Project: COMPASS
Grant Agreement: 287829
Comprehensive Modelling for Advanced Systems of Systems

Report on Guidelines for System Integration for SoS
Document Number: D21.4
Date: September 2013
Public Document

http://www.compass-research.eu
Contributors:
Jon Holt (JDH), ATEGO
Simon Perry (SAP), ATEGO
Finn Overgaard Hansen (FOH), AU
Stefan Hallerstede (SHA), AU
Klaus Kristensen (KK), B&O

Editors:
Simon Perry, ATEGO
Jon Holt, ATEGO

Reviewers:
Alvaro Miyazawa, UY
Lucas Albertins de Lima, UFPE
Richard Payne, UNUT
# Document History

## Release Versions

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2013-08-01</td>
<td>S. Perry</td>
<td>Initial complete version for review by COMPASS members.</td>
</tr>
<tr>
<td>1.1</td>
<td>2013-08-29</td>
<td>F.O. Hansen, J. Holt, S. Perry</td>
<td>Final release version with comments addressed.</td>
</tr>
<tr>
<td>1.2</td>
<td>2013-09-30</td>
<td>F.O. Hansen, J. Holt, S. Perry, K. Kristensen</td>
<td>Additional review comments addressed</td>
</tr>
</tbody>
</table>

## Pre-release versions

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2013-05-08</td>
<td>S. Perry</td>
<td>Initial draft outline version – incomplete</td>
</tr>
<tr>
<td>0.2</td>
<td>2013-06-21</td>
<td>J. Holt</td>
<td>Initial version of Atego’s content.</td>
</tr>
<tr>
<td>0.3</td>
<td>2013-06-25</td>
<td>S. Hallerstede</td>
<td>Added some ideas to Section 8</td>
</tr>
<tr>
<td>0.4</td>
<td>2013-06-27</td>
<td>S. Perry</td>
<td>Tool support section added.</td>
</tr>
<tr>
<td>0.5</td>
<td>2013-07-03</td>
<td>K. Kristensen</td>
<td>Inserted text for chapter 9. Application of System Integration guidelines</td>
</tr>
<tr>
<td>0.6</td>
<td>2013-07-08</td>
<td>F.O. Hansen</td>
<td>Inserted chapter 9, data exchange standards and technologies.</td>
</tr>
<tr>
<td>0.7</td>
<td>2013-07-10</td>
<td>F.O. Hansen</td>
<td>Finalized chapter 9.</td>
</tr>
<tr>
<td>0.8</td>
<td>2013-07-12</td>
<td>F.O. Hansen</td>
<td>Updated chapter 9 with a Summary and inserted review comments to other parts of the document</td>
</tr>
<tr>
<td>0.9</td>
<td>2013-07-31</td>
<td>K. Kristensen &amp; S. Perry</td>
<td>B&amp;O case study section complete</td>
</tr>
<tr>
<td>1.0</td>
<td>2013-08-21</td>
<td>K. Kristensen</td>
<td>Review updates in case study section</td>
</tr>
<tr>
<td>1.01</td>
<td>2013-08-27</td>
<td>F.O. Hansen</td>
<td>Updated chapter 9 with two standards</td>
</tr>
</tbody>
</table>
Abstract
This deliverable contains guidelines for the integration of Constituent Systems (CSs) into a System of Systems (SoS). At the core is the ontology, framework and viewpoints approach that is used throughout the various COMPASS guidelines. An integration ontology is defined and is used to define a framework of viewpoints required when adopting a model-based approach to integration. Each viewpoint is defined in terms of the ontology elements that can be shown on it. A set of integration processes are also defined that guide the engineer through the use of the framework. The deliverable also contains an overview of data exchange standards that may be relevant to the integration of systems together with information on the application of the guidelines which includes its application to the B&O case study.
Table of Contents

1. Introduction .................................................................................................................. 10
   1.1. Scope ...................................................................................................................... 10
   1.2. Context .................................................................................................................. 10
   1.3. Writing convention ............................................................................................... 10
   1.4. Structure of the Document ................................................................................. 12

2. Existing Best Practice Guidelines ............................................................................. 13
   2.1. Introduction ........................................................................................................... 13
   2.2. Best Practice from DoD SoS Guidelines ............................................................... 13
   2.3. Best Practice from ISO 15288:2008 .................................................................... 19
       2.3.1. ISO 15288 – the ‘Process Content View’ .................................................... 21
   2.4. Summary .............................................................................................................. 25

3. Ontology .................................................................................................................... 28
   3.1. The Integration Ontology ..................................................................................... 28

4. The Integration Framework ...................................................................................... 31

5. System Integration Guidelines - Processes .............................................................. 34
   5.1. System Integration Processes .............................................................................. 34
       5.1.1. Integration Technical Group .......................................................................... 35
       5.1.2. Integration Management Group ...................................................................... 45

6. Compliance .................................................................................................................. 50

7. Application of System Integration Guidelines .......................................................... 52
   7.1. Incremental integration ....................................................................................... 52
   7.2. Change in Constituent System .......................................................................... 53
   7.3. Addition of Constituent System .......................................................................... 54

8. Tool Support ............................................................................................................... 55
   8.1. Dissemination of Integration Framework and Process .......................................... 55
   8.2. Using SysML and COMPASS Tool to Support Integration ................................. 55
       8.2.1. Model system structure ............................................................................... 57
       8.2.2. Model system behaviour ............................................................................. 58
       8.2.3. Model system test cases .............................................................................. 58
       8.2.4. Generate CML from SysML & use COMPASS Tool ..................................... 59
   8.3. Initial integration of a system .............................................................................. 60
   8.4. Regression testing new system .......................................................................... 61

9. Data Exchange Standards and Technologies ............................................................ 63
   9.1. Domain Specific Standards ................................................................................ 63
       9.1.1. ISO 15926 Standard ...................................................................................... 63
       9.1.2. ISO 10303 (STEP) Standard ....................................................................... 65
       9.1.3. Healthcare Domain Standards ...................................................................... 66
   9.2. Text Based Data Exchange Standards ................................................................. 67
       9.2.1. XML & XML Schema .................................................................................. 67
       9.2.2. XMI .............................................................................................................. 68
       9.2.3. SOAP ........................................................................................................... 68
       9.2.4. JSON ............................................................................................................ 69
       9.2.5. IDL ................................................................................................................. 70
   9.3. Middleware Technologies ...................................................................................... 70
       9.3.1. CORBA .......................................................................................................... 70
9.3.2. Data Distribution Service Standard ....................................................................... 72
9.4. Summary .............................................................................................................. 73

10. Application to Case Studies ...................................................................................... 74
  10.1. B&O Interoperability Challenges .................................................................. 74
  10.2. SoS Integration Requirement for Constituent Systems ............................... 77
  10.3. The B&O AV Control Layer Architecture and AV Concepts ................. 83
  10.4. The DLNA AV Control Layer Architecture .............................................. 91
  10.5. Integration Challenges between DLNA CS and B&O CS ..................... 95
  10.6. Summary ....................................................................................................... 97

11. Conclusions .......................................................................................................... 98

12. References ............................................................................................................ 99
Figures

Figure 1 - Ontology overview for DoD SoS Guidelines ................................................. 13
Figure 2 - Ontology for DoD SoS guidelines with a focus on 'Focus Area' .............. 14
Figure 3 - Ontology for DoD SoS Guidelines with a focus on 'Core Element' ...... 15
Figure 4 - Process Content View for DoD SoS guidelines showing 'Technical Process' ........................................................................................................................................... 17
Figure 5 - Process Content View for DoD SoS guidelines with a focus on 'Technical Management Process' .................................................................................................................. 18
Figure 6 - ISO 15288 - Process Structure View ............................................................ 20
Figure 7 - ISO 15288 - Process Structure View showing 'Stage' and 'Gate' ....... 20
Figure 8 - Process Content View for ISO 15288 ......................................................... 22
Figure 9 - Process Content View for the Technical Processes Group .................. 23
Figure 10 - Process Content View showing activities and outcomes for the Integration Process ........................................................................................................................................ 24
Figure 11 - Process Content View showing activities and outcomes for the Validation Process ........................................................................................................................................... 25
Figure 12 - Requirement Context View showing the System integration Context ............................................................................................................................................ 26
Figure 13 - The existing COMPASS Ontology ............................................................... 28
Figure 14 - The Ontology for the Interface Pattern ..................................................... 29
Figure 15 Enhanced COMPASS Ontology with a focus on Integration aspects..... 30
Figure 16 - Information View showing Framework Views for Systems Integration Processes ...................................................................................................................................... 31
Figure 17 - Process Content View for Systems Integration Process showing Process Groups ............................................................................................................................................ 35
Figure 18 - Process Content View for Integration Technical Group .................... 36
Figure 19 - Process Content View showing the Constituent System Identification View ........................................................................................................................................ 37
Figure 20 - Information View for the Constituent System Identification Process 38
Figure 21 - Process Behaviour View for the Constituent System Identification Process ........................................................................................................................................ 39
Figure 22 - Process Content View showing the Integration Process ..................... 40
Figure 23 - Information View for the Integration Process .......................................... 41
Figure 24 - Process Behaviour View for the Integration Process ........................... 42
Figure 25 - Process Content View showing the Validation Process ....................... 43
Figure 26 - Information View for the Validation Process ......................................... 44
Figure 27 - Process Behaviour View for the Validation Process ............................ 45
Figure 28 - Process Content View for Integration Management Group ............... 46
Figure 29 - Process Content View showing the Integration Strategy Process ....... 47
Figure 30 - Information View for the Integration Strategy Process ........................ 48
Figure 31 - Process Behaviour View for the integration Strategy Process .......... 49
Figure 32 - Process Instance View showing incremental development scenario ...... 52
Figure 33 - Process Instance View showing change in CS scenario ........................ 53
Figure 34 - Process Instance View showing new CS scenario ............................... 54
Figure 35 - Using SysML to model System structure .............................................. 58
Figure 36 - Using SysML to model system behaviour .......................................... 58
Figure 37 - Using SysML to model system test cases (scenarios) ........................................... 59
Figure 38 - SysML to CML translation .................................................................................. 60
Figure 39 - Initial integration of a system .............................................................................. 60
Figure 40 - Regression testing new system ............................................................................ 61
Figure 41 Ontology Definition View from the AV CL framework .......................................... 76
Figure 42 CL CallStack layers ................................................................................................. 77
Figure 43 Media Content Browsing Requirement Description View ....................................... 79
Figure 44 Context Definition View for B&O CL SoS .............................................................. 80
Figure 45 The B&O CL Context Interaction View ................................................................... 82
Figure 46 B&O DD, DS and RD Interface Definition View ..................................................... 84
Figure 47 Interface Identification View for B&O’s CL Architecture ........................................ 85
Figure 48 B&O CL Interface Connectivity View ........................................................................ 86
Figure 49 B&O DD and SD Interface Behaviour View ............................................................ 87
Figure 50 B&O SR Interface Behaviour View .......................................................................... 87
Figure 51 high level meta-model view for CML translation ..................................................... 88
Figure 52 SysML Protocol Definition View for B&O’s DD and SD interfaces ....................... 89
Figure 53 CML Protocol Definition View for B&O’s DD and SD interfaces ......................... 89
Figure 54 refinement of the announcement operation ............................................................ 90
Figure 55 CML Protocol Definition View(s) for B&O’s SR system elements and interfaces .......................................................... 90
Figure 56 DLNA Interface Definition View ........................................................................... 93
Figure 57 DLNA Interface Connectivity View .......................................................................... 93
Figure 58 DD and SD CML Interface Protocol View(s) for DLNA ...................................... 94
Figure 59 CML SR Interface Protocol View(s) for DLNA ...................................................... 94
Figure 60 B&O DD and SD CML Protocol Definition View(s) supporting DLNA ............... 96
Figure 61 CML Validation Interaction View for B&O and DLNA CSs ..................................... 96
Tables

Table 1 - Example sentences to illustrate the convention ........................................ 12
Table 2 - Mapping between Systems Integration Framework Viewpoints and existing Viewpoints ................................................................................................................. 33
Table 3 - Compliance mapping between best practice and COMPASS Systems Integration Processes ........................................................................................................ 51
1. Introduction

This section defines the scope and context for this report. Also included in this section is a set of writing conventions that will be used throughout this report.

1.1. Scope

This document presents guidelines on the integration activity to be used in the COMPASS Project and for wider System of Systems (SoS) applications for the integration of Constituent Systems into SoSs. The document will take a model-based approach to developing the guidelines.

The basic approach taken will be consistent with the approach taken throughout the project, which is:

• Develop the COMPASS Ontology to include elements that relate to the integration of Constituent Systems into SoS.
• Develop a Framework that identifies the Viewpoints required for the integration of Constituent Systems into SoS.
• Define the Viewpoints that are required for the integration of Constituent Systems into SoS.

The Ontology, Framework and Viewpoints that are generated will then be used to generate a set of Processes for integration, which will then be applied to the Project itself via the industrial case studies.

1.2. Context

The guidelines presented in this document must be compliant with the other work that has been carried out on the COMPASS Project and, therefore, a substantial amount of re-use of existing work will be demonstrated.

The basic needs for the work will be based on best practice in the form of Standards and then these needs will be satisfied by the Processes that are defined.

Note that the architectural design is not being performed here, as the architectural Views for each CS and for the SoS will already exist; they will have been produced as part of the architectural design Processes (see [COMPASS D21.2]). It should also be noted that all CSs will have been verified and validated in their own right as part of their own development.

1.3. Writing convention

The following writing conventions are adopted in this report:
All terms from the SysML notation, that form part of the standard, are written in italics. Therefore, the use of *block* refers to the SysML construct, whereas the same word without italics – *block* - refers to an impediment.

All terms that are defined as part of the overall COMPASS model, such as the COMPASS Ontology, COMPASS Framework, etc. are presented with capitalised words. Therefore the use of *Project* refers to the Ontology Element, whereas the same word without capitals – *project* – refers to a non-specific usage of the term as a noun or verb.

All words that are being referenced from a specific diagram are shown in quotes. Therefore, the use of ‘Ontology Element’ is referring to a specific element in a specific diagram.

All View names are shown as singular. Therefore, the term Process Behaviour View may be referring to any number of diagrams, rather than a single one.

Any word that requires emphasis is shown in “double quotes”.

Domain-specific words in the B&O case study section are emphasised using **bold font**.

Some examples of this are:

<table>
<thead>
<tr>
<th>Example sentence</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Use Case may be visualised as a <em>use case</em></td>
<td>Use Case – term from the COMPASS Ontology</td>
</tr>
<tr>
<td></td>
<td><em>use case</em> - the term from SysML notation</td>
</tr>
<tr>
<td>Engineering activity can be shown as an Activity on a Process and may be visualised as an <em>activity</em></td>
<td>activity – the everyday usage of the word</td>
</tr>
<tr>
<td></td>
<td>Activity – the term from the MBSE Ontology</td>
</tr>
<tr>
<td></td>
<td><em>activity</em> – the term from the SysML notation</td>
</tr>
<tr>
<td>‘The diagram here shows the ‘COMPASS Process Framework’ is made up of one or more ‘Process Behaviour Viewpoint’</td>
<td>‘COMPASS Process Framework’ – a specific term from a specific diagram that is being described.</td>
</tr>
<tr>
<td></td>
<td>‘Process Behaviour Viewpoint’ - a specific term from a specific diagram that is being described.</td>
</tr>
<tr>
<td>When defining Processes it is typical to create a number of <em>activity diagrams</em> that will visualise the Process Behaviour View.</td>
<td>Processes - the term from the COMPASS Ontology</td>
</tr>
<tr>
<td></td>
<td><em>activity diagram</em> - the term from SysML notation</td>
</tr>
<tr>
<td></td>
<td>Process Behaviour View – the term from the COMPASS Framework</td>
</tr>
</tbody>
</table>
Example sentence | Meaning
--- | ---
It is important to understand the “why” of MBSE | “why” – emphasis of the word
The B&O SoS Context for this section is integration of the AV control layer technologies. | AV control layer technologies – a term from the B&O domain.

Table 1 - Example sentences to illustrate the convention

Table 1 shows some example sentences, the convention adopted and how they should be read. In summary, consideration must be given to ‘Quotes’, Capitalisation and italics as these have specific meaning. Finally, the use of “double quotes” simply represents emphasis.

1.4. Structure of the Document

Following this Introduction, Section 2 considers a number of best practice sources that are used as inputs to this report. In Section 3 the existing COMPASS Ontology is expanded to include the necessary concepts relating to integration. This Ontology is used in Section 3 to define an integration framework with supporting Processes defined in Section 5. In Section 6 the Processes are mapped back to the best practice sources discussed in Section 2.

Section 7 then discusses typical integration Scenarios, showing how the defined Processes can be used to address a number of SoS integration tasks. Tool support for SoS integration within the COMPASS Project is discussed at a high-level in Section 8, considering the dissemination of the framework and Processes and the use of SysML and CML to address common integration issues. This is followed, in Section 9, with a discussion of common data exchange standards and technologies that may be relevant to SoS integration.

In Section 10, the final major section, the framework and Processes defined in the document are applied to part of the B&O case study.

The document ends with conclusions in Section 11 and references in Section 12.
2. Existing Best Practice Guidelines

A key part of the COMPASS Project is to comply with and to re-use information from existing best practice guidelines. This section, therefore considers a number of best practice sources that are used for the SoS Integration Guidelines.

2.1. Introduction

As with all of the work on the COMPASS project, it is essential that all the guidelines produced can be mapped back to best practice. The integration work uses two main sources for this:


These two sources were modelled and the relevant parts are shown in the following two sections.

2.2. Best Practice from DoD SoS Guidelines

The first step in modelling the DoD SoS guidelines was to model the basic concepts and terms that are used in the form of an Ontology, which is shown in the following diagrams.

![Ontology overview for DoD SoS Guidelines](image)

The diagram in Figure 1 shows an overview of the Ontology for the DoD SoS guidelines.
According to the guidelines, ‘SoS Systems Engineering’ is made up on seven ‘Core Element’ and one or more ‘Systems Engineering Process’. The seven Core Elements provide the context for the application of all of the Systems Engineering Processes.

There are two types of Systems Engineering Process, which are: ‘Technical Process’ and ‘Technical Management Process’.

Finally, five ‘Focus Area’ are used across ‘SoS Systems Engineering’.

Each of these main elements is described in more detail on the following diagrams.

A ‘Focus Area’ is a cross-cutting concept that concentrates on an area of application that has been identified as being important for SoS engineering and that exists in DoD policy. These five Focus Areas, shown in Figure 2, are described as:

- ‘Program Requirements’, which defines how the program will manage all requirements whether they are statutory, regulatory, derived or certification requirements.
- ‘Technical Staffing and Organization Planning’, which defines how the program will structure and organize the program team to satisfy requirements.
- ‘Technical Baseline Management’, which establishes a technical baseline approach.
- ‘Technical Review Planning’, which shows how the program will manage the technical effort, including the technical baselines, through event-based technical reviews.
- ‘Integration with Overall Management of the Program’, which links Systems Engineering to other management efforts, including the
Acquisition Strategy, test planning, sustainment planning, configuration management, risk management, and lifecycle management.

These Focus Areas form the basis of all of the SoS engineering activity.

The diagram in Figure 3 shows the types of ‘Core Element’ that are identified in the guidelines. The set of seven Core Elements is used to describe system-of-systems environments, and are described as:

- ‘Translating Capability Objectives’, that involves codifying the SoS capability objective, which may be stated at a high level, leaving the task of clarifying and operationalizing the objectives and expectations to the SoS manager, systems engineer, and stakeholders. This Core Element uses the following processes: Requirements Development, Requirements Management, Risk Management, Configuration Management and Data management.

- ‘Understanding Systems and Relationships’, that involves developing an understanding of the systems involved in the SoS and their relationships and interdependencies. This also involves considering the System of Systems from a number of different perspectives, including: Organizational relationships among the systems, Stakeholders, Resourcing relationships, Requirements, Relationships among the development processes and life cycles. This Core Element uses the following processes: Logical Analysis, Risk Management, Configuration Management, Data Management and Interface Management.

- ‘Assessing Performance to Capability Objectives’, that involves focuses on developing metrics and collecting data from a variety of settings over time to monitor the performance of the SoS with respect to the user objectives. These assessments can take a variety of forms, including analysis, demonstration, and inspection. This Core Element
uses the following processes: Validation, Decision Analysis, Technical Assessment, Risk management and Data management.

- ‘Developing and Evolving an SoS Architecture’, that involves establishing a persistent technical framework for addressing the evolution of the SoS to meet user needs, including possible changes in systems functionality, performance or interfaces. This architecture does not address the details of the individual systems as it defines the way the systems work together to meet user needs and addresses the implementation of individual systems only when the functionality is key to crosscutting issues of the SoS. This Core Element uses the following processes: Requirements Development, Logical Analysis, Design Solution, Decision Analysis, Technical Planning, Requirements Management, Risk Management, Configuration Management, Data Management and Interface Management.

- ‘Monitoring and Assessing Changes’, which involves anticipating changes outside the control of the SoS that could affect the functionality or performance of an SoS capability. This includes changes to the technologies used to support the SoS or changes to the missions of the individual systems as well as external demands on the SoS. This Core Element uses the following processes: Decision Analysis, Risk Management, Configuration Management, Data Management and Interface Management.

- ‘Addressing Requirements and Solution Options’, that involves the systems engineer working with the SoS manager and stakeholders to review, prioritize, and recommend which SoS requirements to implement in each iteration. This analysis includes controlling top-level SoS requirements changes to maintain stability and coherence. This Core Element uses the following processes: Requirements Development, Design Solution, Decision Analysis, Technical Planning, Requirements Management, Risk Management, Configuration Management, Data Management and Interface Management.

- ‘Orchestrating Upgrades to SoS’, which involves SoS systems engineer facilitating, monitoring, and coordinating the changes being implemented in the systems to effect SoS performance improvements and added capability. This Core Element uses the following processes: Implementation, Integration, Verification, Validation, Transition, Decision Analysis, Technical Assessment, Requirements Management, Risk Management, Data management and Interface Management.

The Processes that are mentioned in these descriptions are described in the following two diagrams.
The diagram in Figure 4 shows the different types of 'Technical Process', which are:

- 'Requirements Development', which takes all inputs from relevant stakeholders and translates the inputs into technical requirements. This process is applied in the following core elements of SoS SE: Translating Capability Objectives, Developing and Evolving an SoS Architecture and Addressing Requirements and Solution Options.

- 'Logical Analysis', which obtains sets of logical solutions to improve understanding of the defined requirements and the relationships among the requirements (e.g., functional, behavioural, temporal). This process is applied in the following core elements of SoS SE: Understanding Systems and Relationships, Developing and Evolving an SoS Architecture.

- 'Design Solution', which translates the outputs of the Requirements Development and Logical Analysis processes into alternative design solutions and selects a final design solution. This process is applied in the following core elements of SoS SE: Developing and Evolving an SoS Architecture and Addressing Requirements and Solution Options.

- 'Implementation', which actually yields the lowest level system elements in the system hierarchy. The system element is made, bought, or reused. This process is applied to a single core element of SoS SE—Orchestrating Upgrades to SoS.

- 'Integration', which incorporates the lower level system elements into a higher-level system element in the physical architecture. This process is applied to a single core element of SoS SE: Orchestrating Upgrades to SoS.

- 'Verification', which confirms that the system element meets the design-to or build-to specifications and answers the question: did you build the system right? This process is applied to a single core element of SoS SE — Orchestrating Upgrades to SoS.

- 'Validation', which answers the question, did you build the right system? This process is applied in the following core elements of SoS SE: Assessing Performance to Capability Objectives and Orchestrating Upgrades to SoS.
• 'Transition', which is concerned with moving the developed system to the user. This process is applied to a single core element of SoS SE — Orchestrating Upgrades to SoS.

The following diagram describes the Technical Management Processes.

![Diagram](image)

Figure 5 - Process Content View for DoD SoS guidelines with a focus on 'Technical Management Process'

The diagram in Figure 5 shows the different types of 'Technical Management Process' and that are described as:

- 'Decision Analysis', which provides the basis for evaluating and selecting alternatives when decisions need to be made. This process is applied in the following core elements of SoS SE: Assessing Performance to Capability Objectives, Developing and an SoS Architecture, Monitoring and Assessing Changes, Addressing Requirements and Solution Options and Orchestrating Upgrades to SoS.

- 'Technical Planning', which ensures that the systems engineering processes are applied properly throughout a system's life cycle. This process is applied in the following core elements of SoS SE: Developing and Evolving an SoS Architecture and Addressing Requirements and Solution Options.

- 'Technical Assessment', which measures technical progress and the effectiveness of plans and requirements. This process is applied in the following core elements of SoS SE: Assessing Performance to Capability Objectives and Orchestrating Upgrades to SoS.

- 'Requirements Management', which provides traceability back to user-defined capabilities as documented through the Joint Capabilities Integration and Development System. This process is applied in the following core elements of SoS SE: Translating Capability Objectives, Developing and Evolving an SoS Architecture, Addressing Requirements and Solution Options and Orchestrating Upgrades to SoS.
• ‘Risk Management’, which helps to ensure program cost, schedule, and performance objectives are achieved at every stage in the life cycle and to communicate to all stakeholders the process for uncovering, determining the scope of, and managing program uncertainties. This process is applied in the following core elements of SoS SE: Translating Capability Objectives, Understanding Systems and Relationships, Assessing Performance to Capability Objectives, Developing and Evolving an SoS Architecture, Monitoring and Assessing Changes, Addressing Requirements and Solution Options and Orchestrating Upgrades to SoS.

• ‘Configuration Management’, which applies sound business practices to establish and maintain consistency of a product’s attributes with its requirements and product configuration information. This process is applied in the following core elements of SoS SE: Translating Capability Objectives, Understanding Systems and Relationships, Developing and Evolving an SoS Architecture, Monitoring and Assessing Changes and Addressing Requirements and Solution Options.

• ‘Data Management’, which addresses the handling of information necessary for or associated with product development and sustainment. This process is applied in the following core elements of SoS SE: Translating Capability Objectives, Understanding Systems and Relationships, Assessing Performance to Capability Objectives, Developing and Evolving an SoS Architecture, Monitoring and Assessing Changes, Addressing Requirements and Solution Options and Orchestrating Upgrades to SoS.

• ‘Interface Management’, which ensures interface definition and compliance among the elements that compose the system, as well as with other systems with which the system or system elements must interoperate. This process is applied in the following core elements of SoS SE: Understanding Systems and Relationships, Developing and Evolving an SoS Architecture, Monitoring and Assessing Changes, Addressing Requirements and Solution Options, and Orchestrating Upgrades to SoS.

All of the information contained in the DoD SoS Ontology was then used as a basis for deriving the basic Needs for the COMPASS SoS Integration Processes which is shown in Figure 12.

2.3. Best Practice from ISO 15288:2008

The Ontology for ISO 15288 is shown in the Process Structure View.
The diagram in Figure 6 shows the Ontology for ISO 15288. It is interesting to see that the Standard basically defines one or more ‘Process’ and proposes a ‘Life Cycle’, and that the terminology used in the Ontology reflects that. This is a typical pattern that can be seen in many Standards as many quality standards are essentially process-based and, hence, the emphasis will be solely on Processes and Life Cycles.

The Ontology is extended on the next diagram to emphasise the types of ‘Stage’ that make up the ‘Life Cycle’.

**Figure 6 - ISO 15288 - Process Structure View**

**Figure 7 - ISO 15288 - Process Structure View showing ‘Stage’ and ‘Gate’**
The diagram in Figure 7 shows a slightly different aspect of the overall Ontology for ISO 15288. In this view, the emphasis is on the different types of Life Cycle Stages and also the concept of the ‘Decision Gate’.

There are six Stages that are identified in the Standard, which are:

- The ‘Conception’ Stage, that covers the identification, analysis and specification of all the Needs for the Project. This may also include prototyping, trade studies, research or whatever is necessary to formalise the requirements set.
- The ‘Development’ Stage, that covers all the analysis and design for the Project and results in the complete System solution.
- The ‘Production’ Stage, that covers the production of the System of Interest, either as a one-off in the case of bespoke systems, or full manufacture, in the case of volume Systems. This Stage also includes the transition to operation.
- The ‘Utilization’ Stage, that covers the operation of the System at the target environment.
- The ‘Support’ Stage, that provides the maintenance, logistics and support for the system of interest whilst it is being operated.
- The ‘Retirement’ Stage, that covers the removal of the System of Interest and all related Processes or support mechanisms from service.

A ‘Decision Gate’ will usually manifest itself as some sort of review that takes place at the end of a ‘Stage’ and that makes a decision (‘Decision Option’) on what action to take next. The five basic types of ‘Decision Option’ are:

- ‘Execute next stage’ where the current Stage has gone according to plan and the Project may progress.
- ‘Continue this stage’ where more work needs to be carried out before the Project can progress to the next Stage. This will often require another ‘Decision Gate’ review to be executed.
- ‘Go to previous stage’ where something has happened that results in re-work and going back to a previous Stage.
- ‘Hold project activity’ where all work on the Project is halted, pending further investigations.
- ‘Terminate project’ where, for whatever reason, it has been decided that the Project will be killed off.

It should be stressed that the information in the Standard is there to provide recommendations and suggestions as to how the Life Cycle should be executed.

### 2.3.1. ISO 15288 – the ‘Process Content View’

The Process Content View for ISO 15288 is by far and away the most populated of the “seven views” (see [COMPASS D21.1] or [Holt2009] for a description of the “seven views”). This is because one of the main Requirements for the Standard, from the Requirement Context View, was to define a set of Processes in four areas.
The first of the Process Content Views shows the four basic types of 'Process Group', which are:

- 'Organizational Project-enabling Processes Group', which collects together all Processes that apply across the whole Organisation, all staff and all Projects.
- 'Technical Processes Group', which collects together the Processes that most people will associate with systems engineering and that cover areas such as requirements and design.
- 'Project Processes Group', which collects together Processes that are applied on a project-by-project basis, such as project planning, risk management, etc.
- 'Agreement Processes Group', which collects together Processes that describe the customer and supplier relationship in the Project.

Due to the size of the standard and the number of processes, only a subset of the Technical Processes Group will be discussed in any detail.
The diagram in Figure 9 shows that there are 11 Processes that make up the Technical Processes Group, which are described as:

- ‘Stakeholder Requirements Definition Process’, where the original requirements are captured and defined.
- ‘Requirements Analysis Process’, where the defined requirements are analysed and understood.
- ‘Architectural Design Process’, where a number of candidate solutions are defined.
- ‘Implementation Process’, where the system components are defined.
- ‘Integration Process’, where the system is constructed from its components.
- ‘Verification Process’, where the system is tested to ask the question: has the system been built right?
- ‘Validation Process’, where the system is tested to ask the question: have we built the right system?
- ‘Transition Process’, where the system is transported and installed in its target environment.
- ‘Maintenance Process’, where the system is monitored and maintained as necessary.
- ‘Operations Process’, where the system is utilised by the operators and end users.
- ‘Disposal Process’, where the system is decommissioned in a safe fashion.

Due to space limitation, the only two Processes that will be considered in detail are the Integration Process and Validation Process.
The diagram in Figure 10 shows the breakdown on the ‘Integration Process’ into its component Activities, each of which comprises a number of Tasks. The Process also shows the desired Outcomes in the form of SysML properties that have been stereotyped to «outcome».

The ‘Integration Process’ comprises two main Activities:

- ‘Plan integration’, which defines the order that the System will be assembled in and also identifies any constraints that will impact the assembly.
- ‘Perform integration’, which carries out the actual integration of the System elements into the System.

The result of the integration activities in the form of the delivered System must also be validated, hence the ‘Validation Process’ is also of interest.
The diagram in Figure 11 shows the breakdown on the 'Validation Process' into its component Activities, each of which comprises a number of Tasks. The Process also shows the desired Outcomes in the form of SysML properties that have been stereotyped to «outcome».

The 'Validation Process' comprises two main Activities:
- 'Plan validation', that defines a strategy for the integration and then defines a plan based on this.
- 'Perform validation', that is concerned with executing and managing the validation activities.

**2.4. Summary**

This section has presented high-level models of the two main sources for the creation of the SoS Integration Processes.

Based on the information contained in the models, the following processes were identified as being the most relevant for this work:

- DoD SoS guidelines: 'Integration' process and 'Validation' process
- ISO 15288: 'Integration Process' and 'Validation Process'

These processes are now used as a basis for generating the Needs for the SoS Integration Processes, which can be seen on the following diagram.
The diagram in Figure 12 shows the System integration Context that was abstracted from the two best-practice standards that were modelled in the previous sections.

It can be seen that the main *use case* is concerned with providing guidelines for SoS integration (‘Provide guidelines for SoS integration’) that has two main constraints:

- To meet best practice, in terms of established Standards (‘Meet best practice’)
- To be compliant with the other Processes that have been developed as part of the COMPASS Project work (‘Be compliant with other processes’) such as the Requirements Processes and the Architecture Processes.

The main *use case* has a single inclusion that is concerned with defining a set of Processes for SoS integration (‘Define process’). This *use case* has several inclusions, which are:

- To allow Constituent Systems to be identified that exist within the SoS (‘Identify constituent systems’).
- To identify the Interfaces that exist between Constituent Systems (‘Identify interfaces’) that are relevant to the SoS.
- To define the Interfaces that have been identified as being relevant to the SoS (‘Define interfaces’). In some situations the Interface definition will already exist, in which case the definitions will be used here. In other situations, however, the Interface definitions will not exist and will need to be defined here.
- To validate the SoS (‘Validate SoS’) to ensure that the interactions between the Constituent Systems satisfy the Original Needs of the SoS.
This Context will be used as a basis for creating the Systems Integration Processes.
3. Ontology

This section looks at the existing COMPASS Ontology and considers changes and enhancements that will be made in order to make it suitable for the SoS Integration Guidelines.¹

The COMPASS Ontology identifies and defines all of the terms and concepts that are used in the Project.

The existing COMPASS Ontology was used as a start point and then expanded to include concepts that relate to integration.

3.1. The Integration Ontology

The start point for the ontology modelling was the existing COMPASS Ontology that has been used throughout the Project, for example, see SoS Requirements Guidelines [COMPASS D21.1] and the SoS Architecture Guidelines [COMPASS D21.2].

![Figure 13 - The existing COMPASS Ontology](image)

The diagram in Figure 13 shows the existing COMPASS Ontology that has been used so far in the Project for the SoS Requirements and Architecture work.

In order to understand how this existing Ontology may be expanded to include integration-related concepts, the Interface Pattern was used [COMPASS D22.3] as a source of additional concepts that were added to the Ontology. The Interface

¹ The COMPASS Ontology, the frameworks based on the Ontology and the COMPASS processes are made available through Atego Process Director. See Section 8.1 for details.
pattern has its own set of concepts and terms that are shown in the Interface Ontology on the following diagram.

The diagram in Figure 14 shows the Ontology for the Interface Pattern. By comparing the existing COMPASS Ontology (Figure 13) and the Interface Pattern Ontology, it is quite clear that the connection point between the two Ontologies is in the area of the ‘System’. Part of the definition of the ‘System’ is that it is made up of one or more ‘System Element’ [COMPASS 2012a] which itself exists on the Interface Pattern. Therefore, these two Ontologies may be brought together to create the expanded Ontology shown below.

---

2 ‘System’ is not currently included as an Ontology Element on the Interface Pattern. It will be added in deliverable D22.6 – Final Report on SoS Architectural Models, due at the end of the COMPASS Project.
The diagram in Figure 15 shows the enhanced Ontology that combines both Ontologies. (It should be noted here that for the sake of brevity, the requirements-related concepts from the COMPASS Ontology have been omitted for clarity).

**Figure 15 Enhanced COMPASS Ontology with a focus on Integration aspects**
4. The Integration Framework

The Integration Framework identifies a number of Viewpoints required for the integration of SoSs, each of which is re-used from other areas of the COMPASS project.

Each Viewpoint defines the structure and content of a View that can be produced for a particular purpose, in this case SoS integration. Each Viewpoint is defined in terms of the elements from the Ontology that are relevant to that Viewpoint; only elements from the Ontology can appear in the definition of a Viewpoint – anything that is required to be shown on a Viewpoint must be part of the Ontology (an Ontology Element). For a discussion of (Architectural) Frameworks, Viewpoints and Views see [COMPASS D21.2].

The diagram in Figure 16 shows the 11 Integration Framework Viewpoints that make up the Integration Framework. These Viewpoints may be described briefly as:

- 'Context Definition Viewpoint' - identifies the Constituent Systems that make up the SoS.
- 'Context Interaction Viewpoint' - combines a number of Contexts for Constituent Systems that make up the SoS.
- 'Interface Identification Viewpoint' - identifies the Interfaces that exist between the Constituent Systems, based on the Context Interaction Viewpoint.
- 'Interface Connectivity Viewpoint' - defines how the Interfaces, identified in the Interface Definition Viewpoint, are connected together.
- 'Interface Definition Viewpoint' - defines the structure of each Interface identified in the Interface Identification Viewpoint.
- 'Interface Behaviour Viewpoint' - defines the interactions between the System Elements where the Interfaces lie.
- 'Protocol Definition Viewpoint' - defines any Protocols, if required, for each Interface.
'Validation Interaction Viewpoint’ – allows comparison between the Scenarios for the SoS and the Interface Behaviour Viewpoints, in order to demonstrate the validation of the SoS.

'Requirement Context Viewpoint’ – has two uses: defines the Needs in Context for each Constituent System, forming the basis for the Context Interaction Viewpoint; defines the Needs in Context for the Integration Processes.

'Process Content Viewpoint’, where the Processes for the Integration are identified.

'Process Instance Viewpoint’, where the Processes from the Process Content Viewpoint are validated against the Requirement Context Viewpoint.

All of the Viewpoints in the Integration Framework are re-used from other Frameworks within COMPASS and therefore have already been defined. The Viewpoints, along with a reference to their definition, are shown in the following table.

<table>
<thead>
<tr>
<th>Integration Viewpoint Name</th>
<th>Source</th>
<th>Document Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Identification Viewpoint</td>
<td>Interface Pattern</td>
<td>D22.3 Report on Modelling Patterns for SoS Architecture</td>
</tr>
<tr>
<td>Interface Connectivity Viewpoint</td>
<td>Interface Pattern</td>
<td>D22.3 Report on Modelling Patterns for SoS Architecture</td>
</tr>
<tr>
<td>Interface Definition Viewpoint</td>
<td>Interface Pattern</td>
<td>D22.3 Report on Modelling Patterns for SoS Architecture</td>
</tr>
<tr>
<td>Interface Behaviour Viewpoint</td>
<td>Interface Pattern</td>
<td>D22.3 Report on Modelling Patterns for SoS Architecture</td>
</tr>
<tr>
<td>Protocol Definition Viewpoint</td>
<td>Interface Pattern</td>
<td>D22.3 Report on Modelling Patterns for SoS Architecture</td>
</tr>
<tr>
<td>Validation Interaction Viewpoint</td>
<td>Interface Pattern</td>
<td>D22.3 Report on Modelling Patterns for SoS Architecture</td>
</tr>
<tr>
<td>Integration Viewpoint Name</td>
<td>Source</td>
<td>Document Reference</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Requirement Context Viewpoint</td>
<td>Requirements Framework and “Seven Views” Process modelling approach (this Viewpoint is reused from the Requirements Framework in the “Seven Views” approach.)</td>
<td>D21.1 Report on Guidelines for SoS Requirements</td>
</tr>
</tbody>
</table>

*Table 2 - Mapping between Systems Integration Framework Viewpoints and existing Viewpoints*
5. System Integration Guidelines - Processes

At the heart of the systems integration guidelines lies a set of Processes that may be executed to satisfy the basic needs identified in Figure 12. These Processes may be collected into two broad Process Groups, which are the Integration Technical Group and the Integration Management Group.

These Processes may be executed in a number of sequences to show different Scenarios, depending on why the Processes are needed.

There are four Processes that are described as:

- 'Integration Strategy Process', that defines the overall strategy for the integration exercise. This consists of identifying the need for the integration and then identifying a set of Processes that may be used to satisfy these needs. The Processes are then validated against the needs and, based on this, an integration plan is created.
- 'Constituent System Identification Process', where the relevant Constituent Systems are identified, based on the Context Definition View for the SoS. A number of Interfaces are then identified between the Constituent Systems that are relevant to the SoS. Note that the architectural design is not being performed here, as the architectural Views for each CS and for the SoS will already exist; they will have been produced as part of the architectural design Processes (see [COMPASS D21.2]).
- 'Integration Process', where the main integration Views are created based on the Interfaces between the Constituent Systems and their associated behaviour.
- 'Validation Process', where the behaviours for the Interfaces are validated against the behaviour of the SoS. It should be noted that all CSs will have been verified and validated in their own right as part of their own development.

A number of possible Scenarios are explored in Figure 32, Figure 33 and Figure 34 in Section 6.

The Processes are defined in the following sub-sections using the “seven views” approach that has been used throughout the COMPASS project. A summary of the “seven views” approach can be found in [COMPASS D21.1].

5.1. System Integration Processes

The Systems Integration Processes are categorised in terms of two main groups that are shown in the following diagram.
The diagram in Figure 17 shows the high-level Process Content View for the Systems Integration Processes. It can be seen that there are two process groups:

- 'Integration Technical Group', which contains processes that perform the integration itself.
- 'Integration Management Group', which contains a process to define the integration strategy and planning.

Each of the Processes in these groups will be described in the following two sections.

NOTE: Many of the Processes produce Artefacts that are of type Requirement Context Viewpoint (RCV), for example the CS requirement context views produced by the Constituent System Identification Process. Such RCVs throughout represent the minimum that is needed; the whole of the SoS-ACRE Process and all its associated Views could be carried out here in order to generate these RCVs. See [COMPASS D21.1].

### 5.1.1. Integration Technical Group

This section describes the processes that make up the Integration Technical Group.
The diagram in Figure 18 shows Process Content View for the Technical Integration Process Group. Each Process is described in more detail below.

**Constituent System Identification Process**

The following diagram shows the Constituent System Identification Process.
The diagram in Figure 19 shows the detailed Process Content View that focusses on the Constituent System Identification Process.

The Activities for the Process are represented as SysML operations and are described below:

- ‘assemble SoS Context’, that combines the Requirement Context Views (represented by the ‘CS requirement context view’ part) into the ‘Context interaction view’ for the SoS.
- ‘identify interfaces between CS’, where the Interface Identification Views, represented by the ‘CS interface’ part, for each Interface between two Constituent Systems are defined. These are then combined to form a collection of these Interface Identification Views, using the ‘Interface set’ part.
- ‘review’, where all Artefacts are reviewed.
- ‘baseline’, where all Artefacts are placed under formal configuration control.

The Artefacts that are produced and consumed by the Activities of this Process are shown as SysML properties and can be seen in more detail on the following Information View.
The diagram in Figure 20 shows the Information View for the Constituent System Identification Process that identifies the Artefacts and the relationships between them. The Artefacts represent Views based on the following Viewpoints:

- ‘Context Definition Viewpoint’ – This has the ‘Context definition view’ Artefact based on it, which identifies the Constituent Systems that make up the SoS and may identify Interfaces between these Constituent Systems.
- ‘Requirement Context Viewpoint’ – This has the ‘CS requirement context’ Artefacts based on it, which define the Contexts for each CS.
- ‘Context Interaction Viewpoint’ – This has the ‘Context interaction view’ Artefact based on it, which combines the Contexts for each of the Constituent Systems into a single View that may then be used to identify Interfaces.
- ‘Interface Identification Viewpoint’ – This has two Artefacts based on it: the ‘CS interface’ and the ‘Interface set’. The ‘CS interface’ identifies a single Interface or group of Interfaces between two Constituent Systems. The ‘Interface set’ is the collection of all the individual ‘CS interfaces’.

The execution of the Process is shown in the following diagram.
The diagram in Figure 21 shows the Process Behaviour View for the Constituent System Identification Process.

**Integration Process**
The following diagram shows the Integration Process.
The diagram in Figure 22 shows the detailed Process Content View that focusses on the Integration Process.

The Activities for the Process are represented as SysML operations and are described below.

- 'select interface', where a single Interface is selected from the 'Interface set'.
- 'define interface connectivity', where the connections between the System Elements represented by the Interfaces are defined by creating the 'Interface connectivity view'.
- 'define interface structure', where each Interface has its structure defined by creating the 'Interface definition view'.
- 'define interface behaviour', where the interactions between the System Elements and across the selected Interface are defined by creating the 'Interface behaviour view'.
- 'define protocol', where the Protocol, if required, is defined for each Interface by creating the 'Protocol definition view'.
- 'review', where all Artefacts are reviewed.
- 'baseline', where all Artefacts are placed under formal configuration control.
It should be noted that some of the Artefacts generated by these Activities may already exist and, therefore, the relevant activity will be one of collation rather than generation.

The Artefacts that are produced and consumed by this Process are shown as SysML properties and can be seen in more detail on the following Information View.

The diagram in Figure 23 shows the Information View for the Integration Process that identifies the Artefacts and the relationships between them. The Artefacts are described as:

- 'Interface Identification Viewpoint' – This has the 'Interface set' Artefact based on it, which identifies the Interfaces that exist between the Constituent Systems.
- 'Interface Connectivity Viewpoint' – This has the 'Interface connectivity view' Artefact based on it, which defines how the Interfaces, identified in the 'Interface definition view', are connected together.
- 'Interface Definition Viewpoint' – This has the 'Interface definition view' Artefact based on it, where each Interface identified in the 'Interface set' has its structure defined.
- 'Interface Behaviour Viewpoint' – This has the ‘Interface behaviour view’ Artefact based on it, where the interactions between the System Elements where the Interfaces lie are defined.
- 'Protocol Definition Viewpoint' – This has the ‘Protocol definition view' based on it, which defines any Protocols, if required, for each Interface.

The execution of the Process is shown in the following diagram.
The diagram in Figure 24 shows the Process Behaviour View for the Constituent System Identification Process.

**Validation Process**
The following diagram shows the Validation Process.
The diagram in Figure 25 shows the detailed Process Content View that focuses on the Validation Process.

The Activities for the Process are represented as SysML *operations* and are described below.

- ‘assemble SoS validation’, where the ‘Validation interaction views’ for the SoS are collected together.
- ‘identify interfaces’, where the ‘Interface set’ is used to show the Interfaces.
- ‘assemble interface scenarios’, where the ‘Interface behaviour views’ for the Interfaces are collected together.
- ‘perform analysis’, that performs the actual validation by comparing the set of ‘Interface behaviour views’ for the Interfaces to the ‘Validation interaction views’ for the SoS.
- ‘review’, where all Artefacts are reviewed.
- ‘baseline’, where all Artefacts are placed under formal configuration control.

The Artefacts that are produced and consumed by this Process are shown as SysML properties and can be seen in more detail on the following Information View.
The diagram in Figure 26 shows the Information View for the Validation Process that identifies the Artefacts and the relationships between them. The Artefacts are described as:

- 'Interface Identification Viewpoint' – That has the 'Interface set' Artefact based on it, which identifies the Interfaces that exist between the Constituent Systems.
- 'Interface Behaviour Viewpoint' – That has the 'Interface behaviour view' Artefacts based on it, where the interactions between the System Elements where the Interfaces lie are defined.
- 'Validation Interaction Viewpoint' – That has the 'Validation interaction view' Artefacts based on it, where the Scenarios for the SoS are compared to the 'Interface behaviour views' to demonstrate the validation of the SoS.

The execution of the Process is shown in the following diagram.
The diagram in Figure 27 shows the Process Behaviour View for the Constituent System Identification Process.

### 5.1.2. Integration Management Group

This section describes the processes that make up the Integration Management Group.
Figure 28 - Process Content View for Integration Management Group

The diagram in Figure 28 shows Process Content View for the Integration Management Group. The single Process is described in more detail below.

**Integration Strategy Process**

The following diagram shows the Integration Strategy Process.
The diagram in Figure 29 shows the detailed Process Content View that focuses on the Constituent System Identification Process.

The Activities for the Process are represented as SysML *operations* and are described below.

- ‘define integration needs’, where the ‘Requirement context view’ from the point of view of the Integration is created. Note that any pre- or post-integration Needs must be defined here. For example, that a CS must meet a minimum technical specification before it can be considered for integration into the SoS.
- ‘identify processes’, where the relevant Processes for Integration are identified by creating an appropriate ‘Process content view’ or using an existing ‘Process content view’ that includes the necessary Processes. In the case of the COMPASS Project, this will be the Systems Integration Processes.
- ‘validate processes vs Needs’, where the ‘Process instance view’ for the Processes will be used to validate that the Processes selected satisfy the original Needs.
- ‘create integration plan’, where, based on the previous Activities, an ‘Integration plan’ for the integration activities is created.
- ‘review’, where all Artefacts are reviewed.

The Artefacts that are produced and consumed by this Process are shown as SysML *properties* and can be seen in more detail on the following Information View.
The diagram in Figure 30 shows the Information View for the Constituent System Identification Process that identifies the Artefacts and the relationships between them. The Artefacts are described as:

- ‘Requirement Context Viewpoint’ – That has the ‘Requirement context view’ Artefact based on it, where the Needs for the Integration Processes are defined.
- ‘Process Content Viewpoint’ – That has the ‘Process content view’ Artefact based on it, where the Processes for the Integration are identified.
- ‘Process Instance Viewpoint’ – That has the ‘Process instance view’ Artefact based on it, where the Processes from the ‘Process content view’ are validated against the ‘Requirement context view’.
- ‘Integration Plan’ – That has the ‘Integration plan’ Artefact based on it, created from the ‘Process content view’ and the ‘Process instance view’.

The execution of the Process is shown in the following diagram.
The diagram in Figure 21 shows the Process Behaviour View for the Integration Strategy Process.
## 6. Compliance

The Processes presented in this report map back onto the best practice information presented previously. The following table shows this compliance mapping.

<table>
<thead>
<tr>
<th>COMPASS Systems Integration Processes</th>
<th>ISO 15288</th>
<th>DoD SoS Guidelines</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Constituent System Identification Process</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>identify SoS and Constituent Systems</td>
<td>Integration Process, Perform integration</td>
<td>Integration</td>
</tr>
<tr>
<td>assemble SoS Context</td>
<td>Integration Process, Perform integration</td>
<td>Integration</td>
</tr>
<tr>
<td>identify interfaces between Constituent Systems</td>
<td>Integration Process, Perform integration</td>
<td>Integration</td>
</tr>
<tr>
<td>Review</td>
<td>Integration Process, Perform integration</td>
<td>Integration</td>
</tr>
<tr>
<td>Baseline</td>
<td>Integration Process, Perform integration</td>
<td>Configuration Management</td>
</tr>
<tr>
<td><strong>Integration Process</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>select interface</td>
<td>Integration Process, Perform integration</td>
<td>Integration</td>
</tr>
<tr>
<td>define interface connectivity</td>
<td>Integration Process, Perform integration</td>
<td>Integration</td>
</tr>
<tr>
<td>define interface structure</td>
<td>Integration Process, Perform integration</td>
<td>Integration</td>
</tr>
<tr>
<td>define interface behaviour</td>
<td>Integration Process, Perform integration</td>
<td>Integration</td>
</tr>
<tr>
<td>define protocol</td>
<td>Integration Process, Perform integration</td>
<td>Integration</td>
</tr>
<tr>
<td>Review</td>
<td>Integration Process, Perform integration</td>
<td>Integration</td>
</tr>
<tr>
<td>Baseline</td>
<td>Integration Process, Perform integration</td>
<td>Configuration Management</td>
</tr>
<tr>
<td><strong>Validation Process</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>assemble SoS validation</td>
<td>Validation Process, Perform validation</td>
<td>Validation</td>
</tr>
<tr>
<td>identify interfaces</td>
<td>Validation Process, Perform validation</td>
<td>Validation</td>
</tr>
<tr>
<td>assemble interface scenarios</td>
<td>Validation Process, Perform validation</td>
<td>Validation</td>
</tr>
<tr>
<td>perform analysis</td>
<td>Validation Process, Perform validation</td>
<td>Validation</td>
</tr>
<tr>
<td>Review</td>
<td>Validation Process, Perform validation</td>
<td>Validation</td>
</tr>
<tr>
<td>Baseline</td>
<td>Validation Process,</td>
<td>Configuration</td>
</tr>
<tr>
<td>COMPASS Systems Integration Processes</td>
<td>ISO 15288</td>
<td>DoD SoS Guidelines</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Integration Strategy Process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>define integration needs</td>
<td>Integration Process, Plan integration</td>
<td>Interface Management</td>
</tr>
<tr>
<td>identify processes</td>
<td>Integration Process, Plan integration</td>
<td>Interface Management</td>
</tr>
<tr>
<td>validate processes vs needs</td>
<td>Integration Process, Plan integration</td>
<td>Interface Management</td>
</tr>
<tr>
<td>create integration plan</td>
<td>Integration Process, Plan integration</td>
<td>Interface Management</td>
</tr>
<tr>
<td>review</td>
<td>Integration Process, Plan integration</td>
<td>Interface Management</td>
</tr>
<tr>
<td>baseline</td>
<td>Integration Process, Plan integration</td>
<td>Configuration Management</td>
</tr>
</tbody>
</table>

Table 3 - Compliance mapping between best practice and COMPASS Systems Integration Processes
7. Application of System Integration Guidelines

This section shows some possible Scenarios for executing the SoS Integration Processes for different purposes. The examples here are not intended to represent an exhaustive list of Scenarios, but are simply intended to show the flexibility of the Processes.

7.1. Incremental integration

This scenario considers the situation where the SoS is being developed in an incremental fashion.

In some instances it may not be feasible nor desirable to deliver the entire SoS in one single iteration. This may be for a number of reasons, including:

- The size of the SoS. The SoS may simply be too large to be delivered in a single iteration.
- Evolution of the SoS. One of the characteristics of a SoS is that it may evolve over time, in which case it may be thought of growing and evolving in an incremental fashion.
- Risk. In many cases it may be considered too high a risk to deliver the SoS in a single iteration and it may be more desirable to implement a single Constituent System at a time.

In this case it is essential that the integration strategy be defined before the Constituent Systems are integrated into the SoS.

In any of these cases, the following Scenario may be applicable.

The diagram in Figure 32 shows the Process Instance View for the scenario where the SoS is developed in an incremental fashion. The basic sequence of the Processes is quite straightforward and they are executed in a linear fashion. The
incremental aspect of the scenario can be seen by the *combined fragment* that shows that the 'Integration Process' and 'Validation process' are executed once for each increment. See [Holt & Perry 2008] for a discussion of *combined fragments* in SysML. This results in the final SoS being delivered over a number of increments, rather than in a single deployment.

### 7.2. Change in Constituent System

This scenario considers the situation where the SoS is potentially impacted by a change occurring on one of the Constituent Systems.

A typical SoS will exhibit different degrees of operational and management independence. Due to these two characteristics, it is inevitable that one or more Constituent System will change in such a way as to impact the overall SoS.

The identification of a change in a Constituent System may be picked up elsewhere in the COMPASS Processes, for example in the System of Systems Requirements Management Processes as described in [COMPASS D21.1].

In such cases, the following Scenario may be applicable.

![Figure 33 - Process Instance View showing change in CS scenario](image)

The diagram in Figure 33 shows the Process Instance View for the scenario where the SoS is impacted due to a change in a Constituent System.

The scenario starts when a change is made to a Constituent System, shown here with the 'Change in CS' signal arriving from the *gate* on the left-hand side of the diagram. The Processes are then executed in a simple linear fashion.
7.3. Addition of Constituent System

This scenario considers the situation where the SoS is impacted by a new Constituent System being added to the SoS.

A key characteristic of a SoS is that it evolves over time. This may include changes to the Constituent Systems, as described in the previous section, or may include the introduction of an entirely new Constituent System.

In this case the Scenario would be very similar to the one that considers changes to a Constituent System, except that the signal that effects the Scenario would be different. An example of this Scenario is shown in the following diagram.

The diagram in Figure 34 shows the Process Instance View for the scenario where the SoS is impacted due to a new Constituent System being introduced into the SoS.

The scenario starts when a new Constituent System is introduced, shown here with the 'New CS' signal arriving from the gate on the left-hand side of the diagram. The Processes are then executed in a simple linear fashion.
8. Tool Support

This section describes the tool support for the integration framework and Processes both in terms of dissemination and feedback on the framework and Processes and in terms of how SysML and the COMPASS tool chain can be used together to help support integration.

8.1. Dissemination of Integration Framework and Process

In order to ensure the validity and applicability of the integration framework and Processes described in this document it is essential that they be made available to users so that they can be executed and user comments and feedback gathered to allow the processes to be improved as necessary based on such feedback. During the lifetime of the COMPASS project the integration framework and Processes will be used by the industrial partners as part of their case study work investigating COMPASS methods in tasks T4.1.1 and T4.2.2, and as part of the industrial challenge problem in task T4.3.3.

The framework and Processes will be disseminated to other COMPASS members and to members of the COMPASS Interest Group (CIG) by means of Atego Process Director (APD), a web-based tool supporting the capture and dissemination of processes, as stated in the COMPASS ‘Description of Work’ [DoW2011].

As well as allowing for Process dissemination, APD also supports Process feedback. Users of Processes held in APD will be able to submit comments and feedback on the Processes directly in APD. Any comments and feedback can then be used as inputs for improvements to the framework and Processes, which will be updated and reissued through APD.

8.2. Using SysML and COMPASS Tool to Support Integration

This section presents some high-level guidelines for using tools to support integration. It is a vision of the way such tools will be used once the COMPASS Project is finished. The guidelines are not illustrated using an actual example as the various tools being produced by the COMPASS Project do not yet support this vision. Nevertheless, it provides a useful road map to how the tools may be used in the future and presents an illustration of the way in which an analysis based on formal model checking may be used with more semi-formal techniques, giving high confidence, through the formality, in the results produced by the analysis.

Using SysML and the COMPASS Tool together can help with integration as described below. Note that in these descriptions the term “System” is used in a generic way:

- Initial integration of a System. When a Constituent System is being integrated into an SoS (existing or new), the toolset can help to address
the question of whether the System behaviour is verified by the defined test cases. Of course, such tests have to be run against the implemented System. However, at design time the confidence in both the design of the Constituent System and that of the test cases can be increased by demonstrating that the modelled behaviour of the System is verified by the modelled test cases. This can, of course, be done manually. However, when there are large numbers of test cases or complicated System behaviour, then checking each one against the modelled behaviour can become an onerous task and hence one that is rarely carried out.

- Regression testing when integrating a new System as a replacement for an existing System. When an existing System is replaced by a new System it is often the case that the new System must guarantee to deliver at least the same functionality as the original System. This is normally established through the classic regression testing approach, where tests that the original System has passed are run against the new System to show that it delivers the same functionality. Again, these tests have to be run but confidence in the tests and the design can be greatly increased and potential problems identified and addressed, if it is possible to demonstrate such compliance at design time.

- In both cases, it is important to find out whether the System does anything that has not been specified in test cases. If it does, then either it is a simple case of missing test cases or, more significantly, it means that the System provides unintended functionality. Establishing this through actual testing can often be very time consuming if not impossible. Again, the ability to investigate this question at design time greatly increases the confidence in the System design and can help identify potential problems allowing them to be addressed prior to implementation.

The general approach to addressing these points at design time is described below. This is an outline of the approach which, necessarily, depends on familiarity with SysML, CML and their associated toolsets. A detailed description is outside the scope of this document. The approach can be summarised as follows:

- Use SysML to model:
  - System structure, Interfaces & connectivity
  - System behaviour
  - Test cases (Scenarios)

- Translate SysML into CML (see [COMPASS D22.4] for guidance on translating SysML into CML):
  - CML state & behavioural model (based on SysML System structure and behavioural model)
  - CML behavioural model (based on SysML test cases)

- Then use CML models and tools to address integration issues

The Integration Framework presented in section 4 and the System Integration Processes defined in section 5 should be used in the modelling of those aspects
of the System relevant to integration. The intention here is not to show how the various tools are used within the Process, but rather to illustrate the way in which the semi-formal and formal aspects of a model of a System may be used together as part of integration.

8.2.1. Model system structure

Start by using SysML to model:

- System structure – showing the System Elements (for a System) or Constituent Systems (for an SoS) and their relationships.
- System connectivity – show how the System Elements or Constituent Systems are connected, representing the high-level and abstract relationships from the System structure model as connections between Interfaces. The Interface Pattern (see [COMPASS D22.3]) can be used for this.
- System Interfaces – define the functionality provided by the identified Interfaces, together with any pre- and post-conditions. Note that the full Interface Pattern found in [COMPASS D22.3] should be used here. The example shown in the figure is a much simplified representation showing only the defined operations provided by an Interface.
- System properties & operations – define the properties and additional (internal) operations of the System Elements and Constituent Systems. Specify any pre- and post-conditions on the operations and any invariants on the System properties in order to strengthen their definition, allowing a “design by contract” approach to be used. See [Payne & Fitzgerald 2011].

This is summarised in Figure 35.
8.2.2. Model system behaviour

Model the behaviour of the various System Elements or Constituent Systems using *state machine diagrams*. Here it is the behaviour within a System Element or Constituent System that is being modelled, rather than the behaviour between System Elements or Constituent Systems. The *block operations* and *properties* referenced in the *state machine diagram* were specified when modelling the System structure. See Figure 36.

Note also that, depending on how the System is modelled or where the relevant functionality is located, such behaviour may actually reside in the Interfaces between System Elements. That is, the *state machine diagram*, is “inside” an Interface provided by a System Element rather than inside the System Element itself. Such definition of Interface behaviour is described in the Interface Pattern in [COMPASS D22.3].

8.2.3. Model system test cases

Model the interactions between System Elements or Constituent Systems in the form of Scenarios using *sequence diagrams*. Abstract out common behaviour to make Scenarios easier to understand. These Scenarios form verification test cases for the System. They should test, and be consistent with, the modelled System behaviour. That is, they should test all the modelled behaviour and also be consistent with it. Thus Scenarios that model both valid and invalid behaviour are needed that ensure that the System reacts correctly to valid inputs and rejects invalid inputs. See Figure 37.
8.2.4. Generate CML from SysML & use COMPASS Tool

With the SysML model complete, use the SysML to CML translation functionality to generate CML from the SysML (see [COMPASS D22.4] for guidance on translating SysML into CML):

- The SysML structural model becomes the CML state model. State in CML is a largely a structural concept, representing the state variables and properties of the System, although it also includes operations which may modify the state. These operations should be consistent (naming, signature etc.) with those specified in the behavioural part.
- The SysML behavioural model (state machine diagrams) form part of the CML behavioural model (CML processes and actions in CML processes). Note that activity diagrams could also be used to model aspects of the SysML behavioural model.
- The SysML test cases (the Scenarios modelled using sequence diagrams) form the rest of the CML behavioural model.
Note that there will also need to be type definitions and functions defined that are used in the state and behavioural parts of the CML model. These would be modelled as part of the SysML modelling activity, but have been omitted from the diagrams in this presentation in order to simplify the explanation presented here.

The COMPASS Tool can now be used to address the issues integration issues, as described in the following sections.

8.3. Initial integration of a system

Using the COMPASS Tool:

1. Check that the Scenarios specified by the sequence diagrams are consistent with the behaviour specified by the state machine diagrams, i.e. check the Scenarios against the state machine behaviour. This is checking that the Sequence diagrams refine the state machines. If any fail, then:
   - The Scenario is correct but is modelling invalid behaviour which has therefore been detected and hence a successful test can be recorded. Or...
   - The Scenario is incorrect and should be corrected and retested against the state machine behaviour. Or...
   - The System behaviour as modelled is wrong and therefore the state machine behaviour should be corrected and the tests performed again.

2. Check that the behaviour specified by the state machine diagrams is covered by the union of all the Scenarios. This is checking that the state machines
refine the *sequence diagrams*. If this is not the case, then the COMPASS Tool will find additional sequences that are also valid for the *state machine*. In this case:

- If the discovered counter examples are considered to be valid Scenarios, then the new Scenario should be added to the set of test cases.
- If the discovered counter examples are considered to be invalid Scenarios, then the System behaviour as modelled is wrong; it is allowing some functionality that it should not and should be corrected and retested.

The analysis carried out here is formal model checking which, therefore, gives a high confidence in the results produced by the analysis.

### 8.4. Regression testing new system

Using the COMPASS Tool, test the new System behaviour against the old Scenarios, using the approach defined previously:

1. Ensures that the tests that the System tests passed previously are still passed by the new System i.e. it does what it did before.
2. Ensures that either the System doesn’t do anything that it didn’t do previously or that such behaviour is identified, in which case an analysis can be made as to whether this behaviour is desired.

If the new System has intended new functionality, then Scenarios will have been modelled to test this new functionality. These must also be tested against the new System behaviour using the approach defined previously:

3. Ensures that the new behaviour passes the new tests.
4. Ensures that there are no missing tests, and identifies any if there are. If additional Scenarios are identified in 2) AND the new System intentionally added new functionality, then these new Scenarios should already be included in those tests in 3).

Again, the analysis carried out here is formal model checking which, therefore, gives a high confidence in the results produced by the analysis.
9. Data Exchange Standards and Technologies

This section will introduce examples of different data exchange standards and middleware technologies that could be used in the integration of one or more Constituent Systems to form a SoS. The information in this section is primarily useful for SoS integration of networked CS, which are IT systems.

As data exchange standards are used to interconnect CSs in a SOS and allowing them to communicate they form an essential part of the integration process into a SoS and can in some situations exclude a given CS to be part of a SoS if it uses incompatible standards or it is too expensive to adapt it to the common used SoS standards.

The selection of the standards and technologies is based on the following criterias:

- Selection of standard based exchange protocols, which are currently used by many distributed systems. Ideally only internationally recognized standards should be selected, but in some examples also de-facto standards are included like JSON.
- Support for integration and interoperability of heterogeneous constituent systems, where each CS can have its own operating system and programming language.

These criteria’s have excluded vendor specific communication solutions like Microsoft’s .NET, with Windows Communication Framework (WCF) and Oracle’s Java RMI. These technologies can be used internally in the constituent systems, but are excluded here, as it will not be the common case that all constituent systems are homogenous systems based on the same vendor based technology.

9.1. Domain Specific Standards

This section will introduce three different domain specific standards used for different purposes and illustrating data exchange standards used in integration scenarios for SoS.

9.1.1. ISO 15926 Standard

ISO 15926 is an international ISO standard originally developed for the process industry e.g. oil and gas industry and later extended to include the automation industry for product development [ISO15926Std]. It is currently foreseen that this standard could be used in other domains as well. The standard has a long history started back in 1980/90 and first part of the ISO 15926 published around 2000. ISO 15926 is still being developed and extended with new sub standards called parts. This long history illustrates a development, where the standard has evolved toward a more integrated and automated data exchange between the many CSs participating in a SoS for these target domains.
The idea behind the standard was initially to make a common reference library for all kinds of equipment used in the process industry e.g. pumps, valves, pipelines etc. Each of these assets is equipped with a unique identifier and a unique set of description attributes. These assets are described in the ISO 15926-Part 4: “Initial Reference data”. Based on this information two independent systems can now reference the same type of asset e.g. a pump in a uniform and standardized way including transferring this information from one system to another. Another important part of the standard is ISO 15926-Part 2: “Data Model”, which describes an ontology for possible relations between assets in the reference library e.g. that a pump can be connected with a valve. To bring the automation of the communication further ISO 15926-Part 7: “Implementation methods for the integration of distributed systems: Template Methodology”, describes a large number of templates which can be used for building concrete instances of asset data to be communicated including their internal relationships.

The next step towards interoperability is obtained by the definition of ISO 15926-Part 8: “Implementation methods for the integration of distributed systems: Web Ontology Language (OWL) implementation”. Using the OWL standard enables two different communicating systems to make automated data exchange and interpret the received data based on the OWL description. The last step towards an automated interaction is obtained by using ISO 15926-Part 9: “Implementation standards”, with focus on Facades, web servers, web services and security issues. The ISO 15926 standard is currently supported by a number of different tools.

**ISO 15926 Integration challenges**

This standard enables different CSs to exchange domain information based on the standardized reference data library ISO 15926-Part 4. It is a huge standard to get started with for new companies entering in the domain, but it enables international exchange of data for assets in the process and automation industry domain.

One of the challenges is to implement the newer updates to the standard e.g. Part 8 and Part 9, which allows more automated and integrated exchange of information between the participating CS. An example if one of the CS has implemented part 8 and part 9 using OWL and has made a Façade with access via web services, then the other communicating CSs have to implement these facilities as well. A more simple integration example is the case where the participating CS implements Part 2, Part 4 and Part 7, where the data exchange is based on information sharing via file exchange.

An actual integration based on this standard can use one or more of the available supporting tools available.
9.1.2. ISO 10303 (STEP) Standard


STEP can as an example be used to exchange data for 3D objects in Computer aided design (CAD). STEP can handle product data from different disciplines for example mechanical and electrical designs as well as data specific to different industries such as automotive, aerospace, building construction and ship building. The Objective of STEP is to be able to describe products through their life cycle and both provide a file exchange format and also enable sharing of product databases.

ISO 10303 consists of a large set of application protocols (APs), where the application protocol ISO10303-233: “Part 233: Application protocol: Systems engineering” [ISO10303-233Std2012] called AP233, describes an information model for the exchange of data to support model based systems engineering, where data are exchanged between Systems and Software Engineering tools.

The AP233 information model captures the concepts used in Systems Engineering. AP233 is divided into two major set of capabilities: System Modelling and Program Management. The system modelling capabilities include system structure, system behaviour, modelling, decision support, requirements, analysis and trade studies. The program management capabilities include project management, verification and validation, risk and issue management. AP233 is designed to integrate the other more domain specific application protocols in ISO 10303, enabling integration of systems engineering processes and tools among organizations working in a supply chain or on the same project, where each organization can use its own sets of processes and tools.

The AP233 standard has a set of the same goals as the SysML standard, which combined with the XMI standard (described in 9.2.2) can be used to exchange systems engineering models in the same way as with AP233.

ISO 10303 and ISO 10303-233 Integration challenges

ISO 10303 enables different CSs to exchange product domain information with the same goals as defined for ISO 15926, but with a broader scope and covering different domains than ISO 15926.

Using ISO 10303-233 different CSs can exchange systems engineering models and in this way collaborate in development and maintenance of a SoS. Important information to exchange is definition of the CSs interfaces.
STEP has defined an XML-based file exchange capability for exchanging models for example AP233 models. These models needs to be translated to XMI files for integration into SysML based Systems Engineering tools. The other way around is when AP233 based tools needs to import models from SysML based tools, where these tools needs to either export the models in AP233 format or the AP233 tools needs to have an XMI-SysML import facility. Another integration possibility is to develop application level interfaces to access XML based models in either AP233 XML format or in SysML XMI format.

9.1.3. Healthcare Domain Standards

Another domain with extensive on-going standardization activities is the healthcare domain, where several standards currently are being developed for enhancing the interoperability of diverse healthcare CS to communicate with other systems or to form a SoS. The healthcare domain is currently dominated by many incompatible systems and also with a number of different and incompatible standards.

An example of an on-going standard is the IEEE11073 [IEEE11073Std], which initially was developed to enable plug-and-play functionality of intensive care unit equipment in a hospital setting. This standard is currently being developed towards integration of wireless medical equipment to allow these to be used in for example a home setting with home monitoring of patients' vital signs. This standard is an example of an upper layer standard, which standardized the semantic data communicated between the systems. An example could be the communication with a blood pressure device, where the exact format for a blood pressure measurement is described by the standard.

Another standard example is the HL7 standard [HL7-Std1987] for exchange of data between hospital information systems, which as an example enables exchange of Patient Health Records (PHR) stored in different hospitals PHR systems.

Healthcare Standards Integration challenges

One of the actual scenarios in this domain is to enable end-to-end communication between a CS placed in a patient or citizens home or in nursing homes and CSs placed at a Hospital Site, the local doctors CS and possible also the home care CS in the local municipalities home care centre. Making all these systems participating in a SoS, requires use of appropriate standards as these CS typically are developed and operated by different Organisations.

One of the challenges here is to enforce the use of international standards as the HL7 standard and in communication with equipment's connected to patients to use standards like the IEEE11073 standard set.

For healthcare devices an industry alliance like the Continua Alliance [Continua] will enforce use of international recognized standards and will as well provide a certification process, which means that vendors in the future can sell Continua
Certified devices, certified to be interoperable with other Continua certified devices or systems. In this way it will enforce Healthcare SoS to incorporate Continua certified CSs.

One of the current problems in this domain is that it is dominated by stove pipe CSs, which are based on proprietary data exchange standards. In the future it is required that all healthcare’s CS on a national scale are able to interoperating exchange for example a patient’s healthcare data and in this way enabling treatment on different and more specialized hospitals. Another challenge in this domain is to implement the high level of security, which are required for distribution of personal healthcare data. This gives rise to the important challenge in the integration process, to verify, that the required security is obtained and in line with e.g. National Regulation Rules for exchange of confidential personal health related data.

9.2. Text Based Data Exchange Standards

This section will introduce the most commonly used text based data exchange standards. All these example of data exchange standards are general standards which can represents any kind of data exchange.

9.2.1. XML & XML Schema

The eXensible Markup Language (XML) has in recent years been widely used for exchange of information between distributed systems. XML is standardized by the W3C Consortium as Version 1.0 [W3C-XML-Std2008].

XML defines a set of encoding rules for representing information in a format that is both readable by humans and machines. An XML file can refer to a XML Schema, which is a standard for defining rules for the valid contents of an XML file. XML can be passed as files or as data between communicating distributed systems. XML was invented for communication over the internet but is currently used in many other situations e.g. for configuration purposes. There exist currently many open source tools or libraries to generate and pass XML files written in different programming language.

XML Integration Challenges

As XML data is exchanged in text form and therefore is human readable it is very easy to see the structure of the data exchanged. This means it is easy to decode the data send in this format from another CS and make an appropriate adapter or wrapper for processing these data.

The human readable format will also make the integration test easier as it is easy to verify if the right data are send between the participating CS.

An integration of a set of CSs into a SoS will require, that all participating parties have access to the XML Schema file, defining the rules for the XML file. This XML
Schema file should be put under version control as well, as all communicating parties must use the same version of this file.

Security plays an important challenge here as it is easy to read and change the XML data. This is normally solved by using an encryption and decryption standard on both side of the communication.

Another integration challenge is to secure the performance, as the XML format is a fairly space demanding standard due to its structure with start and end tags and due to its text format. Passing an XML file with open source tools can also be a performance challenge and in cases with more limited embedded CS the XML parser normally takes up a great deal of the programming memory space.

9.2.2. XMI

XML Metadata Interchange (XMI) is a standard defined by OMG for exchanging metadata information via XML between application development lifecycle tools in distributed heterogeneous environments. XMI is from 2005 also an ISO/IEC standard [ISO19503Std2005]. XMI integrates three industry standards: XML, UML and MOF (Meta Object Facility, an OMG and ISO/IEC 19502 standard).

The most common use of XMI has been to exchange UML and SysML models between different vendors UML or SysML modelling tools. XMI is also used to transfer UML models to other software engineering tools like code generators and testing tools.

XMI Integration Challenges

For a successful exchange of XMI based models it requires that both the exporting CS and tool and the importing CS and tool has implemented the XMI standard right. This is normally obtained if the tools are from the same vendor, but is more difficult when integrating tools from different vendors. The difficulties come as an example, from different support of the UML standard, where not all vendors have supported all facilities in the newest version of the standard. Diagram interchange is also problematic as XMI do not support graphical layout information.

9.2.3. SOAP

Simple Object Access Protocol (SOAP) is a World Wide Web Consortium (W3C) protocol specification for exchanging structured information in the implementation of Web Services in computer networks [W3C-SOAP-Std2007]. SOAP uses an XML based format for specifying web service method calls with parameters and a format for obtaining return information.

A SOAP message (called an envelope) consists of a header and a body containing the actual exchange data. The envelope starts by defining encoding rules etc. The header is optional and can as an example contain authentication data. The body
specify either the service call with appropriate parameters or the return results from a previous SOAP call.

SOAP messages are normally sent over the HTTP protocol. One of the advantages of this standard is its ability to traverse firewalls due to the use of the HTTP as the transport protocol.

SOAP is widely used to implement Service Oriented Architecture (SOA) systems based on web services. SOA is discussed as a pattern for SoS architecture in [COMPASS D22.3]. Web services are in SOA registered at a service registry, where each service is described in a standardized way using the Web Service Description Language (WSDL) [WSDL-Std2001].

**SOAP Integration Challenges**

As SOAP is based on XML it has some of the same challenges as described in the XML section e.g. with security and performance.

In addition to this the call semantic between the CS calling a service and the service provider must conform to the same interface specification. This means that the service/method name should be right and having the right number of parameters with the right data types. In according to this they should also both agree on the return SOAP message and the type of the results.

Using a service registry where the providing CS can register its services in the WSDL format, will help the calling part to form a correct SOAP message with a call semantic according to the registered format.

On the server site the server can have its services specified in one or more provided interfaces implemented by the server. This specification can use the COMPASS Interface Pattern described in [D22.3].

**9.2.4. JSON**

JavaScript Object Notation (JSON) is a text-based lightweight standard for data exchange [JSON]. JSON is based on a subset of the JavaScript Programming Language, but can be used independent of the JavaScript Programming Language [JSON-Std2006]. JSON is a language-independent text format, with parsers available for many programming languages. JSON can represent simple data structures and associative arrays.

JSON has a schema called JSON Schema which is used for validation and control of JSON data. The JSON format is often used for serializing and transmitting structured data over a network connection. It was designed primarily to transmit data between web client application and a server, serving as a simpler alternative to XML, but is now used as a general data exchange format between communicating systems.
JSON has also a remote procedure call protocol called JSON-RPC, which is based on the same concepts as SOAP. JSON is in many way compatible with XML, but simpler in format with a smaller overhead.

**JSON Integration Challenges**

JSON has some of the same integration challenges as mentioned for XML and SOAP. In comparison with XML, JSON is a little more compact and easy to parse just like Java-script declarations. The syntax is much more readable than XML. As a JSON parser is simpler it is more feasible to use JSON instead of XML in constrained CS like an embedded system.

### 9.2.5. IDL

Interface Definition Language (IDL) or alternatively Interface Description Language is originally specified by OMG as a part of CORBA 3.0 [OMG-IDL-Std2002]. IDL is a language-independent way of specifying an interface for a remote method invocation call with parameters and return types. It is used as an integral part of a CORBA system, where IDL compilers translate the IDL interface description to auto generated stub and skeleton files in the concrete programming language, which are used by each side of the client/server communication. In this way the IDL-file is used to secure that both the method call and the method implementation are using the same number and type of parameters and return types. The idea behind IDL has been adopted by many other technologies. For this reason there exist many IDL based dialects e.g. CORBAs IDL and IDL for the Android operating system (AIDL).

An IDL interface is specified in IDL source code and then converted to the appropriate exchange format. When used in CORBA it is transferred as binary information between the participating CSs.

**IDL Integration Challenges**

The challenge here is to use the same version of the IDL interface on both sites of the communication and have both sites to agree upon using the right versions of the IDL files. This means that these files should be shared between all participating CS to be used in the development of the necessary integration software. Another challenge is to use the same version or dialect of the IDL standard as well. As IDL is normally used in standardized middleware technologies it requires all participating CSs to use the same middleware, but can be middleware implementations from different vendors.

### 9.3. Middleware Technologies

#### 9.3.1. CORBA

The Common Object Request Broker Architecture (CORBA) is another OMG middleware standard published back in 1991. The latest version is CORBA
version 3.3 from November 2012 [OMG-CORBA-Std2012]. CORBA is based on Remote Method Invocation (RMI) from clients to CORBA based servers, where the objects called are located. CORBA supports heterogeneous operating systems and programming languages. CORBA call is transferred by binary protocols, where the most used is the Internet InterOrb Protocol (IIOP), allowing calls over the Internet. CORBA use the Interface Definition Language (IDL) to specify the method names, data types for parameters and return types.

Standard CORBA calls are based on a synchronous remote method call, waiting on a return from the called server object. Later versions of CORBA have also introduced asynchronous method calls.

CORBA has coupling in space and time and partly also flow coupling. Coupling in space is required because the client must know the identification of the server and the server object before the call can be performed. Coupling in time is required as a client can't call a server object which is not available. The normal CORBA solution to obtain a decoupling in space is by using a Naming Server, where remote servers can register their object references. The CORBA Naming Service specification defines the functionality for a naming server.

It is also possible to implement publish/subscribe systems on top of CORBA by using the supplementary CORBA standards for Event and Notification Services.

**CORBA Integration challenges**

As IDL is a part of CORBA, the challenges mentioned for IDL will also be the challenges for CORBA. In addition to this all sites of the communication should support the same version of CORBA. If different vendor implementations of object request brokers (ORBs) are used, there can be compatibility problems if the ORBs implement different versions of the standards.

Another integration problem with CORBA used to communicate between different networks is that it requires open communication ports in the CSs firewalls for allowing the CORBA traffic to traverse the firewalls.

A CORBA based solution requires each CS to have the CORBA software i.e. the object request broker (ORB) linked into the application.

For enabling a decoupling between the participating CS, the SoS should include a Naming Server in one of the CS, where the CS CORBA servers can register their CORBA objects and where the CS CORBA clients can query to get the remote address of a CORBA object called the Interoperable Object Reference (IOR).

Each CORBA object on the CS CORBA server site can be specified with one or more provided interfaces using the COMPASS Interface Pattern described in [D22.3]. On the CS CORBA client site the client object can have one or more required interfaces, which again could be specified with the COMPASS Interface pattern.
As CORBA is based on a binary data exchange it is more difficult to decode the information sent between the communicating parties, which complicate the testing process. The advantage of CORBA in relation to the text based transfer protocols is better performance and a higher security due to the binary data exchange.

### 9.3.2. Data Distribution Service Standard

The Data Distribution Service (DDS) is standardized by the Object Management Group (OMG) in 2004. The current version of the DDS standard is version 1.2 from January 2007 [OMG-DDS-Std2007].

DDS standardizes a data centric publish/subscribe based middleware, where the communication is based on topics. A topic describes a structured data type which is used to share data information between systems and applications. The standard supports many-to-many communication as there can be many publishers, publishing at a given topic as well as many subscribers connected to the same topic. DDS is discussed as a data centric publish/subscribe pattern for SoS architecture in [COMPASS D22.3].

The DDS paradigm enables a decoupling between the participants, where they are decoupled in space, time and flow. Space decoupling is obtained as publishers and subscribes don’t need to know each other’s location or address. Time decoupling is obtained as they don’t need to be online or active at the same time. Flow decoupling is obtained as publishers can publish information asynchronously to the DDS middleware and subscribes can as well be invoked by an asynchronous activation.

DDS is based on a strongly typed communications paradigm, where 21 different Quality of Services (QoS) parameters can be specified at different levels. These QoS parameters can be specified in XML documents and forms contracts between the communicating parties. The topics and data types used can be specified by IDL, XML or UML.

A DDS Interoperability (DDSI) protocol is also standardized, where the latest version 2.1 is specified in January 2009 [OMG-DDSI-Std2009]. DDSI specifies a binary wire protocol for allowing interoperability between different DDS implementations.

DDS is based on a shared information model, where the information about topics, publishers and subscribers are distributed automatically, by the DDS infrastructure components, between the participating systems.

A short introduction to DDS as a communication middleware for scalable and extensible Systems-of-Systems can be found in [Corsaro&2012], where they argue that DDS is the ideal technology for integrating SoS, where it is used as the integration infrastructure. The main arguments for this are DDS support for
interoperability, portability, loose coupling, extensibility, scalability, efficiency and timeliness. In addition to this a Web-enabled DDS standard has recently been defined, which allows a web access to a DDS based system.

**DDS Integration challenges**

All participating CS in the SoS must have an implementation of the DDS middleware linked into their application. As DDSI standardize the wire protocol different vendors DDS solutions should be able to communicate in the specified DDS domain. Currently a DDS SoS works perfectly on the same local area network (LAN) but have challenges in communicating between LANs over wide area networks, especially when different vendors DDS implementation are used.

All CSs in the SoS must use the same topic definition specified by either IDL, XML or UML.

One of the larger integration challenges using DDS is to correctly specify the values of the 21 defined QoS parameters defined in DDS. This must be done for the Topic and also for both the publisher and subscribers. This is a very important integration activity as the QoS parameters defines the way the data exchanges occurs. The DDS implementation helps by this by providing automatic check for example if the QoS defined for a subscription can’t be satisfied by the publisher, then an error message will be given at the subscription time.

As DDS is based on a binary data exchange it is more difficult to decode the information sent between the communicating parties, which complicate the testing process.

The advantage of DDS in relation to the text based transfer protocols is better performance with delivery guarantees and a higher security.

Integration of CS in a DDS based SoS is supported by DDSs support for dynamic discovery of new topic as well as the discovery of new publishers or new subscribers.

**9.4. Summary**

This section has presented an overview of different types of data exchange protocols and technologies for exchange of information between CSs participating in a SoS. For each protocol and technology the integration challenges are described. Integration between CSs using different text based protocols is in general a simple task which involves writing adapters which works as protocol converters. Integration of CSs using binary protocols requires in general these systems to use the same type of middleware, but also in this case a wrapper component could include an instance of the used middleware and work as the integration component towards a CS, who don't use the communication middleware in its internal design.
10. Application to Case Studies

This section will present an application of the integration guidelines based on a B&O case study. The following sections will first define the integration case study and then describe the results of applying the COMPASS integration guidelines to the case study.

This section will use the terminology of the Audio-Visual (AV) technology domain, the AV business domain and the COMPASS SoS domain. The following list tries to clarify the terms used in the different domains:

- A product is the same as a device
- One device can represent several Constituent Systems
- The Context of the SoS can map one device or product to one Constituent System

The B&O SoS Context for this section is integration of the AV control layer technologies, hence:

- Each AV control layer technology is represented by one Constituent System in the B&O SoS Architecture.
- Each Constituent System can be represented by one or more devices in SoS Scenarios.
- Each Constituent System represent both architectural and Organisation Stakeholder Roles.

In the following sections some of the integration views have been omitted for clarity. The domain descriptions in the sections are using abstractions for low-level technical and business concepts.

10.1. B&O Interoperability Challenges

The use of proprietary technology in B&O is causing interoperability challenges with business areas evolution. The two fastest growing B&O business areas are currently Play and Automotive, where products are based on open standard technologies and ecosystems, and therefore conflict with the closed proprietary based technology other B&O products are built from. Content (media streams) is getting more and more complex. The AV content providers enforce rules and even user-experiences that might degenerate the B&O experience space. Furthermore the modern consumer is no longer interested in buying products that only work with other products from the same supplier. The new mantra is interoperability, and not only on a SCART or HDMI connector level, but also transparent network interoperability. The business areas and technology challenges mentioned above are forcing B&O to move away from the old closed technology base towards a more open philosophy.
The move from closed to open interoperable technology challenges the management of emergent behaviours\(^3\) for B&O. The set of emergent behaviours must conform to B&O brand experience rules regarding robustness and quality attributes. The robustness and experience space of the SoS is now heavily influenced by the capabilities of the Constituent Systems. Unmanaged emergent behaviours might even degrade the B&O SoS and lead to brand damage.

The open philosophy is causing a paradigm shift from System to SoS engineering in the B&O development Organisation. B&O is moving from developing Constituent Systems to SoS level engineering. The Organisation’s evolution requirements are also driven by hostile business and technology domains represented by Constituent Systems of the SoS.

One of B&O’s COMPASS tools and method expectations is the capability of expressing and analyzing SoS models during early design stages, before expensive implementation commitments are made. The desire is that construction and analysis of B&O SoS models helps to identify areas of incompleteness or ambiguity in informal system specifications, and provides some level of confidence that a valid implementation will have key properties. Since CML has formal mathematical semantics, which enable a wide range of analyses on models like checking internal consistency or confirmation of models emergent properties, B&O expect guidelines for using CML as part of integration processes.

For this integration guideline application we are using the control layer (CL) part of the AV Architecture. The CL is the top layer which the user interacts with using a local control interface (hence the term control layer). The CL part of the AV Architecture has the following responsibilities:

- Responsible for device discovery and device management
- Responsible for service discovery and service management
- Responsible for connectivity between remote and local services
- Responsible for event propagation between local and remote services

Figure 41 shows the CL Ontology Definition View (ODV) from the AV Control Layer Framework (AVCLF). The ODV is an instance of the COMPASS Architectural Framework Framework’s (CAFF)’s Ontology Definition Viewpoint [COMPASS D21.2]. Ontology Elements from the Interface Pattern [COMPASS D22.3] are used to give meaning regarding Port Connections between Service Interfaces. Using an Ontology Definition View for the AV domain enables consistent communication regarding AV domain concepts and their relationships.

---

\(^3\) emergent behaviours equals positive and desired user experiences provided by the CSs of SoS
The CL ODV Ontology Elements give the following AV CL semantics:

- A 'Product' consists of 'System Element'(s), which can be 'Data' and/or 'Operation'
- The 'System Element'(s) provides one or more AV 'Source Capability' and 'Renderer Capability'
- An 'AV Capability' is exposed on the network by one or more 'Network Service Interface'(s) (through the 'Port' relation)
- A 'Product' takes a 'Product Role', either a 'Source Role' or 'Target Role'
- A 'Source Role' and 'Target Role' communicate using a 'Network Service Interface'
- An 'Network Service Interface' and a 'Service Mechanism Interface' can be a Flow or Service based Interface (since these are the allowed sub-types of Interface from the Interface Pattern Ontology)
- A 'Network Service Interface' use of one or more 'Service Mechanism Interface'(s), either 'Announcement', 'Request' or 'Status'
- A 'Service Mechanism Interface' use a 'Network Transport Strategy'
- A 'Interface' takes place across a 'Port Connection' for communication between 'Source Role' and 'Target Role'(s).
- A 'Port' is connected to one or more other 'Port' via a 'Port Connection'.
- One or more 'Port' is owned by a 'System Element'

The Ontology Definition View covers both the conceptual and technical concepts of the AV control layer domain. The 'Network Service Interface'(s) are high level application layer interfaces. 'Network Service Interface'(s) expose service
operation specifications inside a process\(^4\). ‘Service Mechanism Interface’(s) represent the real IP transport semantics behind the ‘Network Service Interface’(s) operations. ‘Service Mechanism Interface’(s) are expose services on the network in form of data types and IP protocols. Figure 42 shows the runtime ‘CallStack’ ‘Layer’ relations between ‘Network Service Interface’(s), ‘Service Mechanism Interface’(s) and the ‘Network Transport Strategy’(s).

---

\(^4\) The called has access to the same process function environment
Requirement Description View. The subset of Rules used has the following definitions:

- **Source product** is the product having the media content: hence the product provides AV ‘Source Capabilities’
- A **discoverable product** can be discovered by other products on the same network: hence the product has an ‘Announcement’ ‘Service Mechanism’
- **Target product** is the product that the source product’s contents is being browsed and displayed on. The **target product** is the product the user interacts with: hence the source product provides a network ‘Request’ ‘Service Mechanism’

The CL architectural-centric SoS Use Case chosen as the SoS integration and interoperability challenge is the media content browsing Scenario (the CB Scenario). The CB Scenario is representing Requirements derived from a content browsing end-user scenario Source Element [COMPASS D21.1]. The Source Element describes indirectly user interaction Need(s) [COMPASS D21.1] regarding SoS contents browsing.

The content browsing end-user Scenario has the following steps:

**Content browsing End-User Scenario**

- **Actors:**
  - User
- **Pre-condition:**
  - The source and target products are discoverable on the network.
- **Basic flow:**
  - User presses “Browse Content” on **target product**
  - **Target product** discovers **source product**
  - **Target product** discovers browsing service on **source product**.
  - **Target product** connects to browsing service on **source product**
  - **Target product** uses browsing service on **source product**.
- **Post-condition:**
  - **Source product’s** media content information is transferred and displayed on **target product**.

The constraints for the integration solutions regarding the CB Scenario are:

- B&O will not deploy code on any non-B&O constituent systems
- Only the Constituent System’s **target product** Interfaces are to be supported by B&O products

---

5 Browsing means only information (content type, location, transport protocol, ...) about the contents is transferred to target product. Not the contents itself
Figure 43 shows the Requirement Description View (RDV) for the SoS Constituent Systems. The Requirement Description View is a Viewpoint from SoS-ACRE (see [COMPASS D21.1]). The RDV Requirements 'Service Discovery', 'Device Discovery' and 'Service Requests' are referred to as the minimal requirements which any CS, before integration into the B&O SoS, must satisfy regarding the CB Scenario. The 'Remotely Located Content Browsing' Requirement is the SoS-level Requirement which must be satisfied by the B&O SoS after integration of a given CS regarding the CB Scenario.

DLNA and AirPlay are chosen as CL Constituent Systems for this application since they represent Grey- and Black-box Constituent System integration challenges for B&O.

Figure 44 shows the Context Definition View (CDV) for the B&O CL SoS with the DLNA and the AirPlay Stakeholder Roles. It should be noted here that for the sake of brevity each CS on the CDV represents both System Context and business context Stakeholders. The CL Context Definition View is produced using the Constituent System Identification Process; where the first Activity states:

---

6 Minimal requirements must be agreed upon that make the SoS operational. Optional requirements must be agreed upon that extend the capabilities of the SoS.

5 The Digital Living Network Alliance (DLNA) is a non-profit collaborative trade organization established by Sony in June 2003, that is responsible for defining interoperability guidelines to enable sharing of digital media between multimedia devices.

8 AirPlay (previously called AirTunes when it was for audio only[1]) is a proprietary protocol stack/suite developed by Apple Inc. that allows wireless streaming of audio, video, and photos, together with related metadata between devices.

Figure 44 Context Definition View for B&O CL. SoS

The technology and business domains of each Constituent System will constrain the system integration elements and the integration processes for the Architecture. These integration constraints will be identified using the COMPASS CS-openness classification principles from D21.2 and the integration framework Views. On the CL Context Definition View, the Constituent Systems of the SoS are **CS-openness stereotyped** with the following tag values:9

- **DLNA CS** (Grey box tag)
  - Is the core behaviour of a CS specified and public?
    - Yes: DLNA products are certified and must conform to the DLNA specification
  - Is it possible to determine all stakeholders?
    - Yes: DLNA is alliance of global companies with a vision to easily connect AV products based on open specifications
  - Can all CS be forced to share certain protocols?
    - Yes: DLNA products may only be certified, if they meet the protocol specifications.
  - Can all CS be influenced to share certain protocols?
    - Yes: B&O can propose extensions or new DLNA specifications through the DLNA alliance.

- **AirPlay** (Black box tag)
  - Is the core behaviour of a CS specified and public?

---

9 A more detailed descriptions of the CS’s openness characteristics can be found in D21.2
- No: the AirPlay specifications are private and patented by Apple.
  - Is it possible to determine all stakeholders?
    - No: only the business and legal stakeholders are visible.
  - Can all CS be forced to share certain protocols
    - No: B&O have no power over Apple’s protocols or business case.
  - Can all CS be influenced to share certain protocols?
    - No: AirPlay’s business case permit co-operation with B&O

- B&O CS (White box tag)
  - Business and technology aspect of the CS are controlled by B&O

Based on the Requirement Context view (RCV) for each of the CS Stakeholder Roles of the SoS (omitted for brevity), the Context Interaction View (CIV) is created using the ‘assemble SoS Context’ Activity from the Constituent System Identification process:

- ‘assemble SoS Context’, that combines the Requirement Context Views (represented by the ‘CS requirement context view’ parts) into the ‘Context interaction view’ for the SoS.

Figure 45 shows the CIV. Note the B&O CL SoS is added as an actor in the View. The SoS actor represents the SoS-level browsing needs and binds together the Constituent Systems of the SoS. The main purpose of CIV is direct and in-direct impact analysis. The SoS CIV identifies that the AirPlay and DLNA Constituent Systems indirectly impact on the SoS in a B&O context. The impact is through the Constituent Systems interaction with the ‘Device discovery’, ‘Service Discovery’ and ‘Service Requests’ Use Cases. These interaction relations indicate that the B&O SoS Architecture must address integration and management of AirPlay and DLNA based on their Interfaces and CS openness classifications.
The combination of the RCV and the CIV enables communication regarding both Requirements Context inconsistency and the SoS-level impact of the inconsistency among the Stakeholder Roles of the SoS. From the CIV we can see that the ‘AirPlay CS’ is involved in the ‘Device Discovery’, ‘Service Discovery’ and ‘Service Requests’ Use Cases, hence AirPlay meets the minimal requirements for the CB Scenario. But putting the same Use Cases into the AirPlay CS’s Requirement Context View shows that the Use Cases are redefined by Apple’s business case actor (the ‘Apple BC’ actor):

- Apple does not allow non-Apple products or non-partners to discover or browse Apple products.

This is a legal limitation caused by a hostile CS [COMPASS D21.2]. Because of this fact integration of AirPlay is not possible. AirPlay will not be further described in this document.

So far we have produced the Requirement Description View, Context Definition View and the Context Interaction View based on the integration framework and integration guideline Processes. The Views are all based around a common CL Ontology Definition View for consistent communication regarding the AV domain concepts. The following sections will describe the Architecture and AV concepts of each Constituent System of the SoS using the integration framework Views, Process guidelines and the CL Ontology Definition View.
10.3. The B&O AV Control Layer Architecture and AV Concepts

The CIV from Figure 45 has identified a set of interoperability challenges B&O are facing regarding integration of Constituent Systems for the CB Scenario. The main challenges are Device Discovery, Service Discovery and Service Requests. The following sections will give a more technical description of the AV concepts and the integration Views being produced for each Constituent System of the SoS.

**Device Discovery** (DD) refers to the process of identifying devices present on a network. The DD abstraction contains two logical mechanisms and processes:
- device announcement mechanism (DAM)
- device discovery mechanism (DDM (uses DAM))

**Service Discovery** (SD) refers to the process of identifying services provided by devices present on a network. The SD abstraction contains two logical mechanisms and processes:
- service announcement mechanism (SAM)
- service query mechanism (SQM)

**Service Requests** (SR) refers to the process of calling services provided by devices present on a network. The SR abstraction contains two logical mechanisms and processes:
- service call mechanism (SCM)
- service reply mechanism (SRM)

DD, SD and SR Interfaces form the heart of the control layer logic in most AV Architectures. Please note there are no constraints for asynchronous or synchronous service mechanism semantics for any of the Constituent System CL Interfaces in B&O's CL minimal Requirements from Figure 43. For simplicity we will only focus on integration of the Service Requests mechanisms for the content browsing service for each CS of the SoS. The minimal Requirements for a content browsing service refers to the capability of browsing media content owned by a device present on a network, hence the browsing service is equal to the SoS CB Scenario. The sum of the Constituent Systems CL services are often referred to as the network interfaces of the CL Architecture or the network enabled services of the CL Architecture. In the CL Ontology network interfaces are defined as application layer Network Service Interfaces. The next section will describe the Interfaces for B&O's CL Architecture.

**B&O AV Control Layer Architecture Model**

The B&O CL has been designed with these generic AV concepts in mind:
- **Discovery**: A product should be able to detect the presence (or absence) of other products (DD).
- **Service Discovery**: A product should be able to detect services of other products (SD).
- **Requests**: A product should be able to request actions or data from other products (SR).
- **Events**: A product should be able to listen to asynchronous events from other products (might be synchronous replies or status events) (SRM).
- **Power**: A product should be able to reduce power consumption by entering a sleeping state, and other products must be able to wake up the sleeping product when needed.

Using the ‘define interface structure’ Activity from the integration Process defines the following B&O DD, SD and RS interface elements. The IDV view is show on Figure 46

Most of the IDV elements on the IDV are stereotyped with Ontology Elements from the ODV in Figure 41. The Ontology tags ensure better understanding of the AV concepts the elements on the IDV represent. The next two sections will describe the Interface structures on the IDV.

**B&O Device Discovery (DD) and Service Discovery (SD)**

The DD and SD of B&O’s CL are based on Bonjour\(^{10}\). Bonjour provides a general method to discover services on a local area network. B&O use the Bonjour method to implement proprietary DD and SD services. The proprietary design means B&O devices and B&O services cannot be discovered or used by non-B&O devices. The ‘DD_SD_Announcement’ Interface used by the ‘DD_SD_Service’ application layer interface is flow-based. A ‘Device_Record’ data structure

---

\(^{10}\) Bonjour is Apple’s implementation of Zero configuration networking (Zeroconf), a group of technologies that includes service discovery, address assignment, and hostname resolution. Bonjour locates devices such as printers, other computers, and the services that those devices offer on a local network using multicast Domain Name System (mDNS) service records.
containing service and device information are broadcasted to the network using an ‘IP Multicast’ Network Transport Strategy.

**B&O CL Service (SR)**

All B&O services follow the same blueprint. Each service has an application layer client-side proxy and a server-side stub. The B&O CL service technology is based on an object-oriented RPC design like Java’s RMI (Remote Method Invocation) framework. The service proxy ‘Browsing Interface’ is exposed on the network using the ‘DD_SD_Announcement’ interface. The ‘Browsing Request’ interface is marshalling and demarshalling ‘ServiceCallData’ over the network using a ‘IP TCP’ transport strategy. The application layer service operation signatures are IDL defined; meaning the type system and service compatibility are defined at compile time. The B&O CL IDL is proprietary: hence the ‘ServiceCallData’ binary data transmitted are useless without access to B&O’s IDL compiler. ‘The ContentData’ type are the application layer’s return values from a client-side proxy invoke of the queryContent function.

Based on the interfaces identified from the IDV, the Interface Identification View (IDV) is produced for the B&O Architecture. The production of the IIV is the first step of executing the ‘identify interfaces between CS’ operation from the Constituent System Identification process:

- ‘identify interfaces between CS’, where the Interface Identification Views, represented by the ‘CS interface’ *part*, for each Interface between two Constituent Systems are defined. These are then combined to form a collection of these Interface Identification Views, using the ‘Interface set’ *part*.

Figure 47 shows the Interfaces for B&O’s CL. Only the ‘DD_SD_Announcement’ and ‘Browsing Request’ Interfaces are showed on the IIV. The application layer interfaces are not relevant regarding CB integration. Only interfaces exposed on the IP network are relevant. The ‘DD_SD_Announcement’ and ‘Browsing Request’ Interfaces are a subset of the SoS Interface set, which also contain the Interfaces for the DLNA CL Architecture. The Interface Identification View for DLNA will be presented in section 10.4.

---

**Figure 47 Interface Identification View for B&O’s CL Architecture**

---

11 a remote procedure call (RPC) is an inter-process communication that allows a computer program to cause a subroutine or procedure to execute in another address space
The Interface Connectivity View for B&O’s CL defines how the Interfaces identified in the Interface Definition View are connected together. The ICV is shown in Figure 48. On the ICV CL Ontology Elements are used for describing the role concepts of the service user and service provider. The ‘B&O Target Product’ calls services on the ‘B&O Source Product’. The calls are transmitted as ‘ServiceCallData’ data types between the roles. The ‘B&O Target Product’ discovers the ‘B&O Source Product’ and its services through the ‘Device_Record’ broadcasted by the ‘B&O Source Product’. The SysML connectors show the concrete transport strategies (‘TS’) used for sending the ‘ServiceCallData’ and ‘Device_Record’ data types over the IP network. Modelling the transport logic is important for understanding the interaction between the CL roles. The interaction will defined parts of the integration design.

The Integration Process contain the ‘define interface behaviour’ Activity for detailed modelling of the interaction between CS System Elements across their Interfaces.

The Activity states:

- ‘define interface behaviour’, where the interactions between the System Elements and across the selected Interface are defined by creating the ‘Interface behaviour view’.

Figure 49 shows the IBV for the interaction regarding device and service discovery elements of the B&O CL Architecture. On the IBV only the B&O source product role and the transport layer are modelled because of the multicast communication pattern used by the ‘Browsing Request’ service interface. The IBV models all ‘Layer’(s) of the ‘CallStack’ from Figure 42.
The IBV for the SR part of the B&O CL Architecture is shown in Figure 50. The SR IBV does not show the application layer System Elements since they are not relevant from an integration point of view. We are only focusing on the interaction regarding transport of ‘ServiceCallData’ data and the product roles in a service call sequence. On the IBV the ‘Network Service Interface’(s) simple represent the application layers triggering of and response to role in a network call sequence.

As an output from the ‘define protocol’ Activity from the Integration Process we are modelling the Protocols for the B&O CL architecture using CML. We wish to simulate and verify the final integration of the Constituent Systems using the COMPASS tool chain. To enable COMPASS tool chain verification the CML Protocol Definition Views are created by translating the SysML diagram elements to CML processes and data types. The translation is done by using the guidelines described in Section 8.2 and a pattern level translation meta-model [COMPASS 22.5]. The pattern level translation meta-model is a semantics function, which map pattern views and pattern ontology elements to CML models and CML
elements. A set of translation rules provides domain specific semantics to the CML models. The translation rules are applied at different refinement points [COMPASS 22.5] in the pattern level translation meta-model. Figure 51 shows the high level meta-model view for translation of the Interface Pattern ontology elements [COMPASS 22.3] to CML elements.

In the meta-model, IP domain semantics are given to the Interface Pattern by sub-typing the ‘Port’ element with IP domain port types like ‘UDP’, ‘Multicast’ and ‘TCP’. Each ‘IP’ port type has a CML translation rule. As an example, the CML translation rules for the ‘RR04’ ‘IP’ refinement points [COMPASS 22.5] states:

- If ‘Port’ p is of the type ‘Multicast’ then p shall be represented as one CML channel x
  - The channel x shall be named: pre-fix IPMulticast + nameof(p) post-fix with channel, hence the port has no IP connection logic.
- If ‘Port’ p is of the type ‘TCP’ then p shall be represented as two CML channels x and y
  - One CML channel x representing the TCP connect/disconnect logic. The channel shall be pre-fix with IPTCPPORT + nameof(p) and post-fix with channel.
  - One CML channel y representing the TCP data and flow control (y is interruptible by x). The channel shall be pre-fix with IPTCP+ nameof(p) and post-fix with channel.

It’s beyond the scope of the document to give a detailed description of the pattern level translation meta-model, so only behavioural translation rules relevant for CML PDV models are described in the following sections. The ‘define protocol’ operation of the Integration Process states:

- ‘define protocol’, where the Protocol, if required, is defined for each Interface by creating the ‘Protocol definition view’.

Figure 51 high level meta-model view for CML translation
Figure 53 shows the PDVs for the DD and DS part of the B&O CL Architecture. In the model each CML process represents one PDV and one PDV represent a CI product roles. In the CML model, SysML connectors and ports are translated to CML channels according to the translation rules. The PDVs shows both product roles are being powered on by an event on their control layer channel. The ‘SourceProductPowerchannel’ and ‘TargetProductPowerchannel’ on the PDVs represent these channels. The Source product’s SysML PDV state machine diagram, which the CML PDV is translated from, are showed in Figure 52. In the CML PDV, SysML states are translated to CML actions post-fixed with state.

![Figure 52 SysML Protocol Definition View for B&O’s DD and SD interfaces](image)

The products communicate using the ‘IPMulticastChannel’ channel. The ‘IPMulticastChannel’ represents the transport layer logic according to the translation rules for the ‘Port’ refinement points. In the PDVs we simulate the asynchronous behaviour of the IP multicast logic by using CML timeouts for breaking channel synchronization between the processes. Note that at any time in the announcement and query operation state, the products can be interrupted by power-off events on their control channels.

![Figure 53 CML Protocol Definition View for B&O’s DD and SD interfaces](image)

The interruptible part of the CML model is translated from a SysML activity diagram. The activity diagram is a refinement of the announcement operation in the ‘DD_SD_Announcement_State’. The activity diagram is given in Figure 54.
Figure 54 refinement of the announcement operation

Figure 55 shows the PDV for the SR part of the B&O CL Architecture. The service call semantics are using the following channels: The source product waits on service calls coming from the 'IPTCPChannel' transport channel in the form of 'ServiceCallData' data structures. Service replies are sent back over the same TCP channel. Service calls for both AV roles can be interrupted from the application layer via the 'INTERUPT' event. The 'INTERUPT' event is sent over the 'SourceProductEventChannel' and 'TargetProductEventChannel' channels or by a 'DISCONNECT' event on the 'IPTCPORTChannel'. These channels represent the interaction between the application layer and the transport layer of the Architecture. The application layer service call semantics are synchronous, but can timeout because of the TCP transport strategy. The CML timeout operator used in the Views represents the TCP round trip timeout. The timeout, interrupt and power on semantics in the PDV CML models represent the IP dynamicity of the SR protocols.

Figure 55 CML Protocol Definition View(s) for B&O's SR system elements and interfaces

The Views needed for B&O’s Constituent System are now complete; the next section will model the DLNA Views and thereby complete the ‘identify interfaces between CS’ activity. For simplicity only a subset of the integration Views will be presented. Since B&O only wish to integrate DLNA as a source product, only Views showing the source product roles for the DLNA Architecture and business concepts are modelled and described in the following section.
10.4. The DLNA AV Control Layer Architecture

The DLNA allows devices to share media (music, video and pictures) with other devices regardless of who the manufacturer is or what platform the devices are running on. The standard also ensures that consumers don't have to bother with configuring a network to get the DLNA-certified devices to work together. Once a DLNA-enabled device is present on the network, the others will be able to access any content that resides thereon or otherwise use their special capabilities. DLNA's interoperability capabilities are based on the UPnP standard: hence in this section DLNA and UPnP references to the same technology. DLNA also represents an organisation which controls and owns the technology. From a SoS perspective DLNA is a Constituent System representing both organisational and technology stakeholders (as described on page 77).

The DLNA AV Architecture consists of 3 main System Element concepts:

- **Devices**: Any entity (hardware or software) on the network that communicates to other entities on the network using the UPnP-defined communication protocol.
- **Services**: Each **Device** offers zero or more **Services**. A **Service** is an aggregation of zero or more **Actions** which **Control Points** can invoke. The UPnP working group has defined **Services** for some of the **Devices** that exist in the market place but for those **Devices** for which the **Services** have not been defined, the manufacturers have the freedom to define their own **Services**.
- **Control Points**: An entity on the network that discovers, and invokes **Service Actions** offered by **Devices**. A **Control Point** may subscribe to events provided by a **Service** and invoke **Actions** provided by the **Service**.

The DLNA specification explicitly defines the following goals for DLNA:

- To support arbitrary transfer protocols and content formats.
- To enable the AV content to flow directly between **Devices** without any intervention from the **Control Point**.
- To enable **Control Points** to remain independent of any particular transfer protocol and content format. This allows **Control Points** to transparently support new protocols and formats.
- Scalability, i.e. support of **Devices** with very low resources, especially memory and processing power as well as full-featured **Devices**.

The AV Target and Source Roles are defined in following ways in DLNA:

- **Media Server**: The media server typically stores the content – in this case, audio files that the end user would like to render on the media renderer.
- **Control Point** Application: A **Control Point** Application allows an end-user to undertake a particular task/operation enabled by the **Media Server**.
- **Media Renderer**: The Media Renderer is responsible for rendering of the media received from the Media Server. The Media Renderer receives commands from the Control Point.

Only the DD, SD and SR part of the DLNA Architecture is described in the following sections. The streaming operations are not part of the control layer in the AV Architecture and are not part of our focus areas regarding the CB Scenario, hence we are not going to model the DLNA Media Renderer.

**DLNA: Simple Service Discovery Protocol (SSDP)**

SSDP is a network protocol based on the Internet Protocol Suite for advertisement and discovery of network services and presence information. It accomplishes this without assistance of server-based configuration mechanisms, such as the Dynamic Host Configuration Protocol (DHCP) or the Domain Name System (DNS). SSDP is a text-based protocol based on HTTPU. It uses the User Datagram Protocol (UDP) as the underlying transport protocol. Services are announced by the hosting system with multicast addressing a specifically designated IP multicast address at UDP port number 1900. The semantics of DD and SD of SSDP are packet structures communicated over IP networks implying the communication paradigm is flow based. SSDP are used by all DLNA enabled AV products.

**DLNA: CL Service Request (SR)**

The DLNA-enabled applications and devices go through the following process to invoke a service offered by the application or device.

When a device is connected to the DNLA network, the device initiates a discovery process. During the discovery process, a control point queries the network using the Simple Service Discovery Protocol (SSDP) and identifies devices that are hosting various services. Upon query by the control point, the device offering a particular service responds to the control point by offering a URL of the device description and services offered by the device. The control point can interrogate the XML-based device description document to identify the device maker, device model and services offered by the device. A detailed description of various services – input and output parameters are contained in the XML-based service description document. By having access to the various services offered by the device, the control point can invoke actions on one or more services using the SOAP protocol.

The SR communication paradigm for DLNA is flow based. Service calls are transported over the IP network using the HTTP application layer protocol. The SR data are described as SOAP messages. The DLNA browsing service is named the Content Directory Service. Using the Content Directory Service, the media server allows a control point to search for a particular content residing in the content directory. The content directory service upon query by the control point returns meta-data associated with the content. The media server also informs the control point with the data format of the content as well as the communication protocol supported by the media server. In the DLNA domain
SR are referred to as service actions. Figure 56 shows the Interface Definition View for the DD, SS and SR part of the DLNA architecture.

The IDV basically represents the DLNA media server, which expose the ‘Content Directory Service’ interface on the network, using the ‘SSDP_Announcement’ interface. The application layer ‘ContentQuery’ function can be invoked using ‘SOAP’ messages, transmitted over the IP network using the ‘HTTP’ transport strategy. The ‘Content Request’ interface is responsible for the marshalling service calls on the IP network. The ‘SSDP_Service’ use the ‘SSDP_Announcement’ interface for product and service announcements in the IP network via the ‘HTTPU’ transport strategy. We can conclude from the IDV that DLNA satisfies B&O’s minimal requirements for integration as a CL CS. DLNA provides the required DD, DS and SR concepts, which after integration will satisfy the SoS-level CB Requirement. The Interface Connectivity View for DLNA’s CL is shown in Figure 57.

B&O will not expose any contents to other non-B&O products using DLNA technology. So only the SSDP and content directory service interfaces are
relevant for integration regarding the CB Scenario. To gain knowledge regarding the Interface semantics, the Protocol Definition Views are produced. The PDV for the DD and DS part of the DLNA AV architecture are displayed in Figure 58 as CML processes.

The DLNA Protocols are very similar to the Protocols of B&O’s DD and SD Interfaces. Both CS’s DD and SD Interfaces are flow-based and are using multicast transport strategies for data communication. The difference lies more in the data structure format and the level of the Internet protocol suite across which the data transport are taking place. DLNA uses IP application layer protocols where B&O is using IP transport layer protocols like UDP and TCP. Figure 59 shows the Protocol Definition View for the SR part of the DLNA CL Architecture. The DLNA PDVs are produced using the pattern level translation meta-model translation rules. The SR call semantics has some equality to B&O’s SR call semantics. It is common that AV network services operate using the same client/service and call/reply network communication design patterns.
part of the guidelines. The next section will describe the achievements of using the Validation Process and suggest some candidate modifications to B&O’s SoS Architecture for integration of DLNA.

### 10.5. Integration Challenges between DLNA CS and B&O CS

By applying the Validation Process Activities, a set of actual integration-level challenges can be identified from the SoS structural and behavioural views. The Interface Identification Views and Interface Connectivity Views for B&O and DLNA shows that the Constituent Systems contain the same DD, DS and SR System Elements, but are using different data types and IP layer transport strategies. The data types differ in form between B&O’s static IDL binary format versus DNLA’s more dynamic text-based data exchange format. The data exchange mismatch means integration of DLNA’s data types requires data mapping to and from B&O’s static IDL types at runtime. This indicates B&O will have problems handling evolution of DLNA data types. The B&O type system is defined at compile time, where DLNA types can be defined at runtime. The proxy design pattern B&O makes use of for SRs is not to be considered an Architectural mismatch problem for integration of DLNA’s SR flow-based design[COMPASS 22.3]. It simply requires a B&O client-side proxy representing the DLNA source product. This client-side proxy will expose the same control application layer Interface as other IDL proxies in the B&O CL Architecture. Regarding DLNA Content Directory Service evolution, the B&O proxy needs only to support the search logic of the browse SOAP messages. The DLNA browse action SOAP messages are mandatory for all DNLA enabled products and are not changing according to the DLNA specification. The DLNA IP application layer transport strategies must be supported in the B&O architecture by a wrapper integration design. B&O target product’s ‘DD_SD_Service’ and ‘Browsing Service’ System Elements must be extended with ‘HTTPU’ and ‘HTTP’ transport strategies and data type mapping operations.

The Protocol Definition Views and the Interface Behaviour Views shows that the DD and SD operation of the Constituent Systems are equivalent. Both Constituent Systems are using multicast strategies for announcement of products and services on the network. B&O only need to “listen” to the ‘HTTPU’ channel for discover DLNA products. The Constituent Systems SR call semantics are based on the same IP request/reply and source/target role concepts. Integration of DLNA for B&O means translating and transmitting B&O SR calls over the ‘HTTP’ protocol in the form of SOAP messages. The fact that DLNA Protocols are standard IP stack Protocols handles the evolution aspect of integrating DLNA. The Protocol Definition View in Figure 60 shows a simplified version of B&O’s ‘DD_SD_Service’ Interface after integration of DLNA.
Figure 60 B&O DD and SD CML Protocol Definition View(s) supporting DLNA

The DD and SD PDV shows the ‘DD_SD_Service’ Interface now contains a DLNA ‘SSDP_Discovery_State’ CML action, which is handling the discovery of DLNA products as a interleaving composition with the ‘BEO_DD_SD_Discovery_State’ action. The DLNA discovery logic is using the DLNA ‘HTTPUTransportChannel’ transport strategy and ‘Device_Description’ data type.

The Validation Interaction View in Figure 61 is the result of combining the IBVs of B&O and DLNA. In the VIV the IBVs are mapped to CML processes. The IBV processes are used for validating the SoS. The ‘SoS_Validation_View’ SoS process is constructed from the B&O and DLNA DD and SD CS processes from Figure 60, and the ‘SoSDeviceDiscoveryAndServiceDiscovery_Validation_View’ process. The ‘SoSDeviceDiscoveryAndServiceDiscovery_Validation_View’ represent the SoS-level CB Scenario. The VIV is part of the ‘perform analysis’ Activity from the integration Validation Process:

- ‘perform analysis’, that performs the actual validation by comparing the set of ‘Interface behaviour views’ for the Interfaces to the ‘Validation interaction views’ for the SoS.

Figure 61 CML Validation Interaction View for B&O and DLNA CSs

The COMPASS tool chain can verify if the ‘Beo_DD_SD_InterfaceProtocolViews’ on
the VIV is a valid refinement of the ‘SoSDeviceDiscoveryAndService-Discovery_Validation_View’. The COMPASS tool chain also allows simulations of the CL SoS against the SoS-level CB Scenario by running the ‘SoS_Validation_View’ SoS level process. The VIV is the last View produced in this B&O integration application. The following section will describe the achievements of applying the integration guidelines to the B&O CL case study.

10.6. Summary

In the technology domain the integration challenges regarding DLNA and the SoS-level contents browsing Use Case are limited. Most of the challenges consist of data type mapping and supporting high level IP Protocols for communication between products. The integration Views help in the identification of different integration points for B&O’s CL Architecture and needed Architectural modifications. A candidate SoS Architecture model was developed based on knowledge from the Integration Views and Processes. The model was used for verifying the correctness of the architectural modifications needed for supporting DLNA in B&O’s SoS.

In the AV business domain the CS openness of DLNA is not impacting the integration for B&O. The Context Interaction View shows that B&O and DLNA Stakeholder Roles agree on the same CB Use Cases, meaning the CB Requirements are consistent even in different CS Stakeholder Contexts. The Context Interaction View also helped identify that AirPlay cannot be integrated because of legal issues. The integration guidelines did not only help integration at the technical level, but also at the business level. B&O lives in a competitive industry; where the CSs often are hostile. Early verification of a hostile CS’s impact on the B&O SoS are cheaper than real life lawsuits.
11. Conclusions

This report has presented a Framework and set of Processes for the integration of Constituent Systems into a SoS. This work is based on an analysis of current best practice guidelines. It also includes a study of relevant data exchanges standards and technologies that are important in integration.

The report has taken a model-based approach, building on existing Frameworks and patterns described in [COMPASS D21.1] and [COMPASS D22.3] to define an Integration Ontology, Framework and Viewpoints. Based on these, a set of Processes was defined for integration for SoS. A model-based approach to process modelling, known as the ‘seven views’ approach was used to specify these processes.

The integration Framework and Processes were exercised by B&O, one of the COMPASS industrial partners, as part of their COMPASS case study work. In order to further ensure the validity and applicability of the integration Framework and Processes described in this report they will be made available to users so that they can be executed and user comments and feedback gathered to allow the processes to be improved as necessary based on such feedback.

During the lifetime of the COMPASS project the integration Framework and Processes will continue to be used by the industrial partners as part of their case study work investigating COMPASS methods in tasks T4.1.1 and T4.2.2, and as part of the industrial challenge problem in task T4.3.3. The Framework and Processes will be disseminated to other COMPASS members and to members of the COMPASS Interest Group (CIG) by means of Atego Process Director (APD), a web-based tool supporting the capture and dissemination of processes which also supports the capture of user feedback.

The report has also discussed the use of SysML and CML and the COMPASS Tool to support integration. While promising, the COMPASS Tools are not yet at a stage where the proposed approach can be tested.
12. References


[OMG-CORBA-Std2012] Common Object Request Broker Architecture, OMG standard version 3.3 November 2012,


