	double	The upper limit of a value range. <i>Source: ISO 10303:239 Product Life Cycle Support.</i>	ValueRangePropertyType	Data Types
value	double	The value of the quantity. <i>Source: ISO 10303:239 Product Life Cycle Support.</i>	SingleValuePropertyType	Data Types
valueDetermination	ClassificationType	The method by which the value of the property has been determined. <i>Examples:</i> <ul style="list-style-type: none"> - calculated (the value has been calculated) - designed (the value represents a value intended by the design) - estimated (the value has been estimated) - measured (the value has been measured) - setPoint (the value is used as an initialization value) <i>Source: ISO 10303:239 ed.2 Product Life Cycle Support.</i>	PropertyType	Data Types

Applicable to: All

S3000L-A-22-00-0000-00A-040A-A

Chap 22

Data element	Type	Definition	Class	UoF
valueRecordingDate	DateType	The date when the property value was recorded. <i>Source: OASIS PLCS DEXlib.</i>	.PropertyType	Data Types
yearComponent	int	The year element of the DateType expressed as a value between 1 and 9999. <i>Source: ISO 10303:239 Product Life Cycle Support.</i>	DateType	Data Types