Project: COMPASS
Grant Agreement: 287829

Comprehensive Modelling for Advanced Systems of Systems

Final Report on Guidelines for SoS Engineering

Document Number: D21.6

Date: September 2014

Public Document

http://www.compass-research.eu
Contributors:
Jon Holt (JDH), ATEGO
Simon Perry (SAP), ATEGO
Finn Overgaard Hansen (FOH), IHA
Alvaro Miyazawa (AM), UY
Klaus Kristensen (KK), B&OA
Ralph Hains (RH), ATEGO

Editors:
Simon Perry, ATEGO

Reviewers:
J. Bryans, UNUT
S. Hallerstede, AU
## 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>2014-09-08</td>
<td>SAP</td>
<td>First complete version for initial review</td>
</tr>
<tr>
<td>1.1</td>
<td>2014-09-30</td>
<td>SAP</td>
<td>Final version for issue</td>
</tr>
</tbody>
</table>
Abstract
This deliverable contains a summary of the COMPASS approach to System of Systems engineering (SoSE). The COMPASS approach consists of two key parts, the COMPASS Architectural Framework and the COMPASS Process Library. Each of these are described, bringing together the relevant Ontologies, Frameworks and Process from contributing COMPASS deliverables. A number of Scenarios are discussed showing how the various elements of the approach are used in the development or maintenance of a System of Systems (SoS). An extended discussion of the use of the COMPASS approach on an industrial case study is also presented. In order to successfully carry out SoSE, it is important to have people with the correct skills. A Competency Ontology, Framework and Processes are presented, along with a number of Competency Scopes that cover the Stakeholder Roles involved in all of the COMPASS SoSE Processes defined to date.
# Table of Contents

1. Introduction .................................................................................................................. 11
   1.1. Scope ..................................................................................................................... 11
   1.2. Context .................................................................................................................. 11
   1.3. Writing Convention ............................................................................................. 12
   1.4. Existing Best Practice Guidelines ...................................................................... 14
   1.5. Compliance .......................................................................................................... 15
   1.6. Document Structure ........................................................................................... 15

2. The COMPASS Approach .......................................................................................... 16
   2.1. The COMPASS Ontology .................................................................................. 17
   2.2. The COMPASS Architectural Framework ....................................................... 19
   2.3. The COMPASS Processes and Guidelines ......................................................... 23
      2.3.1. SoS Requirements Processes ....................................................................... 23
      2.3.2. Architectural Framework Processes ........................................................... 27
      2.3.3. Architecture Guidelines ............................................................................. 30
      2.3.4. Refinement Process .................................................................................... 33
      2.3.5. System Integration Processes ..................................................................... 37
      2.3.6. Competency Processes ............................................................................. 40

3. Application of the COMPASS Approach ................................................................ 41
   3.1. Life Cycles and Life Cycle Models .................................................................... 41
   3.2. Application to SoS Development ....................................................................... 44
   3.3. Application to SoS Maintenance ........................................................................ 46
   3.4. Application to Retirement of CSs ...................................................................... 48
   3.5. Conclusions ........................................................................................................ 49

4. Competence and SoSE ............................................................................................... 50
   4.1. The COMPASS Competency Framework ......................................................... 51
      4.1.1. AF Context View ........................................................................................ 51
      4.1.2. Ontology Definition View .......................................................................... 52
      4.1.3. Viewpoint Relationships View .................................................................. 54
      4.1.4. Rules Definition View ................................................................................ 55
      4.1.5. Viewpoint Definitions ................................................................................ 56
   4.2. The Competency Processes ................................................................................ 66
      4.2.1. The Requirement Context View .................................................................. 66
      4.2.2. The Stakeholder View .............................................................................. 68
      4.2.3. The Process Structure View ..................................................................... 69
      4.2.4. The Process Content View ........................................................................ 69
      4.2.5. The Process Behaviour Views .................................................................. 73
      4.2.6. The Information Views .............................................................................. 78
      4.2.7. The Process Instance Views ..................................................................... 81
   4.3. Example Competency Scopes ............................................................................. 82
      4.3.1. Configuration Manager Competency Scope .............................................. 83
      4.3.2. Requirement Manager Competency Scope ............................................... 84
      4.3.3. Project Manager Competency Scope ......................................................... 85
      4.3.4. Requirement Engineer Competency Scope .............................................. 86
      4.3.5. Systems Modeller Competency Scope ...................................................... 87
      4.3.6. Tester Competency Scope ....................................................................... 88
      4.3.7. Reviewer Competency Scope ................................................................... 89
      4.3.8. Process Modeller Competency scope ...................................................... 90
      4.3.9. Builder Competency Scope ..................................................................... 91
Figures

Figure 1 - Overview of the COMPASS Approach .......................................................... 16
Figure 2 - The full COMPASS SoSE Ontology ............................................................ 18
Figure 3 - Viewpoint Relationships View for the COMPASS Architectural Framework .......................................................... 20
Figure 4 - Viewpoint Relationships View for the COMPASS Architectural Framework showing overlapping packages to highlight shared Viewpoints …… 21
Figure 5 - Process Content View showing COMPASS Process Library Process Groups ......................................................................................................................... 23
Figure 6 - Process Content View for System of Systems Requirements Engineering Process Group Processes …................................................................. 24
Figure 7 - Process Content View for System of Systems Requirements Management Process Group Processes …............................................................... 25
Figure 8 - Process Content View for Architectural Framework Group Processes …........ 28
Figure 9 Requirement Context View for the Refinement Process ….................................. 33
Figure 10 Stakeholder View for the Refinement Process ….............................................. 34
Figure 11 Process Content View for the Refinement Process …....................................... 35
Figure 12 Process Behaviour View for the Refinement Process …................................... 36
Figure 13 Information View for the Refinement Process ….............................................. 37
Figure 14 - Process Content View for Integration Technical Group Processes ……… 38
Figure 15 - Process Content View for Integration Management Group Processes ……… 38
Figure 16 - Example Life Cycle - the Stages of ISO15288:2008 …................................. 41
Figure 17 - Typical Life Cycle Stages covered by COMPASS approach ….......................... 42
Figure 18 - Typical Life Cycle Stages covered by COMPASS approach for an existing System …........................................................................................................... 43
Figure 19 – COMPASS Processes executed during the Conception Stage for the development of a new System …........................................................................ 44
Figure 20 – COMPASS Processes executed during the Conception Stage for the development of an existing System …........................................................................ 45
Figure 21 – COMPASS Processes executed during the Development Stage ………………… 45
Figure 22 – COMPASS Processes executed during the Production Stage ………………… 46
Figure 23– Overall Life Cycle Model for a maintenance Project with change detected during Support Stage ................................................................. 47
Figure 24– COMPASS Processes executed during the Support Stage ………………… 47
Figure 25 Overall Life Cycle Model for a Project where a Constituent System is retired during the Support Stage …........................................................................ 48
Figure 26 COMPASS Processes executed during the Support Stage ………………… 49
Figure 27 - AF Context View for the COMPASS Competency Framework ….............. 51
Figure 28 - Ontology Definition View focussing on Competence …............................... 53
Figure 29 – Viewpoint Relationships View for the COMPASS Competency Framework …........................................................................................................... 54
Figure 30 - Rules Definition View for the Competency Framework ….......................... 55
Figure 31 - Viewpoint Context View for the Competency Framework Definition Viewpoint …........................................................................................................... 56
D21.6 - Final Report on Guidelines for SoS
Engineering (Public)

Figure 32 - Viewpoint Definition View for the Competency Framework Definition Viewpoint

Figure 33 - An example Competency Framework Definition View

Figure 34 - Viewpoint Context View for the Competency Level Definition Viewpoint

Figure 35 - Viewpoint Definition View for the Competency Level Definition Viewpoint

Figure 36 - An example Competency Level Definition View

Figure 37 - Viewpoint Context View for the Competency Scope Definition Viewpoint

Figure 38 - Viewpoint Definition View for the Competency Scope Definition Viewpoint

Figure 39 - An example Competency Scope Definition View

Figure 40 - Viewpoint Context View for the Competency Profile Definition Viewpoint

Figure 41 - Viewpoint Definition View for the Competency Profile Definition Viewpoint

Figure 42 - An example Competency Profile Definition View

Figure 43 - The RCV for the Competency Processes

Figure 44 - The SV for the Competency Processes

Figure 45 - The PSV for the Competency Processes

Figure 46 - The PCV for the Competency Processes

Figure 47 - The PBV for the Bespoke Competency Definition Process

Figure 48 - The PBV for the Bespoke Framework Definition Process

Figure 49 - The PBV for the Assessment Set-up Process

Figure 50 - The PBV for the Assessment Process

Figure 51 - IV for the Competency Processes

Figure 52 - The IV for the Bespoke Competency Definition Process

Figure 53 - The IV for the Bespoke Framework Definition Process

Figure 54 - The IV for the Assessment Set-up Process

Figure 55 - The IV for the Assessment Process

Figure 56 - PIV for "Support definition of bespoke competency frameworks"

Figure 57 - PIV for "Support carrying out competency assessments"

Figure 58 Competency Scope for ‘Configuration Manager’ Stakeholder Role based on the INCOSE Systems Engineering Competencies Framework

Figure 59 Competency Scope for ‘Requirement Manager’ Stakeholder Role based on the INCOSE Systems Engineering Competencies Framework

Figure 60 Competency Scope for ‘Project Manager’ Stakeholder Role based on the INCOSE Systems Engineering Competencies Framework

Figure 61 Competency Scope for ‘Requirement Engineer’ Stakeholder Role based on the INCOSE Systems Engineering Competencies Framework

Figure 62 Competency Scope for ‘Systems Modeller’ Stakeholder Role based on the INCOSE Systems Engineering Competencies Framework

Figure 63 Competency Scope for 'Tester' Stakeholder Role based on the INCOSE Systems Engineering Competencies Framework

Figure 64 Competency Scope for ‘Reviewer’ Stakeholder Role based on the INCOSE Systems Engineering Competencies Framework

8
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 65</td>
<td>Competency scope for ‘Process Modeller’ Stakeholder Role based on the INCOSE Competencies Framework</td>
</tr>
<tr>
<td>Figure 66</td>
<td>Competency Scope for ‘Builder’ Stakeholder Role based on the INCOSE Systems Engineering Competencies Framework</td>
</tr>
<tr>
<td>Figure 67</td>
<td>Competency Scope for ‘SoS Engineer’ Stakeholder Role based on the INCOSE Systems Engineering Competencies Framework</td>
</tr>
<tr>
<td>Figure 68</td>
<td>The COMPASS approach</td>
</tr>
<tr>
<td>Figure 69</td>
<td>The B&amp;O COMPASS Approach based development process</td>
</tr>
<tr>
<td>Figure 70</td>
<td>B&amp;O SoS level requirement description view</td>
</tr>
<tr>
<td>Figure 71</td>
<td>The SoS level user-experience requirements for “Audio-Visual Streaming”</td>
</tr>
<tr>
<td>Figure 72</td>
<td>B&amp;O SoS level Source Element View</td>
</tr>
<tr>
<td>Figure 73</td>
<td>DRSV Streaming Rule(s)</td>
</tr>
<tr>
<td>Figure 74</td>
<td>subset of the Streaming RDV</td>
</tr>
<tr>
<td>Figure 75</td>
<td>Streaming CDV</td>
</tr>
<tr>
<td>Figure 76</td>
<td>Source CS RCV</td>
</tr>
<tr>
<td>Figure 77</td>
<td>Streaming SoS CIV</td>
</tr>
<tr>
<td>Figure 78</td>
<td>Source CS VV</td>
</tr>
<tr>
<td>Figure 79</td>
<td>Streaming SoS VIV</td>
</tr>
<tr>
<td>Figure 80</td>
<td>B&amp;O’s Streaming Architectural Framework’s AFCV</td>
</tr>
<tr>
<td>Figure 81</td>
<td>SAF’s Ontology Definition View</td>
</tr>
<tr>
<td>Figure 82</td>
<td>SAF’s Viewport Relationship View</td>
</tr>
<tr>
<td>Figure 83</td>
<td>Product Configuration VCV</td>
</tr>
<tr>
<td>Figure 84</td>
<td>“Product Configuration Viewpoint” VDV</td>
</tr>
<tr>
<td>Figure 85</td>
<td>NMM ICV</td>
</tr>
<tr>
<td>Figure 86</td>
<td>NMM’s IPV</td>
</tr>
<tr>
<td>Figure 87</td>
<td>SAF’s RDV</td>
</tr>
<tr>
<td>Figure 88</td>
<td>Test Pattern TCV instants</td>
</tr>
<tr>
<td>Figure 89</td>
<td>The Conceptual stage</td>
</tr>
<tr>
<td>Figure 90</td>
<td>LS Contract Definition View</td>
</tr>
<tr>
<td>Figure 91</td>
<td>AV SoS’s CDDV</td>
</tr>
<tr>
<td>Figure 92</td>
<td>SourceGraph CS Test Configuration View</td>
</tr>
<tr>
<td>Figure 93</td>
<td>Formal stage</td>
</tr>
<tr>
<td>Figure 94</td>
<td>The full COMPASS SoSE Ontology</td>
</tr>
<tr>
<td>Figure 95</td>
<td>Viewpoint Relationships View for the COMPASS Architectural Framework</td>
</tr>
<tr>
<td>Figure 96</td>
<td>Ontology Definition View focussing of SoS Requirements</td>
</tr>
<tr>
<td>Figure 97</td>
<td>Ontology Definition View focussing on Processes and Competency</td>
</tr>
<tr>
<td>Figure 98</td>
<td>Ontology Definition View focussing on Architectures and Architectural Frameworks</td>
</tr>
<tr>
<td>Figure 99</td>
<td>Ontology Definition View focussing on Integration</td>
</tr>
<tr>
<td>Figure 100</td>
<td>Ontology Definition View focussing on traceability</td>
</tr>
<tr>
<td>Figure 101</td>
<td>Ontology Definition View focussing on refinement</td>
</tr>
<tr>
<td>Figure 102</td>
<td>Viewpoint Relationships View for the COMPASS SoS-ACRE Requirements Framework</td>
</tr>
<tr>
<td>Figure 103</td>
<td>Viewpoint Relationships View for the “Seven Views” Process Framework</td>
</tr>
</tbody>
</table>
Figure 104 - Viewpoint Relationships View for the COMPASS Architectural Framework (CAFF). ................................. 163
Figure 105 - Viewpoint Relationships View for the Integration Framework. .... 164
Figure 106 - Viewpoint Relationships View for the Traceability Framework .... 164
Figure 107 - Viewpoint Relationships View for the Refinement Framework .... 165

Tables

Table 1 - Example sentences to illustrate the convention ........................................... 13
Table 2 - Summary of Best Practice Sources Used .................................................... 14
Table 3 - Compliance Information in Input Documents ............................................. 15
Table 4 - Summary of the COMPASS SoS Requirements Processes .................. 27
Table 5 - Summary of the COMPASS Architectural Framework Processes ........ 30
Table 6 - Summary of the COMPASS Architecture Guidelines ............................ 32
Table 7 - Summary of the COMPASS System Integration Processes .................. 40
Table 8 - Descriptions of Ontology Elements ....................................................... 149
Table 9 - Descriptions of Viewpoints ..................................................................... 152
Table 10 - Summary of the COMPASS Processes and Guidelines .................... 157
Table 11 - Original sources for Ontologies ............................................................. 157
Table 12 - Original sources for Frameworks ......................................................... 161
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, information on best practice guidelines and compliance, and an outline of the document structure.

1.1. Scope

This document presents the final report on the Guidelines for System of Systems (SoS) Engineering (SoSE). It completes the COMPASS approach initially described in [D21.3 2013], bringing together the Ontologies, Frameworks and Processes found in a number of COMPASS deliverables (see Section 1.2), combining them into an over-arching COMPASS approach that consists of the COMPASS Architectural Framework and the COMPASS Process Library.

A number of Scenarios are discussed showing how the various elements of the approach are used in the development or maintenance of an SoS. Consideration is given to the tool support that is provided for the dissemination of the approach, the modelling of a SoS through both SyML and CML and formal reasoning about models.

In order to carry out SoSE successfully, it is necessary to have people in place with the necessary skills. A Competency Ontology, Framework and associated Processes are presented, along with Competency Profiles and a number of Competency Scopes that cover the Stakeholder Roles involved in all of the COMPASS SoSE Processes.

One of the key aims of this deliverable is to show how the COMPASS approach can be used in practice. This is done through an extended discussion of its adoption on the B&O industrial case study.

1.2. Context

This report contains the final version of the COMPASS approach. It is intended to be an overview of the approach based on the following documents:

- Definition of the COMPASS Architectural Framework Framework [D21.5b 2014]
- Final Report on SoS Architectural Models [D22.6 2014]
- Report on Refinement Strategies for SoS Models [D22.5 2013]

This report brings together the content of the above documents into a summary of the approach, updating the material abstracted from those documents where necessary to address any inconsistencies that have resulted from advances in the research since the documents were originally issued. No additional content has
been added on the approach in this document, other than the material on Competence, covering the definition of Competency Frameworks and the carrying out of Competency assessments.

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 *Need* refers to the Ontology Element, whereas the same word without capitals – *need* – 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 ‘*Need*’ 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”.

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><em>Use Case</em> – 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><em>activity</em> – the everyday usage of the word</td>
</tr>
<tr>
<td></td>
<td><em>Activity</em> – the term from the MBSE Ontology</td>
</tr>
<tr>
<td></td>
<td><em>activity</em> – the term from the SysML notation</td>
</tr>
</tbody>
</table>
Example sentence | Meaning
---|---
The diagram here shows the ‘COMPASS Process Framework’ is made up of one or more ‘Process Behaviour View’ | ‘COMPASS Process Framework’ – a specific term from a specific diagram that is being described.

‘Process Behaviour View’ – a specific term from a specific diagram that is being described.

When defining Processes it is typical to create a number of *activity diagrams* that will visualise the Process Behaviour View. | Processes – the term from the COMPASS Ontology.

*activity diagram* – the term from SysML notation.

Process Behaviour View – the term from the COMPASS Framework

It is important to understand the “why” of MBSE | “why” – emphasis of the word

Table 1 - Example sentences to illustrate the convention

The table here 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” simple represents emphasis.
1.4. Existing Best Practice Guidelines

In developing the COMPASS approach, a number of best practice sources were consulted and used as significant inputs to the reports covering SoS Requirements engineering, Architectures and Architectural Frameworks and System Integration. A number of best practice sources were also consulted and used as inputs to the additional material on Competence and Competency Frameworks found in this report.

The best practice material abstracted from each is described in each input report. Table 2 provides a summary of the main sources used in each COMPASS report used as an input to this document. For details, see the relevant report.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SoS Requirements Engineering</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>[Lewis et al 2009]</td>
</tr>
<tr>
<td>Architectural Frameworks</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>[TRAK 2013]</td>
</tr>
<tr>
<td>Architectures</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>[Dickerson &amp; Mavris 2009]</td>
</tr>
<tr>
<td>System Integration</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>[D22.3 2013]</td>
</tr>
<tr>
<td>Refinement &amp; Evolution</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>[D21.5b 2014]</td>
</tr>
<tr>
<td>Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Competence (this document)</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>[APM 2013]</td>
</tr>
</tbody>
</table>

Table 2 - Summary of Best Practice Sources Used

In addition to the best practice sources explicitly used and referenced by the various reports that input into this report, COMPASS deliverable D11.2 “Convergence Report 2” [D11.2 2013] includes information on the current state of the common concept definitions which are being developed as the project Wiki¹; and an analysis of the common characteristics of the range of case studies

examined in the project. This provides essential background information to anyone engaging in SoSE. See Section 2 of [D11.2 2013] for further details.

1.5. Compliance

It is important that all Processes that make up the COMPASS approach map back to the best practice sources used as inputs to the work. Each of the previous deliverables that are used as inputs to this report are based on a number of best practice sources. Full details can be seen in the source document. In particular, see the sections shown in Table 3.

<table>
<thead>
<tr>
<th>SoS Engineering Area</th>
<th>Document</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoS Requirements Engineering</td>
<td>[D21.1 Appendix I 2012]</td>
<td>Sections 2.3 &amp; 2.5</td>
</tr>
<tr>
<td>Architectural Frameworks</td>
<td>[D21.2 2013]</td>
<td>Section 1.1</td>
</tr>
<tr>
<td></td>
<td>[D22.3 2013]</td>
<td>Sections 2.4.2 &amp; 2.5</td>
</tr>
<tr>
<td>System Integration</td>
<td>[D21.4 2013]</td>
<td>Section 9</td>
</tr>
</tbody>
</table>

Table 3 - Compliance Information in Input Documents

A complete compliance mapping from individual Processes to best practice sources will be presented in the final version of this document, D21.6 “Final Report on Guidelines for SoS Engineering”.

1.6. Document Structure

Following this introduction the document consists of the following main sections:

Section 2 gives an overview of the COMPASS approach, discussing the COMPASS Ontology, Framework and Processes.

Section 3 discusses the application, at a high-level, of the COMPASS approach to SoS development, SoS maintenance and the retirement of Constituent Systems.

Section 4 discusses competency in SoS engineering, presenting a competency Framework and associated Processes.

Section 5 covers the tool support for the COMPASS approach developed during the project.

Section 6 presents an extended discussion of the COMPASS approach through an example of its application to the B&O industrial case study.

Section 7 presents a concluding summary and Section 8 contains references.

Five appendices summarise the COMPASS Ontology, Framework and Processes. These are presented both in terms of the overarching approach and also in terms of the individual source Ontologies and Frameworks.
2. The COMPASS Approach

The COMPASS approach to SoSE is summarised in Figure 1.

The colour scheme in the "KEY" box on the diagram defines the four key elements of the approach:

- The “System” under development (colour blue). A “System” can be a “System of Systems” or a “ Constituent System”
- The “Approach” which covers the main concepts of ‘COMPASS Approach’ (colour green) used in the development of a ‘System’. This is described further below.
- The “Technique or Pattern”(s) that support and enable the approach (colour orange).
- The “Tool”(s) that enable the ‘COMPASS Approach’ (colour red).

The COMPASS approach consists of three key parts, the ‘Ontology’, the ‘COMPASS Architectural Framework’ and the ‘COMPASS Process Library’.

The ‘Ontology’ defines the concepts relevant to the approach, along with the relationships between these concepts.

The ‘COMPASS Architectural Framework’ is made up of one or more ‘Viewpoint’ and each Viewpoint uses elements from the Ontology to define a standard ‘View’ that can be produced as the visualisation of part of the Architecture of a ‘System’. One or more ‘Perspective’ collect together one or more related Viewpoint. The Ontology is further discussed in Section 2.1 and the Viewpoints in Section 2.2.
The ‘COMPASS Process Library’ is composed of one or more ‘Process’. Each Process addresses an aspect of SoSE and produces one or more Views that are defined by the Viewpoints). The Processes that make up the COMPASS approach are discussed further in Section 2.3.

Each Viewpoint of the COMPASS Architectural Framework is supported by zero or more ‘Enabling Pattern’. For information on Enabling Patterns, see [D22.6 2014]. Finally, the systems modelling language (SysML) and COMPASS modelling language (CML) can be used to realise the Enabling Patterns and the Viewpoints. An example of this can be seen in [D21.4 2013] where both SysML and CML are used to realise Integration Viewpoints.

2.1. The COMPASS Ontology

The COMPASS Ontology identifies and defines all of the terms and concepts that are used in the project and is a basis for the definition of the various Frameworks and supporting Processes that have been produced.

In the separate Guidelines produced, the Ontology Elements relevant to the subject under consideration, such as Requirements or Architectures, have been presented as a partial Ontology, albeit related one to another across deliverables. In this section, each of these Ontologies is brought together into a whole overarching COMPASS SoSE Ontology.

The six separate Ontologies are pulled together into a single Ontology in Figure 2 below. The diagram in Figure 2 is reproduced in Appendix I, which also contains a glossary of the Ontology Elements found on the diagram. It is not the purpose of this document to discuss the COMPASS SoSE Ontology; the original Ontologies can be found, for reference, in Sections 4 and 12, together with references to source documents that provide a detailed description of the Ontologies. Any inconsistencies discovered in the material presented in previous deliverables have been corrected in the full COMPASS SoSE Ontology presented here.

The COMPASS SoSE Ontology is intended to form a useful source of information for anyone engaged in SoSE, showing how the various areas covered by COMPASS relate one to another. It also ensures the consistency of the various Frameworks covered by the COMPASS Guidelines. These Frameworks are themselves brought together into a whole in Section 2.2.
Figure 2 - The full COMPASS SoSE Ontology
2.2. The COMPASS Architectural Framework

In the separate Guidelines produced, the Frameworks relevant to the subject under consideration, such as Requirements or Architectures, have been presented as separate Frameworks, albeit related one to another across deliverables. In this section, each of these Frameworks is brought together into the whole over-arching COMPASS Architectural Framework (COMPASS-AF). Note that this is different from the CAFF. The latter is an Architectural Framework (AF) for defining AFs; the former is an AF that has been developed using the CAFF.

The seven separate Frameworks are pulled together into a single Framework, the COMPASS Architectural Framework (COMPASS-AF) in Figure 3 below. It is not the purpose of this document to discuss the COMPASS-AF; the original source Frameworks can be found, for reference, in Sections 4.1 and 13, together with references to source documents that provide a detailed description of the Viewpoints that make up the COMPASS-AF. Any inconsistencies discovered in the material presented in previous deliverables have been corrected in the full COMPASS-AF presented here.
Both the AF Context Viewpoint and the Viewpoint Context Viewpoint can make use of the full Requirements Perspective if required.

The Traceability Perspective can be applied across all the other Perspectives.
In Figure 3, each of the Perspectives representing the individual Frameworks are shown. There are four points to note:

- What were presented as individual Frameworks in previous deliverables are here represented as Perspectives of the combined COMPASS-AF.
- The ‘Traceability Perspective’ can be applied across all the other Perspectives as shown by the attached comment.
- In the ‘AF & Architectures Perspective’, both the ‘AF Context Viewpoint’ and the ‘Viewpoint Context Viewpoint’ can make use of the full ‘Requirement Perspective’ if required.
- There are a number of Viewpoints that are used in more than one Perspective. These are duplicated in each Perspective were they are used and highlighted in grey.

In order to show the reuse of the Viewpoints more clearly, the COMPASS-AF is presented in Figure 4 with the packages representing each Perspective overlapped and the shared Viewpoints shown only once.

Figure 4 - Viewpoint Relationships View for the COMPASS Architectural Framework showing overlapping packages to highlight shared Viewpoints

As can be seen in Figure 4:
The ‘Requirement Context Viewpoint’ is used in three Perspectives: the ‘Requirements Perspective’, the ‘Process Perspective’ and the ‘Integration Perspective’. In fact, it can be considered to be used in the ‘AF & Architectures Perspective’ too, given that the ‘AF Context Viewpoint’ and ‘Viewpoint Context Viewpoint’ are themselves types of ‘Requirement Context Viewpoint’.

The ‘Process Content Viewpoint’ and the ‘Process Instance Viewpoint’ are used in two Perspectives: the ‘Process Perspective’ and the ‘Integration Perspective’.

The ‘Validation Interaction Viewpoint’, ‘Context Interaction Viewpoint’ and ‘Context Definition Viewpoint’ are used in two Perspectives: the ‘Requirements Perspective’ and the ‘Integration Perspective’.

What Figure 4 makes clear is the importance of understanding Needs (Goals, Capabilities and Requirements) in Context, a point emphasised by the reuse of the ‘Requirement Context Viewpoint’ across all of the Perspectives. This should not be surprising, given the prominence given to Requirements engineering and analysis in Standards such as ISO 15288 [ISO15288:2008] and associated commentaries such as [INCOSE 2011] and Competency Frameworks such as [INCOSE 2010].

The diagram in Figure 3 is reproduced in Appendix II, which also contains a glossary of the Viewpoints found on the diagram.
2.3. The COMPASS Processes and Guidelines

This section summarises the Processes and Guidelines that form the COMPASS Process Library. These Processes form a number of Process groups (groupings of related Processes, such as those to do with System Integration, Architectural Frameworks etc.), as shown in Figure 5.

The Processes in each of the Process groups shown in Figure 5 are shown in the following sub-sections, together with a summary of each Process or Guideline and references to the original deliverable documents where detailed information on each Process can be found.

2.3.1. SoS Requirements Processes

The COMPASS SoS Requirements Processes comprise two Process groups that cover both Requirements engineering (the System of Systems Requirements Engineering Process Group) and the management of Requirements, including monitoring and reacting to change (the System of Systems Requirements Management Process Group). The Process Content Views\(^2\) for both Process groups are shown in Figure 6 and Figure 7 below and are summarised in Table 4.

---

\(^2\) See [D21.1 2012] for a description of Process Content Views
Figure 6 - Process Content View for System of Systems Requirements Engineering Process Group Processes

PCV (Package) System of Systems Requirements Engineering Process Group [SoS Requirements Engineering Processes]

System of Systems Requirements Engineering Process Group

SoS Requirements Development
- Source
- Context definition view
- Requirement model
- Context interaction view
- Validation interaction view
- Review record
- activity: identify SoS stakeholder contexts
- activity: identify SoS constituent system contexts
- activity: select constituent systems
- activity: invoke 'context' process for SoS
- activity: review
- activity: identify interactions between SoS and CS
- activity: baseline
- activity: identify source elements
- activity: invoke 'requirements elicitation' process

Context Process
- Source element view
- Requirement description view
- Requirement context view
- Validation view
- Review record
- activity: select context
- activity: define context
- activity: analyse context
- activity: resolve problems
- activity: review context
- activity: define validation
- activity: review validation
- activity: baseline

Verification and Validation Definition Process
- Context definition view
- Validation view
- Test coverage view
- Review record
- activity: select context
- activity: select use case
- activity: define level of rigour
- activity: define semi-formal scenarios
- activity: define formal scenarios
- activity: review
- activity: trace to model
- activity: review coverage
- activity: baseline

Requirements Elicitation Process
- Source element view
- Requirement description view
- Review record
- activity: identify needs
- activity: elicit requirements
- activity: review
- activity: baseline
Figure 7 - Process Content View for System of Systems Requirements Management Process Group

Processes

Table 4 summarises the Processes shown in Figure 6 and Figure 7. For full details of each Process, refer to the original deliverable document. This table is repeated in Appendix III for ease of reference.

<table>
<thead>
<tr>
<th>Process</th>
<th>Description (Summary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoS Requirements Processes</td>
<td></td>
</tr>
<tr>
<td><strong>System of Systems Requirements Engineering Process Group</strong></td>
<td></td>
</tr>
<tr>
<td>SoS Requirements Development Process</td>
<td>The main aim of the SoS Requirements Development Process is to perform most of the Requirements engineering at the SoS level. This involves defining the Contexts at SoS and Constituent System (CS) level and identifying the relationships and interactions between them.</td>
</tr>
</tbody>
</table>
### Process Description (Summary)

#### SoS Requirements Processes

- **This Process uses the Requirements Elicitation Process, the Context Process (at both SoS and CS levels), the Traceability Process and the Verification and Validation Definition Process (via the Context Process).**

- **Requirements Elicitation Process**
  - The main aim of the Requirements Elicitation Process is to take the Source Element Views and, from them, elicit the parts that are relevant and create the Requirement Description View.

- **Context Process**
  - The main aim of the Context Process is to define a Context based on the Context Definition View. This is a generic Process that may be invoked from the SoS Requirements Development Process and may be applied at both the SoS and the CS level.

- **Verification and Validation Definition Process**
  - The main aim of the Verification and Validation Definition Process is to define a number of Scenarios for each Use Case in a specific Context. These Scenarios may be either semi-formal (diagram-based) or formal (mathematical-based) and form the basis of the testing of the SoS. These Scenarios are defined for both verification (it works) and validation (it does what it is supposed to do) for the Use Cases.

---

#### System of Systems Requirements Management Process Group

<table>
<thead>
<tr>
<th>Process</th>
<th>Description (Summary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traceability Process</td>
<td>The overall aim of the Traceability Process is to enable traceability to be established between any elements in the model. This may be used for the Requirements model but may also trace to any elements in the wider SoS model or its CS models.</td>
</tr>
<tr>
<td>CS Process Analysis Process</td>
<td>The overall aim of the CS Process Analysis Process is to understand the Requirement management Process of the Constituent Systems (CSs) that make up the SoS. It is important to monitor the Requirements of the CSs so that any changes can be identified and evaluated. In order to do this there needs to be an understanding of the Requirement management Process of each of the CSs. This will be achieved by modelling each Requirement management Process and then mapping to the SoS Requirement management Process. Once this understanding has been achieved and mapped to the SoS Requirement management Process, then a number of control points can be set up that allow Requirements changes to be identified periodically. In the event that the Requirement management Processes of the SoS and its CSs are not compatible, then</td>
</tr>
</tbody>
</table>
Table 4 - Summary of the COMPASS SoS Requirements Processes

<table>
<thead>
<tr>
<th>Process</th>
<th>Description (Summary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement Control Process</td>
<td>The overall aim of the Requirement Control Process is:</td>
</tr>
<tr>
<td></td>
<td>• To ensure that all information contained in the Requirement model is communicated to the relevant Stakeholder Roles.</td>
</tr>
<tr>
<td></td>
<td>• To ensure that the Requirements model is reviewed and that a consensus is achieved between the relevant Stakeholder Roles.</td>
</tr>
<tr>
<td></td>
<td>• To obtain commitment from the Stakeholder Roles that the consensus is the most appropriate way forward and to allocate suitable resources to ensure that the Requirements are satisfied.</td>
</tr>
<tr>
<td>Requirements Change Process</td>
<td>The main aim of the Requirements Change Process is to identify any changes to Requirements, assess the impact and take appropriate actions. This process may be applied at both the SoS and the CSs level and can invoke another instance of itself.</td>
</tr>
<tr>
<td>Requirements Monitor Process</td>
<td>The aim of the Requirements Monitor Process is:</td>
</tr>
<tr>
<td></td>
<td>• To allow Requirements from the CSs that make up the SoS to be monitored for change via Control Points.</td>
</tr>
<tr>
<td></td>
<td>• To allow Requirements from the SoS to be monitored.</td>
</tr>
<tr>
<td></td>
<td>Should any change occur in either the SoS or any of its CSs, then the Requirements Change Process will be invoked.</td>
</tr>
</tbody>
</table>

2.3.2. Architectural Framework Processes

The COMPASS Architectural Framework Processes comprise a single Process group that covers the definition and construction of an Architectural Framework, the Architectural Framework Group. The Process Content View is shown in Figure 8 and is summarised in Table 5.
Table 5 below summarises the Process shown in Figure 8. For full details of each Process, refer to the original deliverable document. This table is repeated in Appendix III for ease of reference.

<table>
<thead>
<tr>
<th>Process</th>
<th>Description (Summary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF Definition Process</td>
<td>The aim of this Process is to understand the underlying need for the Architectural Framework (AF). The Context of the AF that is under development is defined via the Context Process, so that the Needs that it is to address are clearly understood and any source Standards that the AF under development may need to comply with are identified. Based on the AF Context View produced, the Ontology for the AF under development is now defined by invoking the Ontology Definition Process. This returns the Ontology Definition View.</td>
</tr>
<tr>
<td>Process</td>
<td>Description (Summary)</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Architectural Framework Definition Processes</strong></td>
<td>Based on the Ontology and the AF Context, a number of Viewpoints are now identified along with the relationships between them. This results in the Viewpoint Relationship View.</td>
</tr>
<tr>
<td>Context Process</td>
<td>The main aim of this Process is to create a Context that can be used to create either an AF Context View or a Viewpoint Context View. The Processes produce a Context that identifies Use Cases based on the Needs, identifies the System boundary, identifies Stakeholder Roles that are external to the System boundary, identifies relationships between the Use Cases and the external Stakeholder Roles and identifies relationships between Use Cases. The Context is then analysed to ensure that all Use Cases, Stakeholder Roles and their relationships are understood. Any problems that may have been identified during the Context analysis are addressed.</td>
</tr>
<tr>
<td>Ontology Definition Process</td>
<td>The main aim of this Process is to identify and define the main concepts and terms used for the AF in the form of an Ontology. The concepts that are relevant to the AF under development are identified and defined, with their provenance established against source material. Such concepts now become Ontology Elements. These Ontology Elements are then related one to another to become part of the Ontology, which is realised as the Ontology Definition View.</td>
</tr>
<tr>
<td>Viewpoint Definition Process</td>
<td>The main aim of this Process is to identify and define the key Viewpoints and to classify them into Perspectives. Each Viewpoint from the Viewpoint Relationship View (produced by the AF Definition Process) is considered in turn. For the Viewpoint, its Context is created by invoking the Context Process. This produces the Viewpoint Context View for the selected Viewpoint. Based on the Ontology Definition View, a subset of the Ontology is now identified that is judged to be relevant for the selected Viewpoint. This is used to define the Viewpoint Definition View for the selected Viewpoint. Any relationships that exist between the Viewpoints are</td>
</tr>
</tbody>
</table>
D21.6 – Final Report on Guidelines for SoS Engineering (Public)

<table>
<thead>
<tr>
<th>Process</th>
<th>Description (Summary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architectural Framework Definition Processes</td>
<td>identified and added to the Viewpoint Relationship View. This is used as an input for defining Rules for ensuring consistency of the Viewpoints defined, captured as the Rules Definition View.</td>
</tr>
</tbody>
</table>

Table 5 - Summary of the COMPASS Architectural Framework Processes

2.3.3. Architecture Guidelines


The table below summarises these Guidelines. For full details of each of the Guidelines, refer to the original deliverable document. This table is repeated in Appendix III for ease of reference.

<table>
<thead>
<tr>
<th>Guideline</th>
<th>Description (Summary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine SoS Requirements and Stakeholders</td>
<td>This activity involves translation of business needs for the SoS into verifiable Requirements, along with identification of a preliminary set of Stakeholders for each Requirement. This is going to be refined by tracing Requirements further to CSs that are also associated with Stakeholders. At this time we do not take into account the CSs. See [D21.5a 2013] Section 8.</td>
</tr>
<tr>
<td>Determine CSs and their Stakeholders</td>
<td>The understanding of the SoS is still incomplete. Understanding the CSs, their relationships and Stakeholders is one of the major objectives of this activity. This has far-reaching consequences on feasible sets of Requirements, degraded operation and on SoS integration. Two principle kinds of Stakeholders must be distinguished: technological and business. Many of the constraints seen in CSs are not technological in nature but related to business goals and interests. We collect the terms like societal, legal, i.e., “non-technological” in the one term: business. The characterisations of SoSs and CSs tend to be driven by business constraints. It is advisable to consider business interests of the Stakeholders first and defer technological considerations to a later point. Whereas for</td>
</tr>
<tr>
<td>Guideline</td>
<td>Description (Summary)</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------</td>
</tr>
<tr>
<td><strong>Architecture Guidelines</strong></td>
<td>technological problems technological solutions are often not far away, business constraints may disallow such solutions. This activity needs to be carried out continually while the Architecture of the SoS is developed and made gradually more specific. See [D21.5a 2013] Section 9.</td>
</tr>
<tr>
<td>Outline SoS Architecture</td>
<td>The high complexity of a typical SoS makes it difficult to reason about its Architecture directly. In order to approach a complete model of the SoS we first make some simplifying assumptions: We assume that all features of all CSs are immediately accessible and all interfaces and protocols are compatible. Essentially, we assume that we are analysing an ordinary system. The SoS constraints are ignored. Using an outline SoS Architecture we argue that the Architecture satisfies the Requirements. This provides assurance that the CSs of our initial list are, in principle, sufficient to realise the SoS. See [D21.5a 2013] Section 10.</td>
</tr>
<tr>
<td>Classify CSs and their components</td>
<td>The outline SoS Architecture shows what is required from each CS in terms of functionality, interfaces and protocols. For each CS it is necessary to determine how it could support all of those, provided the actual functionality, interfaces and protocols were compatible. We identify components with different groups of functionality, interfaces and protocols and classify them. We distinguish an SoS into three decomposition layers: SoS, CS, component. The component layer is associated with concrete functionality. The reason for considering components as subdivisions of CSs is that usually each CS has functionality with different degrees of openness. For instance, service discovery may be freely accessible whereas requesting diagnoses of a patient record stored on some device may not be. A CS and its different components can be classified into the different categories forbidden, closed/legacy, extensible and open. Each category describes the degree of openness towards the integration into a given SoS. See [D21.5a 2013] Section 12.</td>
</tr>
<tr>
<td>Guideline</td>
<td>Description (Summary)</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Improve outline SoS Architecture</td>
<td>The information on what the CSs really provide is added to the outline SoS Architecture. The CS classification determines how this can be done. Depending on how components and CSs are classified: the components can be modified or not; additional components or CSs may have to be included; access to a component may be allowed or disallowed.</td>
</tr>
<tr>
<td></td>
<td>During this activity, communication paradigms and architectural styles are established.</td>
</tr>
<tr>
<td></td>
<td>See [D21.5a 2013] Section 13.</td>
</tr>
<tr>
<td>Vary configurations and classify them as SoS</td>
<td>The purpose of this activity is to determine characteristics of the SoS in order to develop integration and verification strategies. An SoS may belong to several categories depending on the different configurations considered. The result of this step depends on the configurations actually analysed. Which configurations should be considered depends on which configurations are possible and what set of requirements they satisfy. Some configurations may be chosen for verification concerns, even if it would never actually occur. The result of this task includes an analysis of configurations versus Requirements providing insight into full performance with respect to Requirements, or degraded performance.</td>
</tr>
<tr>
<td></td>
<td>See [D21.5a 2013] Section 14.</td>
</tr>
<tr>
<td>Outline detailed SoS Architecture</td>
<td>A detailed Architecture of the SoS is produced. The Architecture must support all configurations. The different configurations should be indicated and their SoS classifications stated.</td>
</tr>
<tr>
<td></td>
<td>See [D21.5a 2013] Section 15.</td>
</tr>
<tr>
<td>Determine integration test strategy for the SoS</td>
<td>Different configurations have their proper test strategy assigned to them. This permits validation against the requirements even if the full system will not satisfy all of them. The classification of the different configurations indicates feasible strategies for testing. The aim of this activity is to contribute towards test planning focusing on SoS specific issues.</td>
</tr>
<tr>
<td></td>
<td>See [D21.5a 2013] Section 16.</td>
</tr>
</tbody>
</table>
2.3.4. Refinement Process

Although deliverable [D22.5 2013] discusses the issues of refinement, presenting a Refinement Framework, it lacked any Processes to use with the Refinement Framework. This Section addresses this omission, and contains a Refinement Point Process that is intended to be used with the Refinement Framework in [D22.5 2013].

Requirement Context View

The diagram in Figure 9 shows the Requirement Context View for the Refinement process. The highest-level Use Case shows that there is a need to ‘Provide guidelines for SoS integration’ that has two main constraints:

- ‘Meet best practice’ that requires that any approach is compliant with relevant Standards
- ‘Be compliant with other processes’ that requires that any Processes that are defined are compliant with the other Processes that are defined as part of the COMPASS Project Process Model.

The main inclusion is to ‘Define Refinement Points’ at two levels:

- ‘...across levels’ where refinement occurs within a single level of abstraction of the model.
- ‘...between levels of abstraction’ where refinement occurs between different levels of abstraction in the model.

The ‘Define Refinement Points’ Use Case has two inclusions:
• ‘Identify Refinement Points’ where the individual Refinement points are identified in the System Model
• ‘Specify Refinement points’ where each of the identified Refinement Points is specified in terms of its Rules.

The Refinement Process that is defined in this section satisfies each of these Use Cases.

**Stakeholder view**

The diagram in Figure 10 shows that there are three Stakeholder Roles that are utilised by the Refinement Process, which are:

- ‘System Modeller’ that performs the modelling on the System and System of Systems
- ‘Reviewer’ that carries out the reviews on the Process Artefacts
- ‘Configuration Manager’ that manages the Model in terms of the version control and configuration management.

These three Stakeholder Roles are specialisations of the ‘Development’ Stakeholder Role which, in turn, is a specialisation of the ‘Supplier’ Stakeholder Role.
The diagram in Figure 11 shows that there is a single refinement-related Process in the form of the ‘Refinement Point Process’.

The main aim of the Process is to identify and define the Refinement Points for a specific System Model.

The Refinement Point Process has the following Activities:

- ‘identify View’, where a View from the System Model that contains a Refinement Point is identified. This Activity is performed by the System Modeller.
- ‘identify Refinement Point’, where the selected View from the System Model is taken as an input and one or more Refinement Points are identified. This Activity is performed by the System Modeller.
- ‘define Rule’, where the Rules that define the translation between two Refinement Points are defined. This Activity is performed by the System Modeller.
- ‘review’, where all the Process Artefacts are reviewed by the Reviewer.
- ‘baseline’, where all the Process Artefacts are versioned and managed.

The Refinement Point Process has the following Artefacts:

- ‘System model’, which represents the model that is used as a basis for the refinement.
- ‘RPIV’, the Refinement Point Identification Viewpoint, where the Refinement Points are identified and related to each other. Each Refinement requires two Refinement Points and one or more Rule.
• **RPDV**, the Refinement Point Definition View, where the Rules that describe a specific Refinement are defined. Each Refinement requires two Refinement Points and one or more Rule

All Artefacts are defined according to the Refinement Pattern.

**Process Behaviour View**

The diagram in Figure 12 shows the Process Behaviour View for the Refinement Process.

The Process begins with the System Modeller who, based on the System Model, identifies the Views that contain the Refinement Points (‘identify View’). These Views are then used to identify a pair of Refinement Points (‘identify Refinement Point’) and then this is repeated until all of the Refinement Points have been identified.

Once all of the Refinement Points have been identified, the System Modeller then defines the Rules that apply to each Refinement Point pair (‘define Rule’), which is repeated until Rules have been defined for all Refinement Point pairs. The Reviewer then reviews all of the Process Artefacts to ensure that they are correct (‘review’) and, if the review is passed, then the Configuration Manager baselines all Process Artefacts (‘baseline’). In the event that the review is failed, the control of the Process flow goes back to ‘define Rule’ or ‘identify View’, depending on whether the review failed because of the Views or the Rules.
Information View

The diagram in Figure 13 shows the Information View for the Refinement Process.

The diagram shows that the ‘Refinement Point Identification View’ is identified from the ‘Model’, and that the ‘Refinement Point Definition View’ defines rules for the ‘Refinement Point Identification view’.

### 2.3.5. System Integration Processes

The COMPASS System Integration Processes comprise two Process groups that cover both the integration itself (the Integration Technical Group) and the strategy and planning of the integration (the Integration Management Group). The Process Content Views for both Process groups are shown in Figure 14 and Figure 15 below and are summarised in Table 7.
The table below summarises the Process shown in Figure 14 and Figure 15. For full details of each Process, refer to the original deliverable document. This table is repeated in Appendix III for ease of reference.
<table>
<thead>
<tr>
<th>Process</th>
<th>Description (Summary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constituent System Identification Process</td>
<td>The Constituent System Identification Process identifies the SoS and its component CSs. It combines the Requirement Context Views for the CSs into the Context Interaction View for the SoS and uses this to help identify the Interfaces between CSs, producing Interface Identification Views for these Interfaces. All Artefacts produced are reviewed and are placed under formal configuration control.</td>
</tr>
<tr>
<td>Integration Process</td>
<td>The Integration Process takes each Interface identified in the Constituent System Identification Process and expands on the definition of these Interfaces by defining the Interface connectivity, where the connections between the System Elements of the SoS (i.e. between the CSs) are defined by creating a number of Interface Connectivity Views. The structure of each Interface is defined by creating the Interface Definition Views and the interactions between the System Elements (the CSs) and across the selected Interface are defined by creating the Interface Behaviour Views. Any Protocols, if required, are defined for each Interface by creating the Protocol Definition Views. All Artefacts produced are reviewed and are placed under formal configuration control.</td>
</tr>
<tr>
<td>Validation Process</td>
<td>In the Validation Process, the Validation Interaction Views for the SoS are produced or collected together. These form the main validation Scenarios for the SoS. The Interface Behaviour Views for the Interfaces (as produced by the Integration Process) are collected together and used to perform the actual validation by comparing the set of Interface Behaviour Views for the Interfaces to the Validation Interaction Views for the SoS. All Artefacts produced are reviewed and are placed under formal configuration control.</td>
</tr>
<tr>
<td>Integration Strategy</td>
<td>The Integration Strategy Process defines the Needs of the integration by creating a Requirement Context View from the point of view of the Integration. 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</td>
</tr>
</tbody>
</table>
### System Integration Processes

The Processes that will be carried out in order to perform integration are identified by identifying an appropriate Process Content View. In the case of the COMPASS Project, this will be the Systems Integration Processes defined in [D21.4 2013]. These Processes identified are validated against the original Needs through a number of Process Instance views.

Based on the above, an ‘Integration Plan’ for the integration activities is created.

All Artefacts produced are reviewed and are placed under formal configuration control.

#### 2.3.6. Competency Processes

The COMPASS Competency Processes are described in Section 4.2 below.
3. Application of the COMPASS Approach

This section provides some examples illustrating the order in which the various Process groups (covering Requirements, Architectural Frameworks, Architectures and Integration) of the COMPASS Process library would be executed during the development and maintenance of an SoS. In order to set this discussion in context, it begins with a discussion of the concepts of Life Cycle and Life Cycle Model.

3.1. Life Cycles and Life Cycle Models

Before considering the way that the COMPASS approach can be used for the development and maintenance of an SoS, it is first useful to discuss the concepts of Life Cycle and Life Cycle Model.

A Life Cycle describes the possible evolution of a System (and here we are using System to apply equally to a SoS or CS) over time, through the Stages that are available as part of the Life Cycle. For example, Figure 16 illustrates the Stages that make up the Life Cycle defined by ISO15288 (see ISO15288:2008).

ISO15288 defines six Life Cycle Stages, which have the following objectives:

- Conception is executed to assess new business opportunities and to develop primary System Requirements and a feasible design solution.
• Development is executed to develop the System of interest that meets acquirer Requirements and can be produced, tested, evaluated, supported and retired.
• Production is executed to produce and manufacture the product, to test the product, and to produce related supporting and enabling Systems as needed.
• Utilisation is executed to operate the product, to deliver services within intended environments and to ensure continued operational effectiveness.
• Support is executed to provide logistics, maintenance and support services that enable continued System of interest operation and a sustainable service.
• Retirement is executed to provide for the removal of a System of interest and related operational and support services, and to operate and support the retirement System itself.

A single System may have any number of Life Cycles associated with it, depending on the context of the system. For example:

• A product Life Cycle
• A project Life Cycle
• An acquisition Life Cycle
• An operational Life Cycle, etc.

Each of these Life Cycles is, however, structural. It represents the Stages that the System can be in at any point in time. The time-based behavioural instance of a Life Cycle is a Life Cycle Model which shows the sequencing of Stages through time. A conceptual Life Cycle Model, based on the Life Cycle Stages of ISO15288 is shown in Figure 17.

Figure 17 shows a conceptual high-level Life Cycle Model based on the ISO15288 Life Cycle. The Project starts with the ‘Conception’ Stage, followed by

Given that COMPASS is aimed at the development of a SoS, the COMPASS approach will typically cover the ‘Conception’, ‘Development’ and ‘Production’ Stages as shown by the frame around those Stages. However, if the SoS is in existence and is being modified then the COMPASS approach will also cover the ‘Utilisation’ and ‘Support’ Stages as shown in Figure 18 since these Stages cover change and evolution of the SoS.

While the COMPASS approach is not shown as applying to the ‘Retirement’ Stage of a SoS Project, that of course depends on the nature of the Project. Given that the Retirement Stage “is executed to provide for the removal of a System of interest and related operational and support services, and to operate and support the retirement System itself”, it is possible that the retirement System may itself be a SoS with its own Life Cycle and Life Cycle Model and hence the COMPASS approach will be applicable to the ‘Retirement’ Stage.

When defining a Life Cycle Model for a System, each Stage will have a number of Process Execution Groups that are executed within it. A Process Execution Group is simply a collection of Processes that are executed to achieve a particular purpose (see the full COMPASS SoSE Ontology in Figure 2, Section 2.1). Whereas the Stages within a Life Cycle Model are constrained by the particular Life Cycle adopted, the Process Execution Groups that are executed within a Stage in a Life Cycle Model vary from Project to Project. The Processes that make up each Process Execution Group will come from the engineering Process library that is being used on the Project. Thus, when adopting the COMPASS approach, the processes will come from the COMPASS Process library. If adopting an approach wholly based on ISO15288, then the Processes available will be those defined by ISO15288.
3.2. Application to SoS Development

This section shows some conceptual breakdowns of Life Cycle Stages for the development of an SoS, covering the Conception, Development and Production Stages. The Life Cycle Models presented use the Process groups of the COMPASS Process library to represent the Process Execution Groups that would normally be found in the Stages of a Life Cycle Model, in order to illustrate the likely sequence of use of the Processes that form part of the COMPASS approach. For a real Project this would not be the case, with the Process Execution Groups defined specifically for that Project and not necessarily bearing any relation to groupings of Processes in an organisation's Process library.

![Figure 19 - COMPASS Processes executed during the Conception Stage for the development of a new System](image)

Figure 19 shows the breakdown of the Conception Stage for the development of a new System. In this Stage, the Processes of the ‘Architectural Framework Process Group’ are first exercised to develop an Architectural Framework for the System. This is followed by use of the Processes from the ‘System of Systems Requirement Process Group’ in order to develop the initial Requirements for the System (the SoS). This may require additional work on the Architectural Framework, followed by additional work on the Requirements before developing an initial Architecture through the use of the Guidelines in the ‘Architecture Guidelines Group’ and using the Architectural Framework developed. The production of the initial Architecture may require further work on the Architectural Framework and then a final pass through the Requirements before moving on to the next Stage (Development). As can be seen from Figure 19, during the Conception Stage of a new System, most of the emphasis is on the production of an Architectural Framework and on understanding the Requirements for the System (the SoS).

If the development work is being carried out on an existing System, then it is likely that an Architectural Framework already exists. In this case, the Stage will be simpler as shown in Figure 20.
Figure 20 – COMPASS Processes executed during the Conception Stage for the development of an existing System

Figure 20 shows the breakdown of the Conception Stage for the development of an existing System. In this Stage, the Processes of the 'System of Systems Requirement Process Group' are executed in order to develop the initial Requirements for the new version of the System (the SoS). After the Requirements are developed, an initial Architecture is produced through the use of the Guidelines in the 'Architecture Guidelines Group'. A final pass through the Requirements is made before moving on to the next Stage (Development). As can be seen from Figure 20, during the Conception Stage of an existing System, most of the emphasis is on understanding the Requirements for the System (the SoS).

Following the Conception Stage is the Development Stage, which is illustrated in Figure 21.

Figure 21 – COMPASS Processes executed during the Development Stage

As can be seen in Figure 21, most of the emphasis during the Development Stage is on the production of the Architecture through the execution of the Guidelines in the 'Architecture Guidelines Group'. This may require some additional work on the Requirements, through the Processes in the 'System of Systems Requirement Process Group', before returning to the Architecture and then starting initial integration of the CS by executing the Processes in the 'Systems Integration Process Group'. Initial integration work may require additional work to the Architecture as shown by a final execution of the 'Architecture Guidelines Group'.
Following the Development Stage is the Production Stage, which is illustrated in Figure 22.

![Diagram showing COMPASS Processes executed during the Production Stage](image)

The COMPASS approach is not specifically aimed at the design of the CS that make up an SoS, but rather at the definition of the Architecture for the SoS and the integration of the CS into the SoS. For this reason the Production Stage for an SoS (from the point of view of the SoS as the Project under development) places the emphasis on integration. The Processes in the ‘Systems Integration Process Group’ will be executed first. As a result of the integration, changes may be required to be made to the Architecture. These changes are made and then further integration carried out. Note that any such changes to the Architecture may require rework to the CSs before further integration can be performed. As with all these Stages, the iteration between Process groups would be carried out as many times as necessary. The examples given here are only conceptual.

### 3.3. Application to SoS Maintenance

In Section 3.2, the breakdown of Life Cycle Stages for the development of an SoS was given. In this section the same is done for the maintenance of an SoS. Figure 23 shows an overall Life Cycle Model for the System.
The Life Cycle Model starts with the ‘Utilisation’ and ‘Support’ Stages running in parallel, as shown in Figure 23. The breakdown of the Support Stage is shown in Figure 24.

During the Support Stage, the Processes of the ‘System of Systems Requirements Management Process Group’ are executed. As a result of executing these Processes, changes to Requirements for the SoS are detected. These changes are then analysed and an understanding of the new Requirements established through execution of the Processes in the ‘System of Systems Requirements Engineering Process Group’. Iteration between these two Process groups takes place until the Requirements changes are fully understood at which point the Stage ends and, as can be seen from Figure 24, the Concept Stage is returned to. This can also be seen on Figure 23 where, following the end of the first Support Stage the Life Cycle Model is essentially one of development of the changes identified during the initial Support Stage and hence the examples in Section 3.3.
serve to illustrate the aspects of the COMPASS approach that are relevant. After the changes have been made, the System re-enters the Utilisation and Support Stages before finally, in this conceptual example, entering the ‘Retirement’ Stage.

### 3.4. Application to Retirement of CSs

This section provides an example of the Life Cycle Model for a Project where a Constituent System is retired during the Support Stage and then shows the COMPASS Processes that are executed during the Support Stage.

The diagram in Figure 25 shows the Overall Life Cycle Model of a Project that starts off in the parallel Support and Utilisation Stages. In this example one of the Constituent Systems that comprise the System of Systems is retired during the Support Stage. When the Constituent System is retired it is necessary to revisit the Concept Stage in order to ensure that the now-depleted System of Systems can still satisfy its original requirements. In this scenario it is judged that the System of Systems can satisfy the original requirements providing that some reconfiguration of the Constituent Systems takes place, which is reflected in the execution of the ‘Development’ and ‘Production’ Stages.
The diagram in Figure 26 shows the COMPASS Processes that are executed during the Support Stage of the Project. The Processes in the ‘System of Systems Requirements Management Process Group’ and the ‘System of Systems Requirements Engineering Process Group’ are being executed when the retirement of a Constituent System is detected. The Support Stage then initiates the Conception Stage in order to ensure that the System of Systems still satisfies its original requirements.

3.5. Conclusions

This section has shown a number of scenarios that are possible to be executed during a System of Systems Project. Three sets of scenarios were considered:

- Application to System of Systems development
- Application to System of Systems maintenance
- Application to System of Systems retirement

In each application scenarios were shown for both the overall Life Cycle Model and one of the Stages in that Life Cycle Model.

This set of scenarios are not intended to be exhaustive but is intended to show a sample of some of the ways in which the COMPASS Processes may be executed on a typical Project.
4. Competence and SoSE

There are a large number of different Stakeholder Roles involved in SoSE, all requiring different mixes of skills and abilities in order to carry out the activities associated with a particular Stakeholder Role. When considering individuals for a specific Stakeholder Role, it is essential that they have the right knowledge, skills and attitude required to perform the activities required for the Stakeholder Role.

One way of ensuring that an individual has the correct knowledge, skills and attitude is to assess that individual against a defined Competency Framework. Many such Competency Frameworks exist. Three that are particularly relevant to those engaged in SoSE are the INCOSE Systems Engineering Competencies Framework [INCOSE 2010], the Skills Framework for the Information Age (SFIA) [SFIA 2013] and the Association for Project Management (APM) Body of Knowledge [APM 2013]. Of these, the INCOSE Framework is the main Competency Framework that should be used, as it contains Competencies that relate directly to Systems engineering. However, the other Frameworks provide additional Competencies relating to software engineering & information systems and to project management respectively. Such Competencies are either not covered by the INCOSE Framework or are covered at only a high level. For a Person engaged in the management aspects of an SoS, Competencies from the APM Body of Knowledge might be more relevant, for example. In practice, an Organisation should develop a bespoke Competency Framework based on a number of source Frameworks. For information on a model-based approach to developing such Competency Frameworks, see [Holt & Perry 2011].

This section discusses the concepts surrounding Competence in more detail through the definition of the COMPASS Competency Framework (CCF) and then presents a number of Competency Scope for the various Stakeholder Roles involved in the various COMPASS Processes defined to date. In this section, the following definitions are adopted:

- Competence reflects the total ability of a Person.
- Competency reflects a single skill. A Person’s Competence is the sum of their Competencies.

While each of the Competency Frameworks discussed above are relevant to SE and SoSE, the issue is one of completeness. No single Competency Framework can every hope to cover all of the needed Competencies. This is a common mistake made within SE Organisations: they adopt a single Competency Framework with no consideration as to whether it is entirely suitable. A better approach is to define a bespoke Competency Framework, tailored to the Organisation, that makes use of a number of existing Competency Frameworks as inputs. That is what has been done with the CCF. The processes defined below can be used by an Organisation in the definition of their own Competency Framework.
4.1. The COMPASS Competency Framework

This Section defines the COMPASS Competency Framework (CCF) using the Viewpoints defined in the COMPASS Architectural Framework Framework (CAFF). The CAFF can be found in [D21.5 2014]. Although discussed in Section 4.1.1, it is worth stressing here that the purpose of the CCF is twofold: it can be used both to help define a bespoke Competency Framework for an Organisation and also be used when carrying out Competency assessments.

The Viewpoints defined in the CCF are the minimum needed and cover the core concepts as discussed in Section 4.1.2. Other concepts, such as acceptable evidence types and their length of validity are not discussed. See [Holt & Perry 2011] for a discussion of such concepts.

4.1.1. AF Context View

The Needs that the COMPASS Competency Framework address are shown in Figure 27.

The main Need addressed by the COMPASS Competency Framework is to ‘Provide organisational support for competency’. It does this in two ways, by providing support for the definition of Competency Frameworks (‘Support definition of bespoke competency frameworks’) and by providing support to those carrying out Competency assessments (‘Support carrying out competency assessments’).

Each of these two main use cases are broken down further. ‘Support definition of bespoke competency frameworks’ includes:
• ‘Support definition of bespoke competencies’ – that is, the COMPASS Competency Framework must support the definition of any bespoke Competencies that are required by a Competency Framework, rather than forcing the ‘Competency Framework Definer’ to use existing Competencies from existing Frameworks.

• ‘Support definition of bespoke competency levels’ – that is, the COMPASS Competency Framework must support the definition of any bespoke Levels that are required by a Competency Framework, rather than forcing the ‘Competency Framework Definer’ to use existing Levels from existing Frameworks.

It is constrained by the Need to ‘Allow use of existing frameworks’. Thus, while supporting the definition of bespoke Competencies and Levels, the ‘Competency Framework Definer’ must be able to use the concepts (Competencies, Levels, Competency Areas etc.) from any existing Competency Frameworks if desired and in a way that enables the use of such concepts from a number of different Competency Frameworks.

‘Support carrying out competency assessments’ includes:

• ‘Support definition of required competencies’ – that is, the COMPASS Competency Framework must support the definition of Competency Scopes as required by ‘Human Resources’ and the ‘Assessor’ before a Competency assessment can be carried out.

• ‘Support capture of held competencies’ – that is, the COMPASS Competency Framework must support the production of Competency Profiles for an ‘Assessee’ as a result of a Competency assessment that has been carried out.

The concepts and Viewpoints for the COMPASS Competency Framework are given in the following Sections.

4.1.2. Ontology Definition View

The various concepts relating to Competence are shown below in Figure 28.
Figure 28 - Ontology Definition View focussing on Competence

The diagram shows that a ‘Person’ exhibits ‘Competence’ which is made up of one or more ‘Competency’, each of which is held at a particular ‘Level’ and which has a one or more ‘Indicator’ associated with it that describes the knowledge, skill or attitude required to meet the ‘Competency’. It is each ‘Indicator’ that is assessed as part of Competency assessment.

The link between a ‘Person’ and the ‘Competence’ that they exhibit is through the one or more ‘Stakeholder Role’ that they hold. Each such ‘Stakeholder Role’ requires a number of ‘Competency’ as defined on a ‘Competency Scope’. Each ‘Competency’ is required to be held at a ‘Level’. The set of ‘Competency Scope’ that are relevant to a ‘Person’ describes the desired ‘Competence’ of that ‘Person’.

The actual, measured ‘Competence’ of a ‘Person’ is described by the set of ‘Competency Profile’ held by that ‘Person’. Each such ‘Competency Profile’ contains the results of a Competency assessment performed against a particular ‘Competency Scope’.

In order to aid in the organisation of Competencies into a Competency Framework, most such Frameworks group related Competencies, such as those related to Requirements engineering or to Architectures, into a number of ‘Competency Area’.

Most such Competency Frameworks define a number of Levels at which a Competency can be held. For example, [INCOSE 2010] defines four levels: awareness, supervised practitioner, practitioner and expert. Four levels are also defined for COMPASS, but with different names and a slightly different emphasis. They are:
• **Level 1 – Awareness Level** - The aim of this Level is for the assessee to demonstrate that they possess the ability to *speak knowledgeably* about the Competency. The main aim is for the assessee to demonstrate that they understand each Indicator fully, and back this up with examples - either theoretical or real-life.

• **Level 2 – Support Level** - The aim of this Level is for the assessee to demonstrate that they can reflect the ability to *implement* the concepts that were discussed at Level 1 for this Competency.

• **Level 3 – Lead Level** - The aim of this Level is for the assessee to demonstrate that they can reflect the ability to be able to *lead the activity* that was described at Level 1 and implemented at Level 2.

• **Level 4 – Expert Level** - The aim of this Level is for the assessee to demonstrate that they can reflect the ability to be a *true, recognised expert in the field* that is described by this Competency.

These Levels map on to the corresponding INCOSE Level but better demonstrate the way that the Levels build on each other since each Level explicitly expands on the Levels beneath it. Thus for any Competency at Level 3, a Person must show that they can describe (Level 1), implement (Level 2) and lead (Level 3) activities described by the Competency. This incremental extension is less clear in the INCOSE Framework [INCOSE 2010], where often a Level introduces new Indicators not found in the lower Levels and often not directly related to those earlier Indicators. For a discussion of Competency Frameworks and assessments that takes a model-based approach to the subject, see [Holt & Perry 2011].

### 4.1.3. Viewpoint Relationships View

The CCF defines four Viewpoints, as shown in Figure 29. Note that on this figure, the CCF is referred to as the Competency Perspective. This is to emphasise that the CCF is a Perspective of the wider COMPASS-AF.
The Viewpoints that make up the CCF are:

- Competency Framework Definition Viewpoint that defines a competency framework in terms of the Competency Areas it contains, the Competencies in each Competency Area and the Indicators that make up each Competency.
- Competency Level Definition Viewpoint that defines the Levels that are used in the competency framework.
- Competency Scope Definition Viewpoint that defines a Competency Scope that is required for a particular Stakeholder Role. It is based on the Competencies defined in the Competency Framework Definition Viewpoint.
- Competency Profile Definition Viewpoint that captures the Competency Profile for a Person, as assessed against a defined Competency Scope.

Each of these Viewpoints will be defined fully in Section 4.1.5 below.

### 4.1.4. Rules Definition View

Four rules apply to the four Viewpoints, as shown in the Rules Definition View (RDV) in Figure 30

#### Figure 30 - Rules Definition View for the Competency Framework

Note that the four rules shown in Figure 30 are the minimum that are needed. Others could be added if required.
4.1.5. Viewpoint Definitions

Each of the four Viewpoints are defined in this Section. For each Viewpoint, a Viewpoint Context View and Viewpoint Definition View is provided, according to the CAFF approach as described in [D21.2 2013]. An example View that conforms to the Viewpoint is also given for each.

Competency Framework Definition Viewpoint

Viewpoint Context View

The Needs that the Competency Framework Definition Viewpoint addresses are shown in Figure 31.

The main purpose of the Competency Framework Definition Viewpoint is to ‘Support definition of bespoke competencies’, an inclusion of the use case ‘Support definition of bespoke competency frameworks’. Any definition of bespoke Competencies must be capable of being made in such a way to ‘Allow use of existing frameworks’, the constraint on the ‘Support definition of bespoke competency frameworks’ use case. That is, when defining bespoke Competencies, it must be possible to use material, concepts etc. from existing Competency Frameworks. The Competency Framework Definition Viewpoint, through the include relationships contributes towards an Organisation’s Need to ‘Provide organisational support for competency’.
**Viewpoint Definition View**
The definition of the Competency Framework Definition Viewpoint (CFDVp) is shown in Figure 32.

![Figure 32 - Viewpoint Definition View for the Competency Framework Definition Viewpoint](image)

The Competency Framework Definition Viewpoint in Figure 32 defines a Competency Framework in terms of the Competencies it contains, which are typically classified into related Competency Areas, such as those that cover modelling or requirements or life cycles etc.

Typically, a Competency Framework will be structured by Competency Area. For each Competency within a Competency Area, its Indicators are given. Typically a description of the Competency is included.

**Example**
An example of a Competency Framework Definition View (CFDV) is shown in Figure 36.
Figure 33 - An example Competency Framework Definition View

The Competency Framework Definition View in Figure 33 defines a single Competency (‘System Concepts’) for the ‘Systems Thinking’ Competency Area. It includes a description of the Competency and the Indicators that relate to the Competency, here shown by Level.

The representation of a single Competency on a single Competency Framework Definition View is common in the definition of a Competency Framework.

**Competency Level Definition Viewpoint**

**Viewpoint Context View**

The Needs that the Competency Level Definition Viewpoint addresses are shown in Figure 34.
The main purpose of the Competency Level Definition Viewpoint is to ‘Support definition of bespoke competency levels’, an inclusion of the use case ‘Support definition of bespoke competency frameworks’. Any definition of bespoke Levels must be capable of being made in such a way to ‘Allow use of existing frameworks’, the constraint on the ‘Support definition of bespoke competency frameworks’ use case. That is, when defining bespoke Levels, it must be possible to use material, concepts etc. from existing Competency Frameworks. The Competency Level Definition Viewpoint, through the include relationships contributes towards an Organisation’s Need to ‘Provide organisational support for competency’.

**Viewpoint Definition View**
The definition of the Competency Level Definition Viewpoint (CLDVp) is shown in Figure 35.
The Competency Scope Definition Viewpoint in Figure 35 defines the allowed Levels at which a Competency may be held. Each Level will typically have a name; they often also have a number (such as Level 1) associated with them. Each Level will have a description that describes how the Level is used.

**Example**

An example of a Competency Level Definition View (CLDV) is shown in Figure 36.

---

**Figure 35 - Viewpoint Definition View for the Competency Level Definition Viewpoint**

**Figure 36 - An example Competency Level Definition View**
The Competency Level Definition View in Figure 36 has been modelled as a SysML block definition diagram, using blocks to define four Levels. The description for each Level is also shown, indicating how the Level is to be used when assessing a Competency. Note the use of dependencies between Levels to show that a given Level is dependent on the lower Level.

**Competency Scope Definition Viewpoint**

**Viewpoint Context View**

The Needs that the Competency Scope Definition Viewpoint addresses are shown in Figure 34.

The main purpose of the Competency Scope Definition Viewpoint is to ‘Support definition of required competencies, an inclusion of the use case ‘Support carrying out competency assessments’. Thus, this Viewpoint is concerned with the definition of Competency Scopes that are required when a Competency assessment is carried out.

The Competency Scope Definition Viewpoint, through the include relationships contributes towards an Organisation’s Need to ‘Provide organisational support for competency’.

**Viewpoint Definition View**

The definition of the Competency Scope Definition Viewpoint (CSDVp) is shown in Figure 38.
The Competency Scope Definition Viewpoint defines a Competency Scope (in fact, the realisation of a Competency Scope Definition Viewpoint, that is a Competency Scope Definition Viewpoint is a Competency Scope; they are the same thing).

The Competency Scope Definition Viewpoint shows the Competencies that are required by a particular Stakeholder Role along with the Level at which each Competency is required to be held. The Competencies are usually grouped by Competency Area.

**Example**

An example Competency Scope Definition View (CSDV) is shown in Figure 39 below.
Figure 39 - An example Competency Scope Definition View

The Competency Scope Definition View in Figure 39 shows a Competency Scope for the ‘Requirements Manager’ Stakeholder Role. It shows that this Stakeholder Role requires nine Competencies to be held, from ‘Systems concepts’ through to ‘Planning, monitoring and controlling’. These nine Competencies are grouped into three Competency Areas (‘Systems Thinking’, ‘Holistic Life Cycle View’ and ‘Systems Engineering Management’). The required (minimum) Level at which each Competency should be held for this Stakeholder Role is also shown. For example, the ‘Systems concepts’ Competency is required to be held at ‘Level 3 – lead’.

Many more examples can be found in Section 4.3.

**Competency Profile Definition Viewpoint**

**Viewpoint Context View**

The Needs that the Competency Profile Definition Viewpoint addresses are shown in Figure 34.
The main purpose of the Competency Profile Definition Viewpoint is to ‘Support capture of held competencies, an inclusion of the use case ‘Support carrying out competency assessments’. Thus, this Viewpoint is concerned with the recording of held Competencies as a Competency Profile for an ‘Assessee’, produced after a Competency assessment is carried out.

The Competency Profile Definition Viewpoint, through the *include relationships* contributes towards an Organisation’s Need to ‘Provide organisational support for competency’.

**Viewpoint Definition View**

The definition of the Competency Profile Definition Viewpoint (CPDVp) is shown in Figure 41.
The Competency Profile Definition Viewpoint defines a Competency Profile (in fact, the realisation of a Competency Profile Definition Viewpoint, that is a Competency Profile Definition Viewpoint is a Competency Profile; they are the same thing).

The Competency Profile Definition Viewpoint shows the Competencies as measured for a particular Person filling a particular Stakeholder Role, along with the Level at which the Competency is held. The Competencies are usually grouped by Competency Area. Optionally, the Competency Scope (in terms of the required Level per Competency) can also be shown so that a comparison between the Competency Profile and the Competency Scope can be made explicitly on the Competency Profile.

**Example**

An example Competency Profile Definition View (CPDV) is shown in Figure 42 below.
The Competency Profile Definition View in Figure 42 shows a Competency Profile for a Person (‘Joe Bloggs’) that was the result of an assessment carried out on 2014-05-09 against a Competency Scope for the ‘Requirements Manager’ Stakeholder Role (this is the Competency Scope in Figure 39).

The actual Level at which each Competency is held is indicated by green or red bars. Green bars show Competencies that met or exceeded the required Level. Thus ‘Systems concepts’ is held at the required Level and ‘Integration and verification’ is held at a higher Level than required. The Red bars show Competencies that fell short of the required Level. Thus, ‘Validation’ was required to be at ‘Level 2 – support’ (as shown by the blue bar) but is only held at ‘Level 1 – awareness’.

4.2. The Competency Processes

This section defines the Competency Processes using the “seven views” approach defined in [D21.1 2012].

4.2.1. The Requirement Context View

The diagram in Figure 43 shows the Requirement Context View for the Competency Processes.
The main Use Case is to ‘Provide organisational support for competency’. This has two main included Use Cases that themselves have a number of inclusions.

The first inclusion of ‘Provide organisational support for competency’ is ‘Support definition of competency frameworks’. This is constrained by the need to ‘Allow use of existing frameworks’ and has the following inclusions:

- ‘Support definition of bespoke competencies’, which addresses the need to support the definition of competencies specific to the organisation, which may not be found in existing frameworks.
- ‘Support definition of bespoke competency levels’, which addresses the need to support bespoke levels at which a competency may be held, if suitable levels are not found in source competency frameworks.

The second inclusion of ‘Provide organisational support for competency’ is ‘Support carrying out of competency assessments’. This must allow such assessments to be carried out for a number of purposes:

- ‘Assess competency for staff appraisal’, where the assessment is undertaken as part of a regular staff appraisal process.
- ‘Assess competency for recruitment’, where the assessment is undertaken as part of a recruitment process.
- ‘Assess competency for self-assessment’, where the assessment is undertaken by an individual so that they can obtain an indication of their competency for a particular role or competency area.
- ‘Assess competency for education’, where the assessment is undertaken in order to perform an educational gap analysis.
- ‘Assess competency for accreditation’, where the assessment is undertaken as part of a professional accreditation process.
This Use Case, ‘Support carrying out of competency assessments’, has two inclusions:

- ‘Support definition of required competencies’, which addresses the need to define the competencies required for a particular role.
- ‘Support capture of held competencies’, which addresses the need to capture the actual competencies held by an individual.

The various Stakeholder Roles involved are described in the following section.

### 4.2.2. The Stakeholder View

The Stakeholder Roles involved in the Competency Processes are shown in Figure 44.

![Figure 44 - The SV for the Competency Processes](image)

The Stakeholder Roles are all sub-types of the abstract ‘Customer’, ‘External’ and ‘Supplier’ Stakeholder Roles, as follows.

**Sub-types of ‘Customer’ Stakeholder Role:**
- ‘Organisation’, which requires the ability to define or carry out competency assessments.
- ‘Human Resources’, which organises the carrying out of competency assessments.
- ‘Assessee’, which represents the Person being assessed.

**Sub-types of ‘External’ Stakeholder Role:**
- ‘Competency Framework’, which represents an external Competency Framework that is either used directly in an assessment or more forms the source for a bespoke organisational Competency Framework.

**Sub-types of ‘Supplier’ Stakeholder Role:**

---

68
• ‘Competency Framework Definer’, which carries out the definition of a bespoke Competency Framework.
• ‘Reviewer’, which is responsible for reviewing all artefacts produced during the definition of a bespoke Competency Framework.
• ‘Assessor’, which represents the role of those carrying out the competency assessment of an ‘Assessee’.

The Use Cases that these Stakeholder Roles are associated with can be seen in the RCV in Figure 43.

4.2.3. The Process Structure View

The Competency Processes use the standard Process structure as used throughout all the COMPASS Processes. This is shown in Figure 45.

![Figure 45 - The PSV for the Competency Processes](image)

Each ‘Process’ is made up of one or more ‘Stakeholder Role’, one or more ‘Activity’ and one or more ‘Artefact’. Each ‘Stakeholder Role’ is responsible for one or more ‘Activity’ which produces/consumes one or more ‘Artefact’.

4.2.4. The Process Content View

There are four Competency Processes defined, as shown in Figure 46.
Each of the four Processes is described briefly below. The Process Behaviour Views, showing how each Process is carried out, are shown and discussed in Section 4.2.5.

**Bespoke Competency Definition Process**
The main aim of the ‘Bespoke Competency Definition Process’ is to define the Competencies for a bespoke Competency Framework based on a source MBSE Ontology and Source Frameworks.

The Bespoke Competency Definition Process has the following Activities:

- ‘identify ontology’ – identify and if necessary model the source MBSE Ontology that forms the concepts against which Competencies will be defined.
- ‘identify source framework’ – identify and if necessary model any Source Frameworks that may form an input to the bespoke Competency Framework.
- ‘define competency areas’ – define the Competency Areas into which Competencies will be grouped.
- ‘identify competencies’ – identify each individual Competency that will be defined.
- ‘define concept-related competencies’ – from the identified Competencies, define those that are related to MBSE concepts (i.e. to those concepts from the source MBSE Ontology).
• ‘define process-related competencies’ - from the identified Competencies, define those that are related to Process and Life Cycle concepts (from the MBSE Ontology) and also to any Organisation-specific Processes for which Competency needs to be assessed.
• ‘define skill-related competencies’ - from the identified Competencies, define those that are related to technical and soft skills, such as use of a particular modelling notation or tool, presentation skills etc.
• ‘review’ – review all Artefacts produced as part of the Bespoke Competency Definition Process.

The Bespoke Competency Definition Process has the following Artefacts:
• ‘MBSE ontology’ – the MBSE Ontology on which the Competencies are based.
• ‘Source framework’ – the Source Frameworks that are used as a source of possible Competencies and Competency Areas.
• ‘Competency area’ – the defined Competency Areas into which the identified Competencies are grouped.
• ‘Competency’ – the identified Competencies.
• ‘Indicator’ – the indicators for each Competency. Each individual Competency will have a number of Indicators which are used during a Competency assessment to indicate whether an Assessee has the Competency.
• ‘Review results’ – the review results for all Artefacts reviewed in this Process.
• ‘Bespoke framework’ – the initial version of the bespoke Competency Framework, consisting of Competencies and their Indicators, grouped into Competency Areas.

All Artefacts are defined according to the Competency Framework.

**Bespoke Framework Definition Process**
The main aim of the ‘Bespoke Framework Definition Process’ is to take the outline Competency Framework produced in the Bespoke Competency Definition Process and complete its definition.

The Bespoke Framework Definition Process has the following Activities:
• ‘analyse bespoke framework’ – take the initial version of the Bespoke Framework and analyse in order to determine how best to define the Levels at which Competencies can be held.
• ‘define levels’ – define one or more Levels at which each Competency can be held. Note that in some Competency Frameworks, a Competency can be held at all defined Levels, whereas other Competency Frameworks contain Competencies that can only be held at a sub-set of the defined Levels.
• ‘review’ - review all Artefacts produced as part of the Bespoke Framework Definition Process.
The Bespoke Framework Definition Process has the following Artefacts:

- 'Bespoke framework' – the initial version of the Bespoke Framework that forms an input to this Process, which is then completed during this Process and which forms its primary output.
- 'Level' – the Levels at which Competencies can be held.
- 'Review results' – the review results for all Artefacts reviewed in this Process.

All Artefacts are defined according to the Competency Framework.

**Assessment Set-up Process**
The main aim of the ‘Assessment Set-up Process’ is to identify the needs and purpose of a planned Competency assessment (or set of assessments) and to define the Competency Scopes that will be used in the assessments.

The Assessment Set-up Process has the following Activities:

- 'identify assessment needs' – identify the needs and purpose of the assessment that is to take place. For example, is it for recruitment, staff appraisal, training needs analysis etc.
- 'define competency scope' – using the identified 'Assessment criteria' and the 'Competency framework', define the required Competency Scopes against which assessments will take place.
- 'review' - review all Artefacts produced as part of the Assessment Set-up Process.

The Assessment Set-up Process has the following Artefacts:

- 'Assessment criteria' – the identified needs, purpose and criteria for the planned assessments.
- 'Competency framework' – the Competency Framework from which Competencies will be taken to produce the Competency Scopes.
- 'Competency scope' – the Competency Scopes for a number of Stakeholder Roles that will be used during the carrying out of a Competency assessment for each Assessee.
- 'Review results' – the review results for all Artefacts reviewed in this Process.

All Artefacts are defined according to the Competency Framework.

**Assessment Process**
The main aim of the ‘Assessment Process’ is to carry out the Competency assessment of a Person (the Assessee) against a Competency Scope that has been produced, from a Competency Framework, for a specific Stakeholder Role.

The Assessment Process has the following Activities:

- 'collate assessment information' – gather the relevant ‘Competency scope’(s) and the ‘Competency framework’, together with necessary recording material and guidance material.
The Assessment Process has the following Artefacts:

- ‘Competency scope’ – the Competency Scope against which the assessment will be carried out.
- ‘Competency framework’ – the Competency Framework used as a source for the ‘Competency scope’ and used as guidance and reference material during the assessment.
- ‘Competency profile’ – the completed Competency Profile for the Assessee.

All Artefacts are defined according to the Competency Framework.

4.2.5. The Process Behaviour Views

This section contains the Process Behaviour Views (PBVs) for the Competency Processes.

**Bespoke Competency Definition Process**

The PBV for the Bespoke Competency Definition Process is shown in Figure 47.
The Process begins with the Competency Framework Definer who starts by identifying and, if necessary, modelling the source MBSE Ontology (‘identify ontology’. Any Source Frameworks are then identified and, if necessary, modelled (‘identify source framework’).

With the MBSE Ontology and any Source Frameworks identified and modelled, the Competency Framework Definer next defines Competency Areas (‘define competency area’) and identifies the Competencies that will be defined (‘identify competencies’).

These identified Competencies are then defined in detail, along with their Indicators and any descriptive material etc., through three parallel activities that look at concept-related, process-related and skill-related Competencies (‘define concept-related competencies’, ‘define process-related competencies’ and ‘define skill-related competencies’ respectively).
All the material is then reviewed (‘review’) by the Reviewer and a review report produced. If the result of the review is a fail, then the Process is repeated, starting with the identification of Source Frameworks. If the review is a success, then the initial version of the bespoke Competency Framework is issued.

**Bespoke Framework Definition Process**

The PBV for the Bespoke Competency Definition Process is shown in Figure 48.

The Process begins with the Competency Framework Definer who starts by taking the initial version of the bespoke Competency Framework and analysing it in order to determine how best to define the Levels at which Competencies can be held (‘analyse bespoke framework’). Having determined the appropriate Levels for each Competency, these Levels are then defined for each Competency (‘define levels’).

All the material is then reviewed (‘review’) by the Reviewer and a review report produced. If the result of the review is a fail, then the Process is repeated. If the review is a success, then the final version of the bespoke Competency Framework is issued.
Assessment Set-up Process
The PBV for the Assessment Set-up Process is shown in Figure 49.

Figure 49 - The PBV for the Assessment Set-up Process

The Process begins with Human Resources who identify the Assessment Criteria (the needs and purpose) of the assessment that is to take place (‘identify assessment needs’). These criteria, along with the bespoke Competency Framework, are then used to define the required Competency Scopes against which assessments will take place (‘define competency scope’).

All the material is then reviewed (‘review’) by the Reviewer and a review report produced. If the result of the review is a fail, then the Process is repeated. If the review is a success, then the Process ends.

Assessment Process
The PBV for the Assessment Process is shown in Figure 50.
The Process begins with the Assessor who gathers together any necessary material, such as Competency Scopes, the bespoke Competency Framework, guidance etc. (‘collate assessment information’). He then selects a Competency from the Competency Scope to assess next (‘select competency’), selects the Level for the Competency being assessed, from those Levels identified on the Competency Scope (‘select level’) and perform an assessment of the Assessee against an Indicator for the selected Competency at the selected Level (‘assess indicator’). This is repeated for all the Indicators in the Competency at the selected Level. If there are more Levels for the selected Competency then this
selection of Level and assessment of Indicators until all the Levels for the selected Competency have been assessed.

The Assessor then updates the Assessee’s Competency Profile with the results so far (‘update competency profile’) and all the above is repeated until all the Competencies in the Competency Scope have been assessed.

The Assessor then discusses the resulting Competency Profile with the Assessee (‘discuss assessment’) and the process ends.

4.2.6. The Information Views

Figure 51 shows a high-level Information View (IV) for the Competency Processes.

![Figure 51 - IV for the Competency Processes](image)

The key Artefact is the ‘Competency Framework’ that is composed of one or more ‘Competency Framework Definition View’ and one or more ‘Competency Level Definition View’.

One or more ‘Competency Framework Definition View’ define Competencies that can be held at Levels defined by one or more ‘Competency Level Definition View’.

The ‘Competency Scope Definition View’ defines a Competency Scope based on the Competencies from the ‘Competency Framework Definition View’ and the ‘Competency Profile Definition View’ shows a Competency Profile assessed against a Competency Scope defined by a ‘Competency Scope Definition View’.

The IVs for each individual Competency Processes are given below.

**Bespoke Competency Definition Process**
The IV for the Bespoke Competency Definition Process is shown in Figure 52.
The key Artefact produced by the Bespoke Competency Definition Process is the ‘Competency Framework’ that is composed of one or more ‘Competency Framework Definition View’. One or more ‘Source Framework’ and one or more ‘MBSE Ontology’ are used in the production on the ‘Competency Framework’.

Each ‘Competency Framework Definition View’ is made up of one or more ‘Competency Area’ and one or more ‘Competency’, which is itself made up of one or more ‘Indicator’.

**Bespoke Framework Definition Process**

The IV for the Bespoke Framework Definition Process is shown in Figure 53.
The key Artefact produced by the Bespoke Framework Definition Process is the ‘Competency Framework’ that is composed of one or more ‘Competency Framework Definition View’ and one or more ‘Competency Level Definition View’.

Each ‘Competency Framework Definition View’ is made up of one or more ‘Competency Area’ and one or more ‘Competency’, which is itself made up of one or more ‘Indicator’. Each ‘Competency Area’ classifies one or more ‘Competency’.

Each ‘Competency Level Definition View’ is made up of one or more ‘Level’.

Each ‘Competency’ is required at one ‘Level’ and is held at one ‘Level’.

**Assessment Set-up Process**

The IV for the Assessment Set-up Process is shown in Figure 54.

The key Artefacts produced by the Assessment Set-up Process are the ‘Assessment Criteria’ and the ‘Competency Scope Definition View’, each of which requires information from the ‘Assessment Criteria’.

A ‘Competency Scope Definition View’ is a subset of the Competencies from the ‘Competency Framework’ and defines a ‘Competency Scope’. It is made up of one or more ‘Competency Area’, one or more ‘Competency’, one or more ‘Level’ and one ‘Stakeholder Role’. Each ‘Competency Area’ classifies one or more ‘Competency’. Each ‘Competency’ is required at a ‘Level’ and each ‘Stakeholder Role’ requires a ‘Competency Scope’.

**Assessment Process**

The IV for the Assessment Process is shown in Figure 55.
The key Artefacts produced by the Assessment Process is the ‘Competency Profile Definition View’, the production of which is guided by the ‘Competency Framework’.

A ‘Competency Profile Definition View’ defines a ‘Competency Profile’. It is made up of zero or more ‘Competency Scope’, one or more ‘Competency Area’, one or more ‘Competency’, one or more ‘Level’, one ‘Stakeholder Role’ and one ‘Person’. Each ‘Competency Area’ classifies one or more ‘Competency’. Each ‘Competency’ is required at a ‘Level’.

The ‘Competency Profile’ describes the measured abilities of the ‘Person’ and is assessed against the ‘Competency Scope’.

### 4.2.7. The Process Instance Views

Two Process Instance Views (PIVs) are given here that illustrate, at a high level, how the Competency Processes can be sequenced in order to address the two main Use Cases in the Requirement Context View in Figure 43.

Figure 56 shows the PIV for the "Support definition of bespoke competency frameworks" Use Case. In order to satisfy this Use Case, the “Bespoke Competency Definition Process” is executed first. Once this is complete then the
“Bespoke Framework Definition Process” is executed as often as needed until the bespoke Competency Framework is considered to be complete and ready for use in the carrying out of a Competency assessment.

Figure 57 shows the PIV for the "Support carrying out competency assessments" Use Case. In order to satisfy this Use Case, the “Assessment Set-up Process” is executed first. This is repeated as often as necessary until all the Competency Scopes required in the Competency assessment have been defined.

Once this is complete then the “Assess Process” is executed once for each Assessee until the Competency assessment is complete.

### 4.3. Example Competency Scopes

A number of Competency Scopes relevant to SoSE are presented here. Each Competency Scope contains a number of Competencies taken from [INCOSE 2010]. Note, however, that the Levels defined above in Section 4.1.2 are used rather than those defined in [INCOSE 2010]. The Stakeholder Roles covered are:

- Configuration Manager
- Requirement Manager
- Project Manager
- Requirement Engineer
- Systems Modeller
- Tester
- Reviewer
- Process Modeller
- Builder
- SoS Engineer
These Stakeholder Roles correspond to those defined in the Systems engineering Processes in [D21.1 2012] and [D21.2 2013], with the ‘Architectural Framework Modeller’ Stakeholder Role from [D21.2 2012] being covered here by the more generic ‘Systems Modeller’ Stakeholder Role.

4.3.1. **Configuration Manager Competency Scope**

The following diagram shows a Competency Scope for the ‘Configuration Manager’ Stakeholder Role.

In this diagram, and those that follow, the relevant Competencies that apply to the Stakeholder Role as shown along the horizontal axis. The vertical bars show the minimum Level required for each Competency, against the Level scale on the vertical axis.

---

**Figure 58 Competency Scope for ‘Configuration Manager’ Stakeholder Role based on the INCOSE Systems Engineering Competencies Framework**
This Stakeholder Role is responsible for ensuring that all the System artefacts are correctly controlled, managed and configured. This will require a basic understanding of modelling, as it is the model itself, in a model-based approach, as well as the artefacts that are generated from it that will be held under configuration control. These artefacts may take on many different forms, such as: models, documents, hardware, software, etc.

**4.3.2. Requirement Manager Competency Scope**

The following diagram shows a Competency Scope for the ‘Requirement Manager’ Stakeholder Role.

![Competency Scope Diagram](#)

This Stakeholder Role will require good management skills but also an understanding of the Requirements engineering activities that are being used on Projects. The manager need not be an expert in this field but certainly needs to understand the fundamental of the work being carried out.
4.3.3. Project Manager Competency Scope

The following diagram shows a Competency Scope for the 'Project Manager' Stakeholder Role.

This Stakeholder Role describes the Stakeholder Role of the Person who will be in charge of the Project as a whole. Note that this Stakeholder Role requires good management skills along with a basic understanding of any areas that they will be managing. For example, if the Project Manager is overseeing a Project where an Architecture is being generated, then it is essential that the Person playing this Stakeholder Role has an understanding of what Architecture is.
4.3.4. Requirement Engineer Competency Scope

The following diagram shows a Competency Scope for the ‘Requirement Engineer’ Stakeholder Role.

The area of Requirements engineering is one that is fundamental to Systems engineering. The Stakeholder Role here has an emphasis on the understanding and modelling of Requirements and, therefore, will require Competencies that relate to Context modelling, Use Cases, Scenarios, validation and traceability. Unlike a traditional Requirements engineering Stakeholder Role, when adopting a model-based approach there is a strong need for modelling skills as well as understanding the fundamentals of Requirements engineering.
4.3.5. Systems Modeller Competency Scope

The following diagram shows a Competency Scope for the ‘Systems Modeller’ Stakeholder Role.

![Competency Scope Diagram](image)

This Stakeholder Role covers a multitude of activities and will, in reality, usually be split into a number of sub-types. Areas of expertise that must be covered here include understanding: interfaces, specification, design, testing, traceability, etc. This is perhaps the most loosely-defined of all the Stakeholder Roles here as the scope is so large. Nevertheless, it should be pointed out that the 'Systems Modeller' requires very strong modelling skills and these skills may be applied to any of the aforementioned activities. Therefore, it is possible for the 'Systems Modeller' to require a high level of Competence in almost any area, depending on the nature of the work.
4.3.6. **Tester Competency Scope**

The following diagram shows a Competency Scope for the ‘Tester’ Stakeholder Role.

![Competency Scope Diagram]

This Stakeholder Role is primarily involved with the verification and validation activities that are applied throughout the Life Cycle. Again, the Competencies necessary for this Stakeholder Role may differ depending on the type of testing activities required.
4.3.7. Reviewer Competency Scope

The following diagram shows a Competency Scope for the ‘Reviewer’ Stakeholder Role.

This Stakeholder Role is essential for all aspects of model-based Systems engineering (MBSE). Interestingly, there are two main variations on this Stakeholder Role (not shown on the diagram) that cover “mechanical reviews” and “human reviews”. A mechanical review is a straightforward verification review that does not require any real human input but simply executes a pre-defined rule. Examples of these include SysML syntactical checks and checks based on a Process. These mechanical reviews tend to be quantitative in that they can be measured in terms of numbers or values and, very importantly, they may be automated. This is essential for MBSE as it is one of its key benefits. The human reviews require reasoning and will tend to be qualitative and are typically very difficult, if not impossible to automate using a tool. The ‘Reviewer’ Stakeholder Role will require a good understanding of any area in which they are involved with reviewing.
4.3.8. Process Modeller Competency scope

The following diagram shows a Competency Scope for the ‘Process Modeller’ Stakeholder Role.

![Competency Scope Diagram]

Figure 65 Competency scope for ‘Process Modeller’ Stakeholder Role based on the INCOSE Competencies Framework

Having a well-defined Process is crucial when defining any approach to work and, in-keeping with the MBSE philosophy, this Stakeholder Role requires good modelling skills as well as an understanding of Process concepts and the business. The Stakeholder Role of the ‘Process Modeller’ will also require a good understanding of any areas in which the Processes will be either defined or applied, therefore, it is possible for the ‘Process Modeller’ to require a large number of Competencies.
4.3.9. **Builder Competency Scope**

The following diagram shows a Competency Scope for the ‘Builder’ Stakeholder Role.

![Diagram showing Competency Scope for Builder Stakeholder Role](image)

This Stakeholder Role is concerned with taking the model and turning it into a real System. This will include building System Elements, integrating them into the System itself, installation and so on. Of course, this is another Stakeholder Role that on real Projects may be broken down into a set of lower-level Stakeholder Roles with different skillsets and, hence, different Competency Scopes.
4.3.10. **SoS Engineer Competency Scope**

The following diagram shows a Competency Scope for the 'SoS Engineer' Stakeholder Role.

![Competency Scope Diagram](image)

The Stakeholder Role of the 'SoS Engineer' is one that may be used in conjunction with any of the other Systems engineering Stakeholder Roles in order to elevate it to the level of SoS. Key skills here will include Integration, understanding of Requirements and verification and validation.

4.4. **Summary**

This Section has presented the COMPASS Competency Framework, together with Processes that can be used by an Organisation in the definition of its own Competency Framework. In addition, a number of Competency Scopes have been presented that are relevant to SoSE.

Two of the authors have successfully used a version of this Competency Framework and Process with a number of industrial partners, including assessing over 60 members of staff from a large SoS procurement Organisation.

An area for further research is that of team Competency Scopes and Profiles: how the Competency Scope for an entire team can be defined and assessed against the Scopes and Profiles of its members.
5. Tool Support

This section describes how the various tools being used and developed on the COMPASS project (Atego Process Director, Artisan Studio and the CML Toolset) support SoSE through the COMPASS approach. Three subsections cover:

- Artisan Studio – Support for SysML modelling of Systems; realisation of Viewpoints and enabling patterns through SysML; generation of CML from SysML.
- CML Toolset - reasoning about models

5.1. Atego Process Director

In order to ensure the validity and applicability of the COMPASS approach described in this document it is essential that it 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 COMPASS approach has been and will continue to be used by the industrial partners as part of their case study work investigating COMPASS methods in tasks T4.1.2 and T4.2.2, and as part of the industrial challenge problem in task T4.3.3.

The COMPASS approach, both 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 are 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.

5.2. Artisan Studio

Artisan Studio provides two capabilities to the tool chain. The first is the ability to enter a model of a SoS, using SysML that conforms to the SysML standard. This is entered predominantly using diagrams. The second is the subsequent ability to transform this SysML model into a text based CML model for use by the rest of the tool chain as detailed in Section 5.3. This will enable complex models to be entered more quickly and with fewer errors than using the text based CML. It will also result in a better overview of the SoS (and one more amenable to be understood by those who do not know the CML language).
Previous to the COMPASS project, Artisan Studio supported the ability to build SysML models. Artisan Studio also has a capability to extend the range of what can be modelled using UML profiles, and the ability to change the way models are checked, entered by (and displayed to) users using an extension to this – Artisan Studio ergonomic profiles. These also provide the ability to define (multiple) additional Viewpoints to those naturally provided by SysML.

As part of the COMPASS project a COMPASS profile is being developed that can be installed subsequent to Artisan Studio’s own installation, after which it is offered as an option on new models, and can also be applied to existing models. The intent is to extend the SysML to make it more suitable for COMPASS use.

To date the main elements of the profile are the various basic types in CML (which are also therefore needed in a SysML model capable of representing CML concepts). It is envisioned that the profile will continue to be developed to support higher level concepts concerning SoS permitting faster and more accurate entry of such models; for example, the profile is being extended to include Viewpoints that cover SoS Requirements engineering as well as the Viewpoints of the CAFF.

Previous to the COMPASS project, Artisan Studio supported the ability to generate code from models into various languages (but not CML), through a code generator called ACS (Automatic Code Synchronizer). Once invoked, model changes automatically trigger (without further user intervention) the update of any and every code file that needs to be changed (if and only if it need to be changed).

Artisan Studio also has the capability to change the code generation algorithms and add new ones using the TDK (Transformation Development Kit). TDK, like ACS itself, acts on a model to produce a generator (i.e. the behaviour of the generator is itself modelled).

As part of the COMPASS project a new generator is being developed that translates SysML models to CML, according to the rules defined in [D22.4 2013]. As explained above, this consists of a model of those rules which the TDK then translated into a generator. Like the profile, this new generator can be installed subsequent to Artisan Studio’s own installation, after which the new CML code generation algorithm will be offered to users alongside those for other languages. The generator is not intended to generate CML from all SysML constructs – for example some areas of SysML are insufficiently formal for this to be feasible – comprehensive details of which SysML constructs are covered is provided in [D22.4 2013], along with rationale. In summary though, the rules describe a translation providing a high level of coverage of the following:

- Structure. This consists of blocks, parts and most other elements usually shown on block definition diagrams and internal block diagrams.
- State behaviour. This consists of states, transitions and most other elements usually shown on state machine diagrams.
- Activities. A subset of actions and flows – a subset of the elements usually shown on activity diagrams.
- Scenarios. A subset of the elements usually shown on sequence diagrams.

To date, the generator implements all of the structural, state behaviour and activity rules, and about half of the rules concerning sequence diagrams. Earlier versions of this generator have been issued to COMPASS partners for feedback [Generator within COMPASS SVN at Theme2\WP22\T2.2.2\CMLGenerator] and at least one further drop will be made.

When using the generator it is important to note that relevant changes made to the SysML model will overwrite changes made directly to the CML code. In other words the decision to use the generator is also a decision to make the SysML model the master, and the CML model a follower. Changes should therefore be entered exclusively in the SysML model for as long as the generator continues to be used (whether or not the generator is running at a particular time). Future versions of the generator might include a limited roundtrip capability that will allow changes to aspects of the CML code that exist in the same (textual) form within the SysML model (text fields) to be made at either SysML or CML level. However any significant structural changes (changes affecting more than the content of text fields in SysML) will still need to be made at the SysML level.

5.3. CML Toolset

Whilst CML is used almost exclusively to model elements such as CSs, protocols etc., we observe that SysML is used in at least three different ways: Process modelling, meta-modelling, System modelling. In principle, provided the resulting models of these three activities are consistent with the Guidelines provided in [D22.4 2013], it is possible to generate a CML model. However, since the semantics proposed in [D22.4 2013] targets specifically the System modelling activity, its usefulness for Process modelling and meta-modelling may be limited and require customisations.

In what follows, we briefly present the tools that are being developed to support CML, and discuss possible applications of these tools with respect to SysML models by means of the translation tool discussed in Section 5.2. The discussion on the tool support for CML is based on [Coleman et al. 2012].

At the most basic level, CML is supported by an integrated development environment (IDE) that provides facilities to parse and type check CML models. Additionally, features such as project management and editing common to many IDEs are provided. Finally, the basic IDE can be extended by a number of plug-ins that provide additional support for both analysis and simulation.

Whilst the parser and type checker provide some assurances regarding the well-formedness of a CML models, to effectively establish the consistency of a model it is necessary to verify a number of conditions known as proof obligations. These
are generated by a proof obligation generator plug-in, and verified by the theorem prover that is being integrated into the IDE. The theorem proving facilities available for the COMPASS IDE are based on the Isabelle/HOL theorem prover [Paulson 1994] and support verification based on proof tactics specialised for CML as well as the existing facilities provided by Isabelle.

Whilst, due to the level of generality, the theorem prover may require some level of expertise from the user for the verification of specific assertions (e.g., deadlock freedom), for a smaller class of models, it is possible to obtain a higher level of automation of verification tasks through a model checker. The CML model checker will support the verification of refinement statements to which CML processes are compared.

On top of the notion of refinement available in CML, a refinement calculus is developed to support the step-wise refinement of CML models. This is achieved by the successive application of proven refinement laws to derive correct-by-construction models. The correctness of the refinement is a consequence of the correctness of the laws as well as the transitivity and compositionality of refinement. In order to support the CML refinement calculus, a refinement plug-in is being developed with the theorem prover as a basis.

Finally, fault tree analysis support is also planned at the level of SysML as well as CML. At the SysML level, this support will be based on the generation of fault trees (from annotated SysML models), which are to be analysed through external tools.

This set of tools provides support for the analysis of SysML models through the translator discussed in Section 5.2, which implements the semantics of SysML described in [D22.4 2013]. As previously mentioned, our semantics of SysML targets the System modelling activity, and as such may not be applicable to Process modelling and meta-modelling. The reason for this is that our semantics requires a level of detail of the SysML model that is usually not present in Process models and meta-models.

For instance, in a Process model, activities are often specified by natural language statements, and this would need to be further detailed in order to allow the generation of meaningful CML models. It is worth mentioning that such detailing is not necessarily possible, and where it is, it might not be desirable. This is because the type of analysis support required by Process modelling and meta-modelling is often different from that of System modelling.

One possible use of the CML toolset with respect to Process models is the simulation of the Processes as a means of visualising their execution. In this case, instead of detailing the Process model into a concrete model (that can be translated into CML), it may be worth substituting the high level activities by signals. This way, the simulation of the generated CML model would simply show which activity (as a signal) is supposed to be executed at any time.
This approach can also support the verification of the consistency of different Processes. In this case, dependencies between Processes can be modelled as synchronisation and the consistency may be specified as deadlock freedom (provided the individual Processes are themselves deadlock free).

Notice, however, that our semantics of SysML does address this model transformation. The Process model must be transformed into a suitable SysML model before CML code is generated and analysed. Whilst outside the scope of this project, the investigation of the potential application of the CML tools to restricted SysML patterns that aim at describing Process models is an interesting topic and is left as future work.
6. Application to COMPASS Case Studies

6.1. Scope

This section will describe the use of the COMPASS approach by Bang & Olufsen A/S for improvement of their SoS development processes. The B&O case study section of this document will describe an industrial model-based SoS/CS development process. The development process is starting from needs to SOS/CS product deployment, with parallel testing, validation and verification activities for the different development stages of the process. For each stage of the MBSE development process, a set of COMPASS technologies has been applied. The following sections will describe how and why these COMPASS technologies are applied, and the cons and pros of applying the technologies to the B&O case study. The case study section starts with defining B&O’s SoS problem domain and the case study selected as SoS problem domain representative for this case study section of the D21.6 document.

For educational and confidentiality reasons some models and explanations are simplified. This section will describe the use of both semi-formal and formal COMPASS technologies in an industrial context, but it’s not the intention to provide detailed user guidelines or in deep technology descriptions for the applied COMPASS technologies. Descriptions of the applied COMPASS technologies are found in other COMPASS documents, and references are provided there needed.

Note during this case study section, the term refinement reference to the refinement concepts from the COMPASS Refinement Framework [D22.4]. The term formal refinement refers to COMPASS’s formal refinement arguments [D22.4].

6.2. Introduction

B&O is known for design, high product quality and distinguished user experiences. B&O is defined as a “lifestyle brand”, and the B&O brand is routinely ranked in the top 10 coolest brands worldwide. B&O manufactures consumer electronics products and technology for four core business areas,

- **Audio and Video**

---

3 A lifestyle brand is a [brand](#) that attempts to embody the values and aspirations of a group or culture for purposes of marketing. Each individual has an [identity](#) based on their choices, experiences, and background (e.g. ethnicity, social class, subculture, nationality, etc.). A lifestyle brand aims to sell product by convincing potential customers that this identity will be reinforced or supplemented if they publicly associate themselves with the brand.
High-end AV products with proprietary connectivity and SoS functionalities between AV and speaker products

- **Automotive**
  - Advanced car speaker systems, deployed in high-end car brands such as Aston Martin and Audi

- **Speaker technology**
  - In addition to distinctive design, the many featuring areas which differentiate B&O’s loudspeakers include integrated amplifiers, ICEpower technology and Acoustic Lens Technology

- **Play**
  - A youth sub brand, which includes products based on standard technologies and ecosystems like Airplay, DLNA, and Android.

A B&O SoS can contain products from all four business areas, and depending on the product configurations, new emergent behaviours will occur and extend the users’ experience space. This is a B&O brand characteristic and a B&O product requirement. The B&O brand experience rule (BER) is defined as “1+1=2+n”, meaning if you have two products (the 1(s)), you get the operational experience from each product (the number 2) and a set of new experiences (n defines the emergent behaviours set). In B&O terminology, the set n is called the extended system experiences, derived from the systems elements of the products in a given B&O SoS.

Historically, B&O has made proprietary HW and SW solutions for dealing with the interoperability challenges consequent on the BER product requirement. Using COMPASS SoS definitions [1], B&O SoS would just be a constituent system, because all system elements are manageable by nature of the underlying proprietary technology. The use of proprietary technology in B&O is causing interoperability challenges with AV 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 experiences 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

---

4 AirPlay (previously called AirTunes when it was for audio only) 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

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

6 Android is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers. Initially developed by Android, Inc.
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 management of emergent behaviours 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 is now heavily influenced by the constituent systems and the contracts among the constituent systems. In addition, unmanaged emergent behaviours might even degrade the B&O SoS and leading to brand damages.

The open philosophy is causing a paradigm shift from system to SoS engineering in the B&O development organization. B&O is going from developing constituent systems to SoS level engineering. The organization evolution requirements are also driven by hostile business and technology domains represented by constituent systems of the SoS. The organization must be in a state where reasoning and validation regarding impact of uncontrollable constituent systems can be communicated among different disciplinary domains in the organization.

B&O’s SoS business and technology challenges define a set of COMPASS expectations, which needs are elicited from and used in COMPASS technology evaluation.

The B&O COMPASS expectations are

- that the COMPASS results will allow them to formally manage SoS requirements for emergent behaviours
- that the COMPASS work will have a positive impact on the B&O SoS software development process by improving software quality and SoS robustness
- in the long-term, that the methods and tools made available by COMPASS will bring together the various project groups that today work quite independently, streamlining software development, and allowing them to work on a single SoS

The B&O COMPASS expectations are translated into following high-level development process needs.

- SoS Requirement Stage Requirements
  - RS_REQ_1: Identification and definition of SoS requirement stakeholders
  - RS_REQ_2: SoS stakeholder requirement impact analysis
  - RS_REQ_3: SoS/CS requirement traceability from/to stakeholders and sources
  - RS_REQ_4: SoS requirements that conform to standard requirement quality attributes as understandable, consistent, testable and traceable.
- RS_REQ_5: Consistent SoS requirement communication among different stakeholders

- SoS Architectural Design Stage Requirements
  - DS_REQ_1: Consistent SoS architectural level communication, development, and decision making among different development disciplines
  - DS_REQ_2: Capability of expressing and analyzing SoS models during early design stages
  - DS_REQ_3: Capability of identified areas of incompleteness or ambiguity in informal system specifications during early design stages
  - DS_REQ_4: Capability of validation and testing of conceptual design models

- SoS Testing Stage Requirements
  - TS_REQ_1: Capabilities of identified needed SoS test cases for different AV product configurations
  - TS_REQ_2: Capabilities of testing SoS properties in different AV product configurations

The COMPASS approach provides a methods-and-toolbox for creating a SoS domain specific development process. Figure 68 shows the key elements of the COMPASS approach.
The colour scheme in the KEY box on the diagram defines the four key elements of the approach (The concepts of the approach has been described previously in this document).

- The System (colour blue) we are developing or want to communicate/understand. A System can be a System of Systems or a Constituent System
- The Approach (colour green) consist of Ontology(s), Process(s) and Framework(s) which enabled development/understanding of a System
- The Approach is enabled by Technique(s) or Pattern(s) (colour skin), which can be modelling languages (SysML, CML), Enabling pattern(s) or refinement Point(s).
- Tool(s) (colour red) enabled the Approach by being operation realization of Technique or Pattern elements

Figure 69 shows the overall MBE development process ontology developed and used for addressing B&O’s case study needs.

![Diagram](image)

**Figure 69 The B&O COMPASS Approach based development process**

The MBE process are the result of applying the COMPASS Approach to the B&O case study SoS development process needs. The development process consist of three main development stages with following COMPASS Approach System concepts (the blue elements in Figure 69) and relationships.

- The Conceptual stage contains the identification/understanding of Need(s). The Needs(s) are capture in the Requirement Model. The
Requirement Model drives the Architectural Design Model. The Architectural Design Test Model(s) which are refinements of the Requirement Model, validates the Architectural Design Model.


- The Real stage contain the B&O Product(s), which is a specialization of the COMPASS ontology concept System. A Product consist of software System(s) and Hardware System(s). A Product can be generates from an Architectural Design Specification Model or being analyse by an Architectural Design Specification Model. A Product are been verified by the Architectural Design Specification Test Model. Software Systems(s) are deployed on to Hardware System(s).

Please note the development model does not have a specific testing stage. The testing activities are imbedded into each stage. Testing does not only happen at the end of development, but runs parallel throughout the development process. Hence different development stages have different testing needs and scopes.

The following sections will describe in detail the motivation for the different development stages and the results (the blue coloured COMPASS Approach ontology concepts) of applying COMPASS technologies for a giving stage. This means describing the red, skin and green coloured ontology concepts on the diagram in Figure 69. The descriptions of the MBSE process will start with defining the AV SoS characteristics using the COMPASS SoS definitions in D11.1 and D11.3. This is the first step in understanding and communicating the fundamental case study’s Need(s) and the AV SoS problem domain.

### 6.3. Need(s) And SoS Characteristics

This section will describe how to scope the Need(s) part the Conceptual stage in the MBSE process by using COMPASS technology. Before captured Need(s) in the Requirement Model. We must enabled communication and understanding of the constraints and possibilities cause by the technology and business stakeholders in the SoS environment.
In the B&O case study development process, the COMPASS Concept Base [D11.2, D11.3] are used as base ontology elements for the SoS domain. The COMPASS Concept base combined with the guideline processes giving in Initial Report on Guidelines for Architectural Level SoS Modelling [D21.2 2013], enabled consistent communication regarding various SoS aspects in the B&O SoS environment. By giving abstract conceptual descriptions of SoS elements like SoS classifications, dominant SoS architectures, SoS properties and the domain specific concepts (ontologies). The COMPASS deliverables enabled the possibility of reasoning and decision making regarding needed capabilities, and limitations among technically and non-technically stakeholders in the B&O organisation. The following AV SoS knowledge has been captured using COMPASS Guidelines.

B&O SoS classification
The SoS can be considered a virtual SoS, since there is no central management authority or agreed-upon purpose. However, at times, the system enters a state where a designated manager is selected and constituent systems recognise some global objectives, which we regard as collaborative.

B&O SoS dominant Architecture
Service-oriented architecture (at the application layer). Constituent systems may offer services, consume services or act as a broker for discovering services available within the system and the policies that are attached to them. Other layers may exhibit other architectures – for example, the streaming layer is characterised as a pipe-and-filter architecture.

B&O SoS properties
Here we consider the B&O case study in relation to the eight COMPASS properties of SoS.

- High autonomy.
  - Some constituents are supplied by third party manufacturers or are standalone products produced by individual teams at Bang & Olufsen.

- High independence.
  - The components each have a purpose (e.g., a television set, a speaker) which means it can continue to have a meaningful existence if the SoS is removed.

- High Distribution.
  - Constituents are distributed within (a) room(s).

- Swift Evolution.
  - The SoS evolves substantially over a short number of years as new technologies make it possible to stream new types of data to new types of device.

- Considerable emergence.
  - The emergent behaviour is the ability to stream and manage content between distributed devices, associated with a user
account. Users can uncover new emergent behaviour, by finding new combinations of content server and playback device. Unexpected emergent behaviour must still conform to brand expectations.

- **High dynamicity.**
  - New unexpected devices and/or content may be added to the system or removed from it at any time, requiring the system to incorporate a new constituent into the architecture or cope with the loss of a former constituent.

- **Strong interdependence**
  - All constituents recognise the need to make sacrifices where necessary to abide by DRM regulations. Some constituents have a dependency on others – for example, devices for rendering content have a dependency on devices that store, stream and manage content and the relevant policies.

- **Some interoperability.**
  - A third party protocol for data exchange is employed and understood by many devices. There are many different standards, however, and not all products are capable of interoperate freely.

Based on the SoS characteristics of the AV SoS and B&O’s business need [D42.1] we can summarized the **SoS business challenges** as **SoS technology integration challenges** which simplified states: “Seamless integration of none-B&O system(s), while preserving B&O brand DNA”. Brand **B&O DNA** means B&O’s unique selling points like extended user-experiences and high product quality in a SoS context, even then non-B&O products being part of an AV SoS configuration.

![Figure 70 B&O SoS level requirement description view](image)

The B&O SoS level Requirement Description view in Figure 70 shows the concept of **B&O brand DNA** map into three SoS level user-experience requirements: Audio-Visual streaming, Availability and consistency of system configurations.
and remotely located contends browsing. The Seamless integration of none-B&O system(s) concepts are represented as the SoS level requirements Constituent system integration, White-box integration and Grey-box integration on the view.

The abstract conceptual descriptions of SoS elements produces in the Need step are further used in the development process. The Need products acting as base input for realization of Viewpoint(s) like Source Elements and Context Definition views from the SoS-ARCE framework. The Need step’s products also act as the foundations for development of domain specific ontology views.

6.4. Requirement Modelling

The Conceptual stage contains The Requirement Model. The Needs(s) are capture in the Requirement Model. The Requirement Model must therefore be a vertical refinement of Need elements, and the Requirement Model must satisfy following development process needs regarding SoS requirement engineering:

- RS_REQ_1: Identification and definition of SoS requirement stakeholders
- RS_REQ_2: SoS stakeholder requirement impact analysis
- RS_REQ_3: SoS/CS requirement traceability from/to stakeholders and sources
- RS_REQ_4: SoS requirements that conform to standard requirement quality attributes as understandable, consistent, testable and traceable.
- RS_REQ_5: Consistent SoS requirement communication among different stakeholders

B&O uses the SoS-ARCE framework for realization of the requirement models in the development process. The Viewpoint(s) from SoS-ARCE provides the processes for creating the concrete requirement views. The set of instantiated views individually or combined meets one of more of B&O’s development process requirements. As proof of concept, the rest of this section will describe the results of applied SoS-ARCE to the SoS level user-experience requirement Audio-Visual Streaming from the SoS Level Requirement Description view in Figure 70.

First, we will describe how SoS-ARCE Viewpoints are uses for the refinement between Need(s) and the Requirement Model, meaning the realization of the “Needs drives the Requirement Model” relation from the MBSE development process in Figure 69. Figure 71 shows an isolated Requirement Description View (RDV) containing the Audio-Visual Streaming SoS level user-experience requirement, and the SoS/CS integration requirement. This RDV scopes the needs the following sections will address.
All elements identified and produced in the Need step (Constraints, BS needs, Stakeholders, ... ) are modelled as Source Element(s) in a Source Element View (SEV). This refinement from Need elements to Source elements establishes traceability in the System and provide links between the Needs and any other aspect of the System. Figure 72 shows the SEV for the Streaming aspect of the B&O SoS.

Figure 71 The SoS level user-experience requirements for “Audio-Visual Streaming”

Figure 72 B&O SoS level Source Element View

A Definition Rule Set Viewpoint is used for translated the SEV’s Source Element(s) into requirement descriptions that conform to standard requirement quality attributes as understandable, consistent, testable and traceable (RS_REQ_4). The Definition Rule Set Viewpoint contains any Rules that may have to be applied to each requirement description. For example, these Rules may be complexity Rules in the form of equations or more general text-based Rules, which provide well-define domain specific semantics meaning to requirement
descriptions. Figure 73 shows a subset of the Rules from the Definition Rule Set View for the streaming aspect of the B&O case study.

In a formal way of thinking, we can consider DRSV(s) as a domain specific semantics translation functions from Source element(s) to Requirement Description(s) in a Requirement Description View. The rule based semantic translation function can provide testable (RS_REQ_4) and consistent communication (RS_REQ_5) regarding requirement descriptions. Figure 74 display a subset of the RDV for the Streaming aspect. Note how the Source Element Audio-Visual Streaming has be further refine into the S_REQ_1 and S_REQ_2 requirements and the descriptions has been giving stronger domain semantics meaning using Rules from the RDV.

One of B&O development process needs for requirement engineering stated the need for Identification and definition of SoS requirement stakeholders (RS_REQ_1). From the Source Element(s), which described the business needs for integration
of non-B&O products. The SoS-ARCE framework provides a Context Definition Viewpoint (CDV) used for modelling the relevant SoS stakeholders. The CDV identifies the points of view that are further explored in the Requirement Context Viewpoint (RCV).

In Figure 75 above we see a CDV for the stakeholders identified as relevant for realization of the B&O Streaming aspect needs. Note that each CS Stakeholder represent both business and technology stakeholders (Stakeholder definitions (RS_REQ_1)). Each stakeholder represent possibilities and limitations. The stakeholder’s possibilities or limitations might have impact on B&O’s business needs, and must be validate. Hence must meet the SoS stakeholder requirement impact analysis requirement defined in RS_REQ_2.

The SoS-ARCE framework provides the Requirement Context Viewpoint (RCV) and the Context Interaction Viewpoint as a model based solution for B&O’s “stakeholder requirement impact analysis” development process needs. The Requirement Context Viewpoint takes the Requirement Description(s) (representing Needs) and gives them meaning by looking at them from a specific stakeholder view (a Context described in the CDV). Needs in Context become the Use Cases. Formally thinking, a Use-Case is a domain specific refinement of a giving Requirement Description, there \( \left( us^{\text{sat}} \sqsubseteq rd, \, us \in USECASE, \, rd \in RD \right) \) might be true, if \( us \) is a valid refinement of \( rd \). The stakeholder will redefine the Requirement Description(s) with the stakeholder’s domain specific capabilities and constrains. The Use-Case(s) can hereby communicate the stakeholder’s impact on the requirements, and enable production of consistent requirements across stakeholder domains.

Correct use of RCVs and the RCV processes can satisfy B&O’s development process needs defined in RS_REQ_2 and RS_REQ_4. Figure 76 shows a RCV for the SoS technology stakeholder Source CS.
The process driving the production of the CDVs is a map-reduce/ divide and conquer approach with following process:

1. forall \( r:REQ \) do
2. identify the main stakeholder \( x \) of \( r \)
3. let \( x \) redefine \( r \) to \( r' \): Use Case, with \( x \)'s domain specific capabilities
4. identify the set of secondary stakeholder \( Y \) of \( r' \), for all \( y \) of \( Y \), start step 3 with \( x = y \) and \( r = r' \)
5. perform req impact analysis and consistent check between all Use Case variants of \( r \)

The problem with this process is that the level of stakeholder requirement refinement coverage depends on our understanding of the relationships between the stakeholders (Step 2 & 4). Concrete view instances of SoS-ARCE.’s Context Interaction Viewpoint (CIV) provides an overview of the relationships between the stakeholders (Contexts) that makes up the SoS. The CIV combined with the CDVs can provide the needed information to ensure full stakeholder requirement refinement coverage. Full stakeholder requirement refinement coverage insures all requirements are consistent and all stakeholder’s impact on requirements are known. This is extreme useful information for B&O because of the SoS level need for integration of none-B&O product. These stakeholder often have a negative impact on what B&O can offer to use in different product configurations. Figure 77 show a B&O CIV for the Streaming aspect of the case study. In D21.4, the reader can find a detailed B&O case study application. The application shows how CIV and CDVs are used for eliminating integration of Airplay products into the B&O SoS. The elimination is caused by the Airplay stakeholder’s negative impact on the requirements.
The requirement RS_REQ_4 states that: *SoS requirements that conform to standard requirement quality attributes as testable.* SoS-ACRE provides the Validation Viewpoint (VV) for producing testable requirements. The Validation Viewpoint provides the basis for demonstrating that the Needs, via the Uses Cases can be validated. Views produced can be informal (such as scenarios at various levels of abstraction) or may be formal (such as mathematical-based representation). The VV(s) are behavioural refinements of the more structural Use Cases. The VV(s) besides showing a given Use Case is testable also enables communication regarding the behavioural aspects of the Use Cases. Often when B&O are in the process of producing RCV(s), they uses VV(s) in parallel for ensuring the behavioural aspect of a requirement descriptions and the produced RCV are well understood. B&O uses SysML sequence diagram as modelling technique for VV(s). Sequence diagram has the needed abstraction level, and can be understood by non-technical stakeholders. Figure 78 show a VV for the Source CS Use Case.
One VV shows one stakeholders view of a requirement is testable (in form of a Use case). However, are all VV(s) for the same Requirement descriptions still testable then there are combined? In addition, which VV(s) do we need to combine? This problem are often seen in the B&O’s SoS domain. One CS c’s Use Case u, might be testable in c’s context, and another CS c’ Use Case u’ are testable in the context of c’. However, at a SoS level the combining VV(s) of u and u’ is not testable because of some constraint for one of the Use Case(s) u or u’, which are only true in a certain state. This problem is close related to the interdependence and independence properties of the B&O SoS.

The SoS-ARCE framework provides the Validation Interaction Viewpoint (VIV) to address these SoS level consensus problems. The Validation Interaction Viewpoint provides a combined view of the scenarios for Use Cases that are involved in the SoS. Context Interaction Viewpoint instances provide SoS level requirement consistent for all CDV view instances. The VIV view instances ensure all VV(s) are testable at a SoS level. As VV(s) are behavioural refinements of CDV(s), VIV(s) are behavioural refinements of CIV(s). VIV and CIV are especially useful for modelling structural and behavioural aspect of emergence properties. Figure 79 shows VIV for B&O’s desired emergent properties contents streaming.

![Figure 79 Streaming SoS VIV](image)

The SoS-ARCE framework meets all B&O development process needs defined for SOS requirement engineering. This section has presented the process of working with SoS-ARCE in a kind of waterfall way (going from production of one view to the next view). In reality SoS-ARCE are much more agile. One of the great things with the framework is that you can use one view to gain the needed information for production of another view. The framework allows production of views in parallel and still preserve requirements quality because of the naturally communication synchronization point there exist between View Point(s) elements. B&O has use SoS-ARCE Views for creating a Concept Clarification Process. B&O developed new product concepts, which there in the beginning exit
no needs for. So that process start with defining USE-CASES \((V\&V(s)\leq CDV(s))\), which helps identify needed stakeholders. The first draft of the USE-Case(s) end up acting as source elements input to SoS-ARCE. The Concept Clarification Process is basic a model-based method for producing the elements for the Need step of the development process. For SOS requirement engineering the SoS_ARCE framework provides views point for CS and SoS level requirements production, and views for ensuring requirement consistency between the CS/SoS requirement levels. This mean SoS-ARCE enabled the needed abstraction for enabled SoS communication, despite the complexity of a giving SoS domain.

The next section will describe how different COMPASS technologies has been uses for production of B&O Architectural Design Model(s) and B&O Architectural Design Test Model(s).

### 6.5. Architectural Modelling

This section will describe which COMPASS technologies applied for realization/production of Architectural Design Model(s) and Architectural Design Test Model(s) in the B&O COMPASS Approach based development process in Figure 27. The Architectural Design Model(s) and Architectural Design Test Model(s) must be vertical refinements of the Requirement Model. The MBSE development ontology elements states following relationships among the models.

- The Requirement Model drives the Architectural Design Model(s). The Architectural Design Test Model(s) are refinements of the Requirement Model and validates the Architectural Design Model.

The applied COMPASS technologies must satisfy following conceptual stage development process requirement regarding SoS test and design models development.

- SoS Architectural Design Stage Requirement(s)
  - DS_REQ_1: Consistent SoS architectural level communication, development, and decision making among different development disciplines
  - DS_REQ_2: Capability of expressing and analyzing SoS models during early design stages
  - DS_REQ_3: Capability of identified areas of incompleteness or ambiguity in informal system specifications during early design stages
  - DS_REQ_4: Capability of validation and testing of conceptual design models

Domain specific Architectural framework

DS_REQ_1 state the need for enabled *Consistent SoS architectural level communication*, ..., This need is based on the SoS level requirements Constituent System Integration from Figure 70. The CSs of the SoS represent different
architectures which must be consistent modelled. The CS models thereby insure consistent communication regarding architectural integration constraints and possibilities. We can redefine DS_REQ_1 as Needs for development of an AV Domain specific Architectural framework. The AV Domain specific Architectural framework must contain a set of structural and behavioural Viewpoints tailored for the AV SoS domain. The AV Domain specific Architectural framework’s viewpoint elements must be giving a well-defined domain semantics for ensuring consistent communication and insure testable validation.

B&O has used the COMPASS Architectural Framework Framework (CAFF) for satisfying these needs. B&O has developed the Streaming Architectural Framework (SAF) using CAFF. Several instantiations of SAF has been done by B&O, and SAF is now adopted into B&O’s design processes.

The creation of SAF starts with defining the needs for modelling streaming architectures. The needs are derived from the requirement model. The needs represent architectures elements we must understand in order to support valid decision-making and informal reasoning regarding different AV architectures. CAFF realization these needs by defining that any valid CAFF instants must has an AF Context Viewpoint (AFCV). The ACV instants represents the Architectural Framework Concerns (Needs) in Context. The Context Process are uses for creation of the AFCV. In Figure 80 SAF’s AFCV view is showed.

SAF’s AFCV defines following Architectural Framework Need(s) for modelling a streaming architecture:

- “model synchronization”
o Needs for understanding/communicating the synchronization models/protocols for enabled the desired emerged property synchronies sound defined in fix x.

• “models streaming roles”,
  o Needs for understanding/communicating the AV roles (renderer, source) concepts for give streaming architecture and their constraints on user level needs.

• “model communication”
  o Needs for understanding/communicating the AV roles protocols regarding “streaming of content” and “fault strategies”.

• “model product configuration”
  o Needs for understanding/communicating capabilities of product configuration sets for a giving CS’s architecture.

• “model streaming constraints”
  o Needs for give semantics meaning to the AF’s View Point(s).

For the SAF we have defined the needs for “...semantics meaning to the AF’s View Point(s)” (a specialization of DS_REQ_1). CAFF implements this need by defining that any valid CAFF instants must have an Ontology Definition View (ODV). In Figure 81 SAF’s ODV is displayed:

It’s beyond the scope of this document to provide a detailed description of SAF’s Ontology Element(s) and their relations. But just for illustrating how the ODV provide semantics meaning to the streaming domain, following phrases are derived from the ODV:

• One “Product Configuration” consists of two or more “Product”(s).
• One “Product” can be connected to one or more other “Product”(s).
• A “Product” takes a one or more “Role”, which can be of the type “Source Role” or “Renderer Role”.
• A “source Role” communicate with one or more “Renderer Role”(s) using a “Connection”.
• A “Connection” use one “Transport strategy”.
• "Content"(s) is transferred over an “Connection”(s) using an “Stream”(s).

Based on the ODV and the AFCV, a number of logical Viewpoints are now identified along with the relationships between them. This results in the Viewpoint Relationship View (VRV). SAF’s mapping of the needs to Ontology elements and their relations results in the VRV showed in Figure 82.

![Figure 82 SAF’s Viewport Relationship View](image)

As an example of the mapping: the CDV need sentence:
• "models streaming roles",
  o Needs for understanding/communicating the AV role concepts for a giving streaming architecture and their constraints on user level streaming Needs
and the ODV ontology sentence:
• A “Product” takes a one or more “Role”, which can be of the type “Source Role” or “Renderer Role” results in the “Streaming Role Viewpoint” on SAF’s VRV. The realization of the “Streaming Role Viewpoint” must provide the Viewpoint modelling design template for the product of the CDV and ODV sentences. The Viewpoints and their relations defined in the VRV Indirectly defines the needs for structural and behavioural realizations of the Viewpoints. The Viewpoints and their relations also indicate that Viewpoints can be behavioural or structural refinements of other Viewpoints in the AF. In SAF’s VRV the relations between the “Streaming Role viewpoint” and the “Streaming communication Viewpoint” states: the “Streaming communication Viewpoint” shows roles communication flow for the
roles defined in the “Streaming Role viewpoint”. The relation represent a modelling requirement saying the “Streaming communication Viewpoint” must be a behavioural refinement of the “Streaming Role viewpoint”. Hence “Streaming Role viewpoint” must be a structural Viewpoint. The VRV is product of CDV and ODV elements, defining a set of needed Viewpoints for describing the structural and behavioural aspect for a domain specific architecture.

Each Viewpoint of SAF are defined by a Viewpoint Context View (VCV), and a Viewpoint Definition View (VDV). The VCVs are Uses Cases(s) from the ACV and the VDVs are the semantics for the VCVs using ODV’s elements. This Viewpoint pair ensure needed traceability and Viewpoint instants consistent (satisfy DS_REQ_1 and DS_REQ_2). In Figure 83 the VCV for the “Product Configuration Viewpoint” are displayed

![Figure 83 Product Configuration VCV](image)

The second element of the “Product Configuration Viewpoint” pair is shown in Figure 84.
The VDV for the “Product Configuration Viewpoint” contain the ODV elements stating the relationships among products. The VDV also defines how are product configuration must be modelled. A “product Configuration” must be modelling as a packages and the relations between products must be “modelled with an association”. The VDVs gives a stronger semantics to the view instants, which enabled validation of correct models and make viewpoint instances comparable.

B&O uses a combination of COMPASS enabling patterns and frameworks for SAF’s viewpoints. These architectural pattern languages B&O has used are well known for component base designs and can easily been re-used at CS/SoS level modelling. COMPASS supports this by providing architectural level design patterns and frameworks tailored for CS and SoS modelling.

SAF are based on following COMPASS enabled patterns and frameworks
- The Integration Framework
- The Fault Modelling Architectural Framework
- The Contract Pattern

Since the Viewpoint’s VDVs are already defined for the COMPASS patterns and framework themselves. This section will only show some SAF viewpoint instantiations. For this purpose, the B&O proprietary Streaming framework NMM is used. NMM are fully modelled using SAF but only simplified NMM view instants will be showed in this document.

The SAF Viewpoint Streaming Communication Viewpoint (SCV) are based on two viewpoints from the Integration framework. The structural Interface Connectivity view (ICV) and a behavioural refinement of the ICV view. The behavioural refinement is based on the Interface protocol View (IPV). In Figure 85 the NMM SCV ICV instants is shown in Figure 85.
The IPV behavioural refinement of the NMM Source Role Source graph is shown in Figure 86

The IPV in Figure 86 models the parallelism of NMM’s SourceGraph implementation of the ARQ streaming protocol. The ARQ interfaces and the connection relation between the streaming roles defined in the protocol are modelled in the ICV (Figure 85). The ARO data types and interfaces are defined using the Interface definition View (IDV) from the Integration pattern.

The Fault Modelling Architectural Framework (FMAF) is used for defining the Communication fault tolerance Viewpoint of SAF. The viewpoints from FMAF provides the needed views for reasoning about different streaming architectures.
fault strategies/protocols and provide the basic for designing integration wrappers regarding fault handling for B&O. FMAF view instants also act as the foundation for designing fault tolerance testing strategies. Developing fault tolerance testing strategies and SAF’s use of FMAF are described in D42.2.

The Contract pattern is uses as formal refinement step for the semi-formal SysML bases SAF Viewpoint. It means that some of SAF viewpoints can be modelled using Viewpoints from the Contract pattern. Then to use Contract viewpoint depends on the analysis and communication needs for a giving streaming architecture. The base SAF viewpoints are based on the interfaces patterns views, and so it the Contract patterns viewpoints. This fact simplifies identification of refinement points between the base viewpoints and the specialization viewpoints. Please note in an architectural design process, there are needs for different levels of abstractions for enabling “Consistent SoS architectural level communication, development, and decision making among different development disciplines” (DS_REQ_1). The use of the Contract pattern as a semi-formal to formal refinement step is describe later in this document.

SAF provide needed viewpoints for enabled consistent AV domain architectural modelling. Nevertheless, the AF must be uses correct otherwise; partly design models will break the benefits of the AF. CAFF provide the Rules Definition Viewpoint (RDV) for handling addressing this problem. The RDV defines the rules that constrain the uses of the Architectural Framework and thereby increase the semantics strength of the AF instants. In Figure 87 SAF’s RDV is shown (simplified).

The SAF_RULE(s) in the RDV defines which view there must in a valid SAF model instant and are enforcing the needed relations among views. The SAF03 Rule states that for each SRV where must be a least one SFV. This enforces the needed architectural model views for satisfied the communication needs required understanding all AV role’s fault protocols.
The used and development of the domain specific architectural framework SAF meets B&O’s development process needs regarding:

- **DS_REQ_1**: Consistent SoS architectural level communication, development, and decision making among different development disciplines
- **DS_REQ_2**: Capability of expressing and analyzing SoS models during early design stages

The development process ontology in Figure 69 states the relationships between the “Requirement Model”, “Architectural Design Model” and the “Architectural Design Test Model”(s) as following:

- The “Architectural Design Test Model”(s) which are refinements of the “Requirement Model”, validates the “Architectural Design Model”.

The relation are also representing by the DS_REQ_3 and DS_REQ_4 requirements stating:

- **DS_REQ_4**: Capability of validation and testing of conceptual design models
- **DS_REQ_3**: Capability of identified areas of incompleteness or ambiguity in informal system specifications during early design stages

For the conceptual development stage, we define validation as being non-formal design model validation of requirements captured in the requirement model. In the MBE process, B&O uses Viewpoints from the Test Pattern for realization of Architectural Design Test Model(s). The Test pattern views instants are capable of validating the SAF based design models against the requirement model since the Test Pattern views are *vertical* requirement model refinements.

The Test Pattern Viewpoints uses in the process has following descriptions:

- The ‘Test Case Viewpoint’ (TCV) that identifies the scope of the Test Case and the anticipated behaviour of the Test Case using the ‘Test Behaviour Viewpoint’

The refinement relation from the SoS-ARCE views to Test Case views are simple translations of the requirement model’s VVs and the IVVs to Test Behaviour Viewpoint’s (a sub viewpoint of the TCV). Recall that VVs and IVV demonstrated that a CS or SoS requirement are testable and consistent, so VVs/IVVs already implicit represent Test Case(s), that can validate requirement ambiguity and incompleteness (stated in DS_REQ_3 and DS_REQ_4).

In B&O’s development process the requirement model’s behavioural VVs and IVVs are modelled using SysML *Sequence diagrams* and the behavioural views (IPVs) in SAF design models are modelled with SysML *State machine diagrams*. This mismatch in behavioural modelling notations raises problems. B&O desired to validate the design models using the COMPASS tool RT-Tester [D33.2] for enabling Design-Model-In-The-Loop-Testing and transition relation auto-test case generations. The B&O MBSE process defines horizontal refinement methods for translation of VVs and IVVs into contract based *State Machine diagrams*.
Figure 88 below shows the Test Pattern TCV instants for the streaming contract defined in the VV diagram in Figure 78.

The Architectural Design Test Model(s) represent the Needs in form of Test Cases, therefore the Architectural Design Test Model can validate the Architectural Design Model, which are, until validated, just a design proposition for the Needs.

So far we have described how the COMPASS technology based development process enabled the production of the Need, Requirement Model, Architectural Design Model and Architectural Test Design Model elements. COMPASS technology has meet all B&O’s needs for producing the models in the conceptual stage of the development model. The development ontology descriptions for the development model Figure 69 only describes the system (the blue elements) and relationships. We can now justify a full description of the relationships for all the ontology elements for the Conceptual Stage.
The “Conceptual” stage contains the identification/understanding of “Need”(s).
- The "Needs"(s) are captured in the “Requirement Model”
- The "Requirement Model" is produced using the “SoS-ACRE Requirement Framework”
- The "Requirement Model" drives the "Architectural Design Model"
- The "Architectural Design Model" is a refinement of the "Requirement Model"
- The "Architectural Design Test Model"(s) are refinements of the "Requirement Model".
- The "Architectural Design Test Model"(s) validates the "Architectural Design Model"
- The "Architectural Design Test Model"(s) are produced using the “Test Pattern”

The conceptual stage is about insuring building the right system or understanding a system is right according to the Needs. However, The MBSE process must also insure/support the verification concept stating: Have we built the system right? Alternatively, can we verify a system is built right?

The next section describes the formal stage of the COMPASS technology based development process, and how the formal stage must meets the verification needs.
6.6. Formal Modelling

This section will describe how formal COMPASS technologies are applying for realization of the formal stage of the development process show in Figure 69. The development models ontology elements relationships regarding refinements from the conceptual stage to the formal stage and the formal stage states:

- The “Formal” stage contains one or more “Architectural Design Specification Model”(s), which are drive by the “Architectural Design Model”. “Architectural Design Specification Model”(s) can be a “Model Checker Model” or a “Theory Proving Model”. The “Architectural Design Specification Test Model” verifies the “Architectural Design Specification Model”(s).

The term verification in the formal stage means preforming formal proofs (the transition from a design model to a proven design model) and the terms proven model and verified model means: A model is only verified or proven, if and only if there exits one or more mathematic proofs that the models holds for some properties $P$. We define a proven model to be an Architectural Design Specification Model. Since a Architectural Design Specification Model holds the mathematic truth for some specification $s$ embedded in the model.

For an industry, point of view it important to be able of benefiting from the formal models in the Real world. Meaning taking the truth from the formal models are using the truth in the real SoS/CS environment. The industrial formal stage motivations are formulated as:

“La mano kiu tenas la veron, povas puni la pekulon”
“The hand which holds the truth can punish the sinner”

The formal stage motivations contain following concepts and tasks

- **Find the truth**
  - Means verification of design models and includes following tasks
    - refinement from Architectural Design Specification Model(s) to Architectural Design Specification Model(s)
    - Verifying the refinement from Architectural Design Model(s) to Architectural Design Specification Model(s)
    - Verifying the Architectural Design Specification Model for some properties $P$

- **Use the truth (punish the sinner)**
  - Architectural Design Specification Model holds the truth. Meaning traces from specifications of the Architectural Design Specification Model can be used to verify, analyse or drive and CS/SOS implementation (which might be the sinner) of the Architectural Design Specification Model in the Real stage. The tasks and realization of the use the truth concepts are described in the Real stage section of this document.
So one of the main purposes of the formal stage is about *design model verification*. Architectural Design Model(s) are only validated against the Requirement Model at a conceptual level. Architectural Design Model(s) might not hold the *truth* according to properties implicit defined in the requirements at a more technically level. Revisiting two of the design stages needs from B&O, which state:

- **DS_REQ_4**: Capability of validation and testing of conceptual design models
- **DS_REQ_3**: Capability of identified areas of incompleteness or ambiguity in informal system specifications during early design stages

We redefine these needs to formal stage needs based on the formal stage definitions and the formal stage motivations

- **FS_REQ_1**: Capability of refinement from *conceptual design models* to *formal specification models*
- **FS_REQ_2**: Capability of testing *formal specification models*
- **FS_REQ_3**: Verification capability regarding identified areas of incompleteness or ambiguity in informal system specifications (conceptual design models) during early design stages

COMPASS provides several solutions for addressing the *design model to design specification refinement model* needs described in **FS_REQ_1**. As briefly mention in the conceptual stage descriptions. The AV domain specific architectural SAF use the Contact Pattern as first step in the translation of the Architectural Design Model to Architectural Design Specification Models. The Contract Pattern is intended be used as an aid for formal definitions of contracts. The contract definitions form the base for further verification. The implicit contracts of the SAF Viewpoints instances are been giving a stronger semantics strength by explicit contract definitions using refinement to Contract patterns views. Some examples of the horizontal viewpoint refinements points between SAF views and Contract Patterns (CP) views are:

The SAF Viewpoint Streaming Communication Viewpoint (SCV) are refined to the CP’s *Contract Definition Viewpoint (CDV)* stating:

- The ‘Contract Definition Viewpoint’ is used for the definition of Contracts in terms of their state variables, state invariants and operations they may provide.

B&O’s AV leadership CDV are shown in Figure 89
The SAF Viewpoint Product Configuration Viewpoint (PCV) are refined to the CP Contractual SoS Definition Viewpoint (CSDV) which state:

- The 'Contractual SoS Definition Viewpoint' is used for the identification of Contracts comprising the Contractual SoS.

Recap the SAF’s PCV defines a SoS there each product represents an CS with some capabilities. SoS-level Needs require a set of CS capabilities. The Need capabilities are equal to a set of needed CS contracts. By modelling the CS/SoS Contracts B&O can both design needed CS Contracts for an implementation, and verify CS Contract conformity of an implementation. B&O’s SoS Integration problem requires designing wrapper solution for integration of non-B&O products. Wrapper designs required understanding and verification of needed contacts and the CSs invariants of the contracts. In Figure 91 shown CDDV instants for the device roles in the AV SoS.
The Contract Patterns Viewpoints are based on the Interface Patterns Viewpoints, as SAF are through the Integration Framework. This relation simplifies the viewpoint to viewpoints refinements. Correct CP viewpoints instants contain the state and behavioural information needed for verification. CP views hereby meet the design model to design specification model refinement needs defined in FS_REQ_1. D22.6 contains a full example of the Contract Pattern applying to the B&O Leadership case study aspect.

It is worth mentioning that CP viewpoints could be directly modelled in a formal language such as CML. The benefits of modelling contracts using a semi-formal domain specific framework are that contract models preserve the needed cross-domain stakeholder communication, and produce formal languages independent contract specifications.

The elementary product of the CP views defines B&O’s domain specific formal languages semantics needs. CML formal arguments must hold for these needed expression capabilities for formal modelling concepts as pre-and post-condition(s), state(s), process(s), time, communication, operator(s), function(s), operation(s) and invariant(s). How the UTP-based formal arguments for CML semantics holds for these needs is describe in D42.2. In this document, we let the CML models act as evidence for CML’s formal arguments.

The formal development stage Need FS_REQ_3 states:

- FS_REQ_3: Verification capability regarding identified areas of incompleteness or ambiguity in informal system specifications (conceptual design models) during early design stages

Architectural Design Specification Models are produced based on contract verification needs. Recall that a specification model $M$ must holds for some
property $P$. The Contracts for $M$ define $P$, and $P$ defines the needed verification of $M$, so $M$ holds for $P$. The Contracts for $M$ also indicate the modelling style and tools for $M$ for enabled verification of $P$. COMPASS provides several tools and guidelines for meeting verification needs. The Development process ontology contains three main Architectural Design Specification Models with following verification needs meaning and relations:

- **The Architectural Design Specification Model (ASPM)**
  - The base ASPM are uses for simulation of M’s completeness of $P$. The verification of $M$ are enable by the Symphony interpreter/debugger plug-ins and CML’s formal arguments.

- **The Model Checker Model (MCM)**
  - The MCM is a specialization of the ASPM model and extend $M$’s verification of $P$ with model-checker verification of $P$. The verification of $M$ are enable by the Symphony Formula plug-in and CML’s formal model-checker arguments.

- **The Theory Proving Model (TPM)**
  - The TPM is a specialization of the ASPM model and extend $M$’s verification of $P$ with theory proving verification of $P$. The verification of $M$ are enable by the Symphony Isabell/proof obligations plug-ins and CML formal theory proving arguments.

Often a set of different Architectural Design Specification models is needed for complete verification of $P$. As an example of how COMPASS technology can enable modelling Architectural Design Specifications this section will show the TPM and MCM CML models for the CS SourceGraph contracts. The high-level contract semantics of the Source graph interface state:

- **SG_Contract_1**: “A valid interface implementation must always reply on a request”.
  - The protocol is a request-reply protocol, aimed at handling distributed state consistency between source and control layer CSs in the AV SoS.

- **SG_Contract_2**: “If a state transition fails, a valid interface implementation stays in the current state”.
  - Defines the error semantics of the protocol for ensuring distributed state consistency.

- **SG_Contract_3**: “The specifications transition system enables distributed CS states consistency regarding streaming states”
  - Defines the protocol’s semantics must be complete and not have ambiguity.

The SourceGraph contracts defined following verification needs (the set $P$ for $M$).

- Verification of the transition relation of the protocol.
This need includes live lock freedom, deadlock freedom and non-deterministic verification (analyses for incompleteness or ambiguity)

- Verification of specification consistency:
  - This need include discarding specification proof obligations for ensuring specification completeness and specification consistency.

The CML belows shows relevant elements of the CML Sourcegraph TPM model addressing verification of \( P \) for SG_Contract_1 and SG_Contract_2.

```cml
actions
  act_mainEntryPointState = actGenericTransitionsLoop /\ \ ch_init->Skip
  actGenericTransitionsLoop =
    mu X@
      (dcl s:StreamingLayerReplyEvent @ ch_genericEvent?e->( s :=
        convertEventToState(e) ));
        if isLegalTransition(s)
          then (StreamingTransition(s); ch_streamingReply!(currentState)->X)
          else (ch_streamingLayerError.<STATE_ERROR->)->X)
    )
  @ch_init ->initObject(); act_mainEntryPointState
```

The CML process’s actions models following protocol semantics. The CS process synchronized on the event channel "ch_genericEvent" representing incoming transition events. The process validated the transition events against the protocol transition relation as a pair of current state and the current received event. If the events are valid, the process reply on the “ch_streamingReply” channel with the state is has transited to. If the events are not valid according to the transition relation, the process reply on the “ch_streamingLayerError” channel and remind in its current state. This part of the model aim as the specification for request-reply and error protocol communication semantics. For ensuring completeness and specification consistency the TPM model explicit defined \( P \) as operations with pre- and post condition and states with invariants. The operations and states relevant for verification of the protocol completeness of \( P \) are shown below.

```cml
state
  transitionMap: map StreamingLayerReplyEvent to TransitionSet :=
    { cid |-> {} | cid in set transitionEventSet }
  inv dom transitionMap = transitionEventSet and dom transitionMap <> {}

  currentState:StreamingLayerReplyEvent := <DISCONNECTED>
  inv currentState in set transitionEventSet
```

129
operations

initObject():() => ()

initObject() ==

(transitionMap(<CONNECTED>) := {<DISCONNECTED>};
transitionMap(<DISCONNECTED>) :=
{<CONNECTED>, <STREAM_READY>, <PLAYING>, <STOPPED>};
transitionMap(<STREAM_READY>) := {<CONNECTED>, <STREAM_READY>, <STOPPED>};
transitionMap(<PLAYING>) := {<STREAM_READY>};
transitionMap(<STOPPED>) := {<PLAYING>}
)

isLegalTransition: StreamingLayerReplyEvent => bool

isLegalTransition(evt) ==

( return evt in set transitionEventSet and
  card {x | x in set dom transitionMap @ evt = x and currentState in set transitionMap(x)} > 0
)

StreamingTransition: StreamingLayerReplyEvent => ()

StreamingTransition(evt) ==

( currentState := evt
)

pre evt in set transitionEventSet and
  card {x | x in set dom transitionMap @ evt = x and
  currentState in set transitionMap(x)} > 0

post currentState = evt

The verification of $P$ is carried out by running the Symphony Isabell/proof obligations plug-ins. The proof obligations for $P$ are generated and auto-
discarded by Isabelle. The result is a proven design specification for the properties in the set $P$. A full description of the verification of The TPM model using the Symphony Isabell/proof obligations plug-ins can be found in D42.2

TPM models are mostly bases on CML’s VDM state and operation semantics, and cannot generate proof for the CSP semantics of CML. TPM models can therefore not verify properties as live lock freedom, deadlock freedom and non-
determinism. These properties are present regarding verification of the transition relation of the Sourcegraph protocol as stated in SG_Contract_3. Hence, the specification completeness of the TPM SourceGraph model depends on the completeness of the protocol transition relation. For completeness verification of $P$ for the SourceGraph protocol contracts an MVM CML model is produces. Below the main protocol process for the SourceGraph MCM CML model is showed:
actions

-- action for starting the main product statemachine, can be interrupted by a sync on the ch_init channel
act_mainEntryPointState = act_processLoop /\ ch_init->Skip

-- main product state machine action. Offer events based on the current state of the product
act_processLoop =

\(\text{[currentState } \neq \text{<DISCONNECTED>} \)\& act_disconnect -- most always offer to disconnect after the product is connected
\([]\)
\(\text{[currentState } = \text{<CONNECTED>} \text{ or currentState } = \text{<STOPPED>}\)\& act_connected -- if the product is not playing or is connected, it offer prepare a song
\([]\)
\(\text{[currentState } = \text{<DISCONNECTED>} \)\& act_waitOnConnect -- if the product is disconnected offer to connect
\([]\)
\(\text{[currentState } = \text{<STREAM_READY>} \)\& act_readyToPlayState -- if we have prepared a song offer to play it
\([]\)
\(\text{[currentState } = \text{<PLAYING>} \)\& act_playingState -- if the product is playing a song, offer to stop it
\); act_processLoop

The protocol transition relation verification of \(P\) are carried out by running the Symphony Formula plug-in on the SourceGraph CS MCM model. \(P\) is satisfied by the SourceGraph MCM model, thereby we had another proven design specification for a subset of the properties \(P\). \(P\) is fully satisfied by the union of the SourceGraph TPM and MCM CML models. So COMPASS technology meets the development process needs defined FS_REQ_3.

The formal stage FS_REQ_2 Need states: “Capability of testing formal specification models”. FS_REQ_2 represent the development model’s ontology elements relationships: Architectural Design Specification Test Models verifies the Architectural Design Specification Models. FS_REQ_2 is also derived from the formal “find the truth” motivation stating: Verifying the refinement from Architectural Design Models to Architectural Design Specification Models. Recall the Architectural Design Models has been validation against the Requirement Model using the Architectural Design Test Model. Architectural Design Specification Models must also be validated against the Requirement Models ensuring the refinement has preserved the validity from the Architectural Design Model. The Architectural Design Specification Test Model are a refinement of the Architectural Design Test Model tailored for requirement validation and verification of the Architectural Design Specification Model instants. Once again, the COMPASS Test pattern is used for development of Architectural Design
Specification Test Models. The Test Pattern address development of a full V&V campaign for a giving development process. The Architectural Design Test Models are only using the validation concepts from the Test Pattern, there Architectural Design Specification Test Models represent the verification concepts from the Test Pattern.

The refinement points between the Test Pattern views depend on the semantics of the Architectural Design Specification Model. Architectural Design Specification Test Model are development based on the semantics of both the Architectural Design Specification Model and Architectural Design Test Model. In Figure 92 the Test Pattern Test Configuration View (TCV) instants for the SourceGraph CS, protocol:

![Figure 92 SourceGraph CS Test Configuration View](image)

The TCV view contains the streaming interface (channels and events) definition from the CML TPM model explained before. The TBV for the Test Environment contain the SourceGraph protocol transition relation and communicating semantics for verification of the CML TPM model’s specification. Recall ADSTM are partly refined from the ADTM and ADTMs are refined from the VVS views from the RM. The proven RM->ADTM->ADSTM refinement/traceability link justify that ADSTM can validate the ADSM regarding requirements define in the RM, and thereby meet our formal motivation regarding verifying the ADTM and the ADSTM refinement points.

For realization of the “Capability of testing formal specification models” formal stage need. COMPASS provides testing methods- and operational solutions for these Design-model-in-the-loop testing concepts. The RT-Tester tool provides auto-test case generations and test case execution based on ADSTM SysML models. The formal arguments: $T(ADSTM) \subseteq T(ADSM)$ from COMPASS’s design-model-in-the-loop testing methodology simplify state:

“The transition system of a valid ADSM model is equal and consistent with the transition system of the ADSTM”
RT-Tester generate from the ADSTM the full transition systems traces set $T$ and verifying that for all $t$ of $T$ the CML ADSM model has a $t'$ there $t' = t$ is true (trace equivalence $= \forall t: T, \exists t': T' \cdot (t = t')$). Note the trace sets of ADSTM and ADSM models can be timed traces: meaning $TT(ADSTM) \subseteq TT(ADSM)$ must hold too. RT-Tester can verify that the timed trace equivalence holds for too.

COMPASS’s Test Pattern, formal testing arguments and the RT-Tester/Symphony tools meets the development process needs define in FS_REQ_2. More details about the COMPASS testing tools and methodology uses for realization of the formal models testing needs can found in D42.2

COMPASS technology meets all B&O’s needs for producing the models in the Formal stage of the development model and hereby fulfils the formal motivation regarding finding the truth. We can therefor now justify a full description of the relationships for all the ontology elements in the Formal Stage:

- The “Formal” stage contains one or more “Architectural Design Specification Model”(s), which are refinements of the “Architectural Design Model”(s).
- The viewpoint refinement from the “Architectural Design Specification Model”(s) to “Architectural Design Model” are produced via “Contract Pattern”.
- The Contract pattern views define the Contract(s) to be specified by the “Architectural Design Specification Model”(s).
- “Architectural Design Specification Model”(s) can be a “Model Checker Model”(s) or a “Theory Proving Model”(s).
- “Tool” verifies “Architectural Design Specification Model”(s)
The "Architectural Design Specification Test Model" validates and verifies the "Architectural Design Specification Model" (s).


There is one Formal Stage ontology relations with has not been cover in this section. It’s the relation saying: “Architectural Design Specification Test Model”s be created from elements (traces, proofs, contracts ...) of a proven “Architectural Design Specification Model”. This relation are described in the Real section since the close relation to verification of real CS implementations concepts and creating test modes from proven model traces.

The next section describes the Real stage of the development process and COMPASS technology applicability for the Real stage needs.

### 6.7. Co-Simulation and Testing modelling

The Real stages ontology elements and relationships in Figure 69 are saying: The “Real” stage contains B&O “Product”(s), which are specializations of the COMPASS ontology concept “System”. “Product”(s) consist of “software System”(s) and “hardware System”(s). “Product”(s) can be generated from an “Architectural Design Specification Model” or be analysed by an “Architectural Design Specification Model”. “Product”(s) are verified by “Architectural Design Specification Test Model”(s). The formal motivation concepts Use the truth are implicit stated in these ontology relations. COMPASS technology must enable specification verification and specification refinement between the proven models and the real CSs. The requirements for the specification verification and specification refinement are:

- RS_REQ_1: Capabilities of using a formal specification for driving the development of a new CS/SoS
- RS_REQ_2: Capabilities of verify a CS/SoS based on a formal specification

COMPASS addresses RS_REQ_1 with the COMPASS Refinement Framework. The Architectural Design Specification Model $M$ contains the truth for all properties $P$ for the contracts of $M$. The contracts elements and the semantics of $M$ defines Refinable elements and Refinement Points. The semantic translation function between the CS/SoS implementation languages and the domain specific CML model are produces using the refinement rule concept from the Refinement Framework. B&O has AV domain specific C/C++ frameworks, which are drive by the same AV domain concepts as the Architectural Design Model and the Architectural Design Specification Model models. Having the same domain semantic foundations for modelling and implementation ease the development of the translation functions and add support for auto-code generation. Note that the refinement point can be formal refinement laws proving that the refinement point translation holds. In the code below, parts of the C++ SourceGraph interface are showed.
// asynchronous interface. Source CSs must implement this interface
class ISourceGraphListener
{
    public:
        // C++ enum representing the state of the source graph,
        enum CallBackEvent{CONNECTED,PLAYING,STREAM_READY,STOPPED,ERROR_STATE,DISCONNECTED};

        virtual ~ISourceGraphListener(){}  
        // the callback function, streaming clients will be notified regarding the
        status of transitions from fan invoked function call
        virtual void streamingEvent (const CallBackEvent& event)=0;
};

// the source graph interface (renamed to fit the CML model)
// the call semantics are asynchronous, all function call will return
// immediately, the result of the state transition will be delegate to the
// callback client
// this source graph only support one renderer, meaning its a PLAY product
source graph
class ISourceGraph
{
    public:
        virtual ~ISourceGraph(){}

        virtual void setCallbackClient(boost::shared_ptr<IStreamingInterfaces::ISourceGraphListener> client) = 0;

        //! @brief Connect the source graph to a sink graph.
        //! This is a asynchronous method, invoking it will always result in the
        //! Streaming Event callback method on the registered ISourceGraphListener
        //! being invoked With either CONNECTED or ERROR_STATE.
        virtual void connect() = 0;

        //! @brief Disconnect a previously established connection.
        //! this is asynchronous method invoking it will always result in the
        //! streamingevent callback method on the registered ISourceGraphListener
        //! being invoked With either DISCONNECTED or ERROR_STATE
        virtual void disconnect ()= 0;
    ...

The C++ SourceGraph class and functions interface definitions are the results of
the semantics translation from the SG_Contract_1 and SG_Contract_2 contract
specifications from the TPM CML models described earlier in this document. The
Refinement Framework concepts enables development of semantics translation
function for formal specification models to CS domain specific languages, hereby
the development process needs stated in RS_REQ_1 are meets COMPASS
technology.

The Real stage development Need RS_REQ_2 states: “Capabilities of verify a
CS/SoS based on a formal specification”. The specification models hold the truth,
so all traces of the CS implementation must be the same as the traces from the
specification. COMPASS technology must provide an operational solution for verifying that for all $t$ of the CML ADSM model transitions system, there is a $t'$ of the CS's transition system there $t'=t \text{ is true}$ (trace equivalence $= \forall t : T, \exists t' : T \cdot (t = t')$). The trace sets of ADSM can be timed traces: meaning $TT(ADSM) \subseteq TT(CS)$ must hold too.

The Development process ontology relationships drives this need are the combination of the Formal stage relations stating: "Architectural Design Specification Test Models can be created from elements (traces, proofs, contracts ...) of a proven Architectural Design Specification Model" and the Real state relation stating: "Product(s) are verified by Architectural Design Specification Test Model(s)". This implies that COMPASS must support construction of Architectural Design Specification Test Models from the traces of an Architectural Design Specification Model.

The COMPASS symphony interpreter tool provides the feature of generating CML traces from CML models. The tool process and theory for meeting the trace equivalence needs are very simple and can be formulated as: since the specification holds the truth, we can ask the model anything (event inputs, operations calls,...). The model’s answer will always be the right answer (traces outputs). The output traces are translated into CML CS/SoS Test Case models. The Test Case models must then be formally refined by the real CS’s software-and hardware systems. Below there is a Sourcegraph CS CML Test Case model, generated from the SourceGraph TPM CML model’s traces.

```
process TestTraces2 =
begin
  actions
    act_connect = ch_genericEvent.(<CONNECT>)-
    >ch_streamingReply.<<CONNECTED>->act_error_connect
    act_error_connect = ch_genericEvent.(<CONNECT>)-
    >ch_streamingLayerError.<<STATE_ERROR>->act_setstream -- deadlock if we
dont get an error
    act_setstream = ch_genericEvent.(<<SETUPSTREAM>)-
    >ch_streamingReply.<<STREAM_READY>->act_play
    act_play = ch_genericEvent.(<<PLAY>>)-ch_streamingReply.<<PLAYING>>-
    >act_stop
    act_stop = ch_genericEvent.<<STOP>>-ch_streamingReply.<<STOPPED>>-
    >act_disconnect
    act_disconnect = ch_genericEvent.<<DISCONNECT>>-
    >ch_streamingReply.<<DISCONNECTED>>-ch_init->Skip
  @ ch_init->act_connect
end
//Test SoS process
process TestTPCS2 = TestTraces2
[||{ch_init,ch_genericEvent,ch_streamingReply,ch_streamingLayerError}||]Str
eamingPlayerTPCSProcess
```
In the TestTPCS2 TestCase model above the SoS level process are defined. The SoS level process contains the TestCase and the SpurceGraph CS processes as a parallel composition synchronized on the streaming SoS channel alphabet. The SoS level process structure enables of us using the COMPASS co-simulation features for the final realization of the RS_REQ_2 need.

The CML SoS models are proven, meaning that it is formally verified that all CS processes in the model meets their contracts, and thereby the SoS desired properties are meet by the specification. So swapping one or more CS processes out with real CS processes helps us validation the real CSs contracts compliance. This approach helps not just with verification of white box CSs but also regarding contract conformance of uncontrollable grey-and blackbox CSs. Analysis the real CS contract semantics helps also in verification of contract wrapper designs. The co-simulation tool and underlying trace equivalence theory is powerful in addressing B&O SoS integration problems. Below there are some C++ code factions showing the CML to C++ semantics Co-Simulation framework mapping for enabled TestCase verification of a C++ SourceGraph CS implementation.

```cpp
//callback class, will be called by the co simulation framework
class GenericStreamingCS: public CoSimulationFramework::ACoSimulationCallback<>
{
  public:
  // inspect function will return the set of events the CS can currently sync on
  CoSimulationTransportLayer::IChannelEventObject::ChannelEventObjectSet inspect()
  {
    CoSimulationTransportLayer::IChannelEventObject::ChannelEventObjectSet eventOptions;
    if ( ! cb ) //== @ch_init ->initObject()
    {
      eventOptions.push_back(createSyncEventObject("ch_init"));
      return eventOptions;
    }
    if ( replyState)
    {
      // ch_streamingReply!(currentState) offer a reply event on the
      // ch_streamingReply channel
      // or ch_streamingLayerError if return value is ERROR_STATE
      eventOptions.push_back(createWriteSyncOnEventObject<std::string>(channelName, CoSimulationFramework::ChannelOperation::SYNC_ON, ConvertStreamingEnumToString(cb->getResult())));
    }
    realStreamingState(eventOptions); ...}
```

In the TestTPCS2 SoS level process definition, the StreamingPlayerTPCSProcess are replaced with real CS C++ implementation in a Co-Simulation configuration.
Then executing the SoS process from the tool. The real CS must offer the same events to the environment as the CML process for enabled process synchronization between the CS processes in the SoS. If not the tool will deadlock the simulation and we can denote the real CS as the sinner. An successful Co-Simulation execution termination proven that the real CS are trace equivalence with the CML specification model; meaning the real CS is not an sinner, since he tells the same truth as the proven formal model does then asked the same question. The capabilities of verification of trace equivalence between proven models and real CS leads to conclude that the RS_REQ_2 development process needs are meet by COMPASS technology. In D42.2, the reader can find more details about B&O’s uses of the COMPASS Co-simulation framework.

Revisiting the testing needs defined in the beginning of the Case study application section. The testing stage needs states:

- SoS Testing Stage Requirements
  - TS_REQ_1: Capabilities of identified needed SoS test cases for different AV product configurations
  - TS_REG_2: Capabilities of testing SoS properties in different AV product configurations

Throughout the COMPASS base development process, B&O has applied COMPASS testing methods and tools like the Test Pattern, RT-Tester and Co-Simulation tools. So these needs are already being met. The TS_REQ_1 is met by the identification of test case based on the traces of a proven specification model. RT_Tester release this need further by providing auto-testcase generation based on the traces of the specification model and requirement based design test model.

The Test pattern provides the views for modelling a full V&V campaign for all stages of a development process. RT_Tester and other COMPASS tools provide the operational realization of the V&V camping modelled in the Test Pattern views. For realization of the TS_REQ_2 need B&O has successful uses COMPASS testing tools and methodologies for CS/SoS software-in-the-loop and CS/SoS hardware-in-the-loop testing. More details about COMPASS testing capabilities and B&O has uses of the tools and methods can found in D42.2.

COMPASS technology meets all of B&O’s needs for producing the models in the Real Stage of the development model and hereby fulfilled the formal motivation regarding using the truth. We can therefore now give a full description of the ontology elements and their relationships in the Real Stage:
The “Real” stage contains the B&O “Product”, which is a specialization of the COMPASS ontology concept “System”.

- A “Product” consist of “software System”(s) and “hardware System”(s).
- “Software Systems”(s) are deployed on to “Hardware System”(s)
- A “Product” can be generates from contract specifications of an “Architectural Design Specification Model”
- A “Product” can be analyse by the traces from “Architectural Design Specification Model”.
- A “product” are been verified by the “Architectural Design Specification Test Model”.
- “Architectural Design Specification Test Model” can be created from the traces of an “Architectural Design Specification Model”

The Real stage was the last stage of the COMPASS technology based development process. The following section will conclude the overall achievements and further work.

6.8. Conclusion

The B&O case study section of this document has described an industrial model-based SoS/CS development process. The process is starting from needs to SOS/CS product deployment, with parallel testing, and validation and verification activities for the different development stages of the process. For each stage of the MBSE development process a set of COMPASS technologies has been applied. The results of applying COMPASS technology has been validated against the
development process needs. The needs for a given stage of the process was
derive from B&O’s SoS problem domain and organisational development
structure. The conclusion regarding COMPASS usefulness versus the SoS
development process needs is:

COMPASS model-based methodologies and tools have met B&O’s needs for
improvement of their development processes regarding SoS engineering.
Arguments can be raised against the industry maturity regarding some of the
formal tools, but the theoretical foundations are proven applicable for the
industry. The motivation for using the formal aspects of COMPASS technology in
an industrial context has been proven possible and realizable by COMPASS. The
semi-formal COMPASS technologies have already been adopted by B&O, there
the formal technologies are troubled by tool maturity and the requirement levels
of formal knowledge. The justification for using the formal tools and formal
argument are proven by the specification model results. The verification results
described could not have been archived using non-formal methods and tools.

One of the unique selling point for COMPASS are the linking/refinements
methods between the stages of the development process. The capabilities of
preserving and using achieved goals from and between stages are highly
decidable for the industry. An example of this is the capability of using the
results from the formal stage to validate the real stage implementations. Another
selling point of COMPASS is the ontology methodology. Correct uses of the
ontology methodology can enable consistent communication and reasoning
about complex SoS domains. The development of SoS level domain specific
semantics is heavy needed in the industry regarding SoS development. Often the
SoS’s CS stakeholders have their own CS subdomain terminology, which differs
from the SoS level semantics, breaking consistent communication among
stakeholders. For more details about B&O, uses of COMPASS technology and
benchmarking of COMPASS technology can be found in D42.2.
7. Conclusions

This report has described the overall COMPASS approach to SoSE, pulling together the various aspects of the approach as found in a number of other COMPASS deliverables.

The COMPASS approach consists of two key parts, the COMPASS Architectural Framework and the COMPASS Process Library. The COMPASS Architectural Framework consists of an Ontology and a number of Viewpoints. The COMPASS Process Library is composed of a number of Processes. Each Process addresses an aspect of SoSE and produces (and consumes) one or more Viewpoint (or, more correctly, one or more Views that conform to Viewpoints). Each Viewpoint of the COMPASS Architectural Framework is supported by zero or more Enabling Patterns. For information on Enabling Patterns, see [D22.6 2014]. Finally, the systems modelling language (SysML) and COMPASS modelling language (CML) can be used to realise the Enabling Patterns and the Viewpoints.

The various Ontologies that form part of the COMPASS Architectural Framework, covering Requirements, Processes, Competency, Architectures & Architectural Frameworks, Traceability, Integration and Refinement as presented in deliverables [D21.1 2012], [D21.2 2013], [D21.4 2013], [D22.5 2013] and [D22.6 2014] were brought together into a complete Ontology. This full Ontology is intended to form a useful source of information for anyone engaged in SoSE, showing how the various areas covered by COMPASS relate one to another. It also ensures the consistency of the various Frameworks covered by the COMPASS Guidelines and themselves brought together into a whole in this document. In a similar way, the various Frameworks developed from the Ontologies were brought together into a complete COMPASS Architectural Framework. The Processes and Guidelines that form the COMPASS Process Library were summarised and a number of scenarios presented that illustrate the Processes and Guidelines that can be used to accomplish a number of SoSE tasks.

In order to successfully carry out SoSE, it is important to have people with the correct skills. This deliverable includes a Competency Framework and set of supporting Processes. A number of Competency Scopes were presented that cover the Stakeholder Roles involved in all of the COMPASS SoSE Processes defined to date. These Competency Scopes were based on the INCOSE Systems Engineering Competencies Framework.

The report summarises the ways in which the COMPASS approach to SoSE will be supported by the various tools: Atego Process Director for dissemination of Processes and AFs; Atego Modeler (formerly known as Artisan Studio) for realisation of Viewpoints and Enabling Patterns through SysML and generation of CML from SysML; and the CML Toolset for formal reasoning about models.

The report concludes with a description of the application of the approach to one of the COMPASS industrial case studies, that from B&O.
8. References


9. Appendix I – The COMPASS SoSE Ontology

The diagram and associated descriptions presented in this appendix are intended to provide a useful summary of the Ontology. For details behind the various concepts please see the relevant deliverable, as detailed in Section 12.

Figure 94 - The full COMPASS SoSE Ontology
<table>
<thead>
<tr>
<th>Ontology Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledged System of Systems</td>
<td>have recognized objectives, a designated manager, and resources for the System of Systems; however, the Constituent Systems retain their independent ownership, objectives, funding, as well as development and sustainment approaches. Changes in the systems are based on collaboration between the System of Systems and the system.</td>
</tr>
<tr>
<td>Activity</td>
<td>An Activity forms part of a Process and represents something that must be done to realise that Process. An Activity produces and consumes Artefacts and has a Stakeholder Role that is responsible for its execution. An Activity also uses one or more Resource.</td>
</tr>
<tr>
<td>Architectural Framework</td>
<td>Architectural Framework - a defined set of Viewpoints and an Ontology. The Architectural Framework is used to structure an Architecture from the point of view of a specific industry, stakeholder role set, or organisation. The Architectural Framework is defined so that it meets the needs defined by its Architectural Framework Concerns. An Architectural Framework is created so that it complies with zero or more Standards.</td>
</tr>
<tr>
<td>Architecture</td>
<td>Architectural Framework Concern - defines a Need that an Architectural Framework has to address.</td>
</tr>
<tr>
<td>Artefact</td>
<td>An Artefact - a description of a System, made up of a number of View. Related View can be collected together into Perspectives.</td>
</tr>
<tr>
<td>Capability</td>
<td>A Capability represents something that is produced or consumed by an Activity.</td>
</tr>
<tr>
<td>Collaborative</td>
<td>In Collaborative System of Systems, the component systems interact more or less voluntarily to fulfill agreed-upon central purposes. The Internet is a Collaborative system. The Internet Engineering Task Force works out standards but has no power to enforce them. The central players collectively decide how to provide or deny service, thereby providing some means of enforcing and maintaining standards.</td>
</tr>
<tr>
<td>Competency</td>
<td>Competence is the ability exhibited by a Person that is made up of a set of one or more individual Competencies.</td>
</tr>
<tr>
<td>Competency Area</td>
<td>Competency Area - a grouping of related Competency, such as those related to requirements engineering or to architectures.</td>
</tr>
<tr>
<td>Competency Profile</td>
<td>A Competency Profile - a description of a Person's abilities that are possessed by a specific Person. The Competency Profile may be generated at the output of a competency assessment exercise that uses a Competency Scope as its input.</td>
</tr>
<tr>
<td>Competency Scope</td>
<td>A Competency Scope defines the abilities that are required for a specific Stakeholder Role.</td>
</tr>
<tr>
<td>Constituent System</td>
<td>A Constituent System - a System in its own right that forms part of an overall System of Systems.</td>
</tr>
<tr>
<td>Context</td>
<td>The idea of the Context is fundamental to the requirements Framework described in this report. In its simplest form, a Context may be thought of as a ‘point of view’ on the system under development. It is possible to view the Needs of a system from any number of different points of view (contexts), so it is essential that it is well understood from what point of view each Context is taken.</td>
</tr>
<tr>
<td>Directed</td>
<td>Directed System of Systems - are those in which the integrated System of Systems is built and managed to fulfill specific purposes. It is centrally managed during long-term operation to continue to fulfill those purposes as well as any new ones the system owners might wish to address. The component systems maintain an ability to operate independently, but their normal operational mode is subordinated to the central managed purpose.</td>
</tr>
<tr>
<td>Element Type</td>
<td>Element Types - represent types of View Element occurring in a model. Element Types may represent underlying modelling element types from the modelling language being used, such as a block if using SysML or a class if using UML. They may also represent defined conceptual elements that are used on views from a Framework that is being used, such as a System Element that is used on a Validation View from the COMPASS SoS-AGRE Framework.</td>
</tr>
<tr>
<td>Flow Type</td>
<td>A Flow Type - defines the type of the things flowing across an Interface. For example, an Interface for a transmitter may have a ‘transmit’ operation that takes a power level parameter. The type of this parameter would be defined using a Flow Type. Similarly, the type of fluid pumped by a pump would be described by a Flow Type.</td>
</tr>
<tr>
<td>Flow-based Interface</td>
<td>Flow-based Interfaces - are used to represent those Interfaces that transfer data, material, energy, personnel etc. between System Elements. For example, an Interface between a fuel pump and an engine would be represented by a Flow-based Interface.</td>
</tr>
<tr>
<td>Formal Scenario</td>
<td>Formal Scenarios - are realised in SysML through parametric constraints and their usages, allowing a more mathematical-based approach to be taken for understanding the Use Cases. The parametric usages are connected together into different networks that allow ‘what if’ analysis and are particularly powerful when considering trade-offs. Formal Scenarios can be given a semantics using formal methods such as VDM or CSP. In COMPASS we use the Unifying Theories of Programming (UTP) as the underlying formalism in which to express both VDM and CSP. As a consequence, we can model state-based properties in VDM, and</td>
</tr>
<tr>
<td>Ontology Element</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Gate</td>
<td>A Gate is a special type of review that must be executed before any one Stage may be exited. A Gate assesses the execution of a Stage.</td>
</tr>
<tr>
<td>Goal</td>
<td>A Goal defines and describes a desired outcome of a System or System of Systems.</td>
</tr>
<tr>
<td>Indicator</td>
<td>A Indicator – a feature of a Competency that describes knowledge, skill or attitude required to meet the Competency. It is the Indicator that is assessed as part of competency assessment.</td>
</tr>
<tr>
<td>Interface</td>
<td>An Interface identifies and defines the boundary across which two System Elements meet and act on or communicate with each other. Interfaces are connected one to another via an Interface Connection and are described by Interface Definitions.</td>
</tr>
<tr>
<td>Level</td>
<td>Level describes the maturity of a Competency. There are four Levels defined for the COMPASS Ontology.</td>
</tr>
</tbody>
</table>
| Life Cycle       | The Life Cycle describes the evolution of a System over time. A single System may have any number of Life Cycles associated with it, depending on the context of the system. For example:  
- A product Life Cycle  
- A project Life Cycle  
- An acquisition Life Cycle  
- An operational Life Cycle, etc. These Life Cycles interact with one another via Life Cycle Interaction Points. Any Life Cycle is made up of one or more Stages. |
| Life Cycle Interaction Point | A Life Cycle Interaction Point defines a specific point at which one, more than one Life Cycle interacts with another. |
| Need             | A Need describes something that can be given meaning by a Use Case. A good example of this is a Requirement, where a Use Case would be defined as a Requirement that has been put into context. |
| Ontology         | Ontology - an element of an Architectural Framework that defines all the concepts and terms (Ontology Elements) that relate to any Architecture structured according to the Architectural Framework. |
| Ontology Element | Ontology Element - the concepts that make up an Ontology. Ontology Elements can be related to each other and are used in the definition of each Viewpoint (through the Viewpoint Element that make up a Viewpoint). The provenance for Ontology Elements is provided by one or more Standards. |
| Person           | A Person is an individual human being. Each Person takes on a number of Stakeholder Roles. Each Person has a number of Competency Profiles associated with it that defines the actual ability of that Person. A Person is also a type of Resource. |
| Perspective      | Perspective - a collection of Views (and hence also their defining Viewpoints) that are related by their purpose. That is, Views which address the same architectural needs, rather than being related in some other way, such as by mode of visualisation, for example. |
| Port             | Ports represent the interaction points between System Elements and may represent the concept of a software port or a physical port, such as the connector for the fuel line on a car engine fuel pump. Ports are connected to each other via Port Connections. |
| Port Connection  | A Port Connection connects together two or more Ports. A fuel rail taking fuel from the fuel pump to fuel injectors in a car engine would be represented by a Port Connection. |
| Process          | A Process describes an approach to achieving an end. A Process is made up of one or more Activity, one or more Artefact and one or more Stakeholder Roles. |
| Process Execution Group | A Process Execution Group represents a distinct set of Processes that are executed for a particular reason. These Process Execution Groups may be defined based on function (so there may be a 'component X' Process Execution Group), or by working area (so there may be a 'software' Process Execution Group), amongst others. |
| Protocol         | A Protocol is a description of the behaviour of a Port or an Interface. |
| Refinable Element| A Refinable Element represents a model element that may be refined (transformed) into one or more other model elements. A Refinable Element may be an element of an Architecture (such as a View or View Element) or of an Architectural Framework (such as a Viewpoint or Viewpoint Element). |
| Refinement Point | A Refinement Point represents the point at which one or more Refinable Elements interact. Refinement Points may occur at between levels of abstraction or within the same level of...
A Requirement is defined as: "...the definition of a property of a system that is either wanted or needed by a stakeholder."

A Requirement is a type of Need.

Rule

When describing any Need in natural language there is a lot of room for ambiguity and misinterpretation. In order to minimise these problems, the Rule concept allows the definition of a number of rules that are applied to Need descriptions.

These rules may apply to the Need itself or, more usually, to the properties of a Need. Sometimes these Rules will apply to the way that the wording in a Need description must be applied. Other rules concern the complexity of the text description that is being used to describe the Need.

Rules can also be used to constrain Architectural Frameworks and Refinement Points.

Scenario

A Scenario is defined as an exploration of a what if for a Use Case. Each Use Case will give rise to a number of different situations that may arise when it is being satisfied.

Semi-formal Scenario

Semi-formal Scenarios are realised by a semi-formal notation such as SysML, making use of, for example, sequence diagrams that show interactions between elements in the System. They can also be described informally using text as a set of scenario steps. Often the two are combined. Scenarios will normally be created for each type of context that has been developed. For example, if Stakeholder Context and System Contexts have been developed then both Stakeholder Role-level Scenarios and System-level Scenarios would be created. The Stakeholder Role-level scenarios typically treat the system as a black box and analyse the interactions between the Stakeholder Roles and the System. The System-level Scenarios would explore the interactions between system elements within the System.

Service-based Interface

Service-based Interfaces are used to represent those Interfaces that are operation or service-based such as are typically found in software-intensive systems.

Source Element

A Source Element describes the type of anything that can be used as the source for a Need. For example, a Book may be used as a source for a Requirement.

Stage

A Stage represents a discrete time period that describes a specific phase of a Life Cycle. These Stages are typically defined by the context in which the Life Cycle is being used. Before a Stage can be exited for any reason, it must pass through a Gate. A number of Process Execution Group may be executed during each Stage.

Stakeholder Role

A Stakeholder Role represents the role of any person, organisation or thing that has an interest in the system of project.

System

A System is defined as an interacting combination of System Elements that work together to achieve a set of Goals and satisfy a set of Needs. The artefact being engineered that an Architecture describes.

System Context

The System Context is a set of points of view that is based on the level of hierarchy of a System that may itself be broken down further into, for example, subsystems, assemblies and components. When considering such a hierarchy, it is usual to have a number of different types of Needs defined that exist at the various levels in the hierarchy: system requirements for a system, sub-system requirements derived from the system requirements for each sub-system and so on. Each hierarchical level will have one or more Contexts associated with it that consider the relevant requirements from that point of view, trace back to requirements at the higher level and establish the meaning of the requirements in that context.

System Element

Every System is made up of a number System Elements. For a System of Systems, each System Element is itself a System, known as a Constituent System.

System of Systems

A System of Systems is a collection of Constituent System that pool their resources and capabilities together to create a new, more complex System which offers more functionality and performance than simply the sum of the Constituent System.

Traceability Relationship

A Traceability Relationship represents the actual trace relationship between Traceable Elements.

Traceable Element

Traceable Elements are the actual Views or View Elements being traced along with the Traceability Relationships that represent the actual relationships which are being considered.

Traceable Type

Traceable Types are the types of elements which can be traced. These may be View Types or Element Types.

Use Case

A Use Case represents something that has been given meaning by being put into context. A good example of this is a Requirement, where a Use Case would be defined as a Requirement that has been put into context. This can also apply to a number of Needs, such as Goals and Capabilities.

View

A View is the visualisation of part of the Architecture of a System, that conforms to the structure and content defined in a Viewpoint. A View is made up of a number of View Elements.

View Element

View Elements are the elements that make up a View. Each View Element visualises a Viewpoint Element that makes up the Viewpoint to which the View, on which the View Element appears,
<table>
<thead>
<tr>
<th>Ontology Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>View Type</td>
<td>A View Type represents the types of View occurring in a model. View Types may represent underlying diagram types from the modelling language being used, such as a block definition diagram if using SysML or a class diagram if using UML. They may also represent defined views from a Framework that is being used, such as a Validation View from the COMPASS SoS-ACRE Framework.</td>
</tr>
<tr>
<td>Viewpoint</td>
<td>A Viewpoint is the definition of the structure and content of a View. The content and structure of a Viewpoint uses the concepts and terms from the Ontology via the Viewpoint Elements that make up the Viewpoint. Each Viewpoint is defined so that it meets the needs defined by its Viewpoint Concerns.</td>
</tr>
<tr>
<td>Viewpoint Concern</td>
<td>A Viewpoint Concern defines a Need that a Viewpoint has to address.</td>
</tr>
<tr>
<td>Viewpoint Element</td>
<td>Viewpoint Elements are the elements that make up a Viewpoint. Each Viewpoint Element must correspond to an Ontology Element from the Ontology that is part of the Architectural Framework.</td>
</tr>
<tr>
<td>Virtual</td>
<td>Virtual System of Systems lack a central management authority and a centrally agreed-upon purpose for the System of Systems. Large-scale behaviour emerges and may be desirable but this type of System of Systems must rely upon relatively invisible mechanisms to maintain it.</td>
</tr>
</tbody>
</table>

**Table 8 - Descriptions of Ontology Elements**
10. Appendix II – The COMPASS Architectural Framework

The diagram and associated descriptions presented in this appendix are intended to provide a useful summary of the COMPASS Architectural Framework. For details behind the various concepts please see the relevant deliverable, as detailed in Section 0.

Figure 95 - Viewpoint Relationships View for the COMPASS Architectural Framework
<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF Viewpoint</td>
<td>The AF Context Viewpoint, that defines the Context for the Architectural Framework. That is, it represents the Architectural Framework Concerns in Context.</td>
</tr>
<tr>
<td>Competency Framework Definition Viewpoint</td>
<td>The Competency Framework Definition Viewpoint defines a competency framework in terms of the Competency Areas it contains, the Competencies in each Competency Area and the Indicators that make up each Competency.</td>
</tr>
<tr>
<td>Competency Level Definition Viewpoint</td>
<td>The Competency Level Definition Viewpoint defines the Levels that are used in the competency framework.</td>
</tr>
<tr>
<td>Competency Scope Definition Viewpoint</td>
<td>The Competency Scope Definition Viewpoint defines a Competency Scope that is required for a particular Stakeholder Role. It is based on the Competencies defined in the Competency Framework Definition Viewpoint.</td>
</tr>
<tr>
<td>Competency Profile Definition Viewpoint</td>
<td>The Competency Profile Definition Viewpoint captures the Competency Profile for a Person, as assessed against a defined Competency Scope.</td>
</tr>
<tr>
<td>Context Definition Viewpoint</td>
<td>The Context Definition Viewpoint identifies the points of view that are explored in the Requirement Context Viewpoint. These points of view, or Contexts, may take many forms including Stakeholder Roles and levels of hierarchy in a System.</td>
</tr>
<tr>
<td>Context Interaction Viewpoint</td>
<td>The Context Interaction Viewpoint is intended to provide an overview of the relationships between the contexts of the various Constituent Systems that make up a System of Systems.</td>
</tr>
<tr>
<td>Definition Rule Set Viewpoint</td>
<td>The Definition Rule Set Viewpoint contains any Rules that may have to be applied to each Need description. For example, these may be complexity Rules in the form of equations or more general text-based Rules.</td>
</tr>
<tr>
<td>Impact Viewpoint</td>
<td>The Impact Viewpoint is used to show a traceability tree for a selected element from a Traceability Viewpoint, allowing the items potentially impacted by changes to the root of the tree (the selected element) to be identified.</td>
</tr>
<tr>
<td>Information Viewpoint</td>
<td>The Information Viewpoint identifies all the Artefact produced or consumed by each Activity within a Process and the inter-relationships between them.</td>
</tr>
<tr>
<td>Interface Behaviour Viewpoint</td>
<td>The Interface Behaviour Viewpoint is used for the identification of typical Scenarios showing how Interfaces are used.</td>
</tr>
<tr>
<td>Interface Connectivity Viewpoint</td>
<td>The Interface Connectivity Viewpoint used to show how Interfaces and Ports are connected.</td>
</tr>
<tr>
<td>Interface Definition Viewpoint</td>
<td>The Interface Definition Viewpoint is used for the definition of Interface in terms of the operations they may provide and the flows of data, material, energy, personnel etc. that take place across an Interface.</td>
</tr>
<tr>
<td>Interface Identification Viewpoint</td>
<td>The Interface Identification Viewpoint is used for the identification of Interfaces and Ports and their relation to the System Element that use them.</td>
</tr>
<tr>
<td>Ontology Definition Viewpoint</td>
<td>The Ontology Definition Viewpoint defines the Ontology for the Architectural Framework. It is derived from the AF Context Viewpoint.</td>
</tr>
<tr>
<td>Process Behaviour Viewpoint</td>
<td>The Process Behaviour Viewpoint shows how each individual Process behaves in terms of the order of each Activity within a Process, the flow of Artefacts through the Process, Stakeholder Role responsibilities and, where relevant, Resource usage.</td>
</tr>
<tr>
<td>Process Content Viewpoint</td>
<td>The Process Content Viewpoint identifies the actual Processes, showing each Activity carried out and the Artefacts produced and consumed. The Process Content Viewpoint may be considered as the library of Processes that is available to any Process-related Stakeholder Roles.</td>
</tr>
<tr>
<td>Process Instance Viewpoint</td>
<td>The Process Instance Viewpoint shows instances of Processes being exercised in a particular Scenario in order to show that the defined Processes meet the Requirements defined on the Requirement Context Viewpoint for the Process model.</td>
</tr>
<tr>
<td>Process Structure Viewpoint</td>
<td>The Process Structure Viewpoint specifies concepts and terminology that will be used for the Process modelling in the form of an Ontology.</td>
</tr>
<tr>
<td>Protocol Definition Viewpoint</td>
<td>The Protocol Definition Viewpoint is used for the definition of any Protocols that an Interface or Port must conform to.</td>
</tr>
<tr>
<td>Refinement Point Definition Viewpoint</td>
<td>The Refinement Point Definition Viewpoint is used to specify Refinement Points and the definition of s that may constrain the Refinement Points.</td>
</tr>
<tr>
<td>Refinement Point Identification Viewpoint</td>
<td>The Refinement Point Identification Viewpoint is used to identify Refinement Points, which may apply between abstraction levels or within a single abstraction level. It identifies source and target Refinable Elements for each Refinement Point.</td>
</tr>
<tr>
<td>Relationship Identification Viewpoint</td>
<td>The Relationship Identification Viewpoint is used to define the types of permissible traceability relationships.</td>
</tr>
<tr>
<td>Requirement Context Viewpoint</td>
<td>The Requirement Context Viewpoint takes the Needs (a Requirement, Goal or Capability) that have been defined on the Requirement Description Viewpoint and gives them meaning by looking at them from a specific point of view (Context).</td>
</tr>
</tbody>
</table>
The Requirement Context Viewpoint contains structured descriptions of each Need. These requirements are considered individually and will usually have a number of attributes, or features, associated with each one. The main purpose of this Viewpoint is to describe each individual Need according to a predefined set of attributes. These attributes will vary depending on the Process that is being followed, the industry that the work is being carried out in, any Standards or best-practice models that may be being used and any other number of factors. This Viewpoint is primarily used for managing the Needs of a System and is often the basis of implementation for many of the commercial requirements management tools that are on the market today. Each Need description provides a non-contextual description of the Need. When a Need is put into Context, it is known as a Use Case and, hence, there is a very strong relationship between the Need descriptions and the Use Cases from the Requirement Context Viewpoints.

<table>
<thead>
<tr>
<th>Viewpoint Definition Viewpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement Description Viewpoint</td>
<td>The Requirement Context Viewpoint contains structured descriptions of each Need. These requirements are considered individually and will usually have a number of attributes, or features, associated with each one. The main purpose of this Viewpoint is to describe each individual Need according to a predefined set of attributes. These attributes will vary depending on the Process that is being followed, the industry that the work is being carried out in, any Standards or best-practice models that may be being used and any other number of factors. This Viewpoint is primarily used for managing the Needs of a System and is often the basis of implementation for many of the commercial requirements management tools that are on the market today. Each Need description provides a non-contextual description of the Need. When a Need is put into Context, it is known as a Use Case and, hence, there is a very strong relationship between the Need descriptions and the Use Cases from the Requirement Context Viewpoints.</td>
</tr>
<tr>
<td>Source Element Viewpoint</td>
<td>The Source Element Viewpoint contains all relevant source information that is required to get the Needs right. It is essential that the origin of all Needs is known and this is what this Viewpoint allows us to define. This Viewpoint is used primarily as a mechanism to establish traceability in the System and provide links between the Needs and any other aspect of the System.</td>
</tr>
<tr>
<td>Stakeholder Viewpoint</td>
<td>The Stakeholder Viewpoint identifies the Stakeholder Roles that have an interest in the Processes being defined. It presents Stakeholder Roles in a classification hierarchy and allows additional relationships, such as managerial responsibility, to be added.</td>
</tr>
<tr>
<td>Traceability Identification Viewpoint</td>
<td>The Traceability Identification Viewpoint is used to define the items that can be traced and the types of trace that can be used between items.</td>
</tr>
<tr>
<td>Traceability Viewpoint</td>
<td>The Traceability Viewpoint is used to capture and visualise traceability. This may be in the form of a diagram, table, matrix etc.</td>
</tr>
<tr>
<td>Validation Interaction Viewpoint</td>
<td>The Validation Interaction Viewpoint is intended to provide a combined view of the Scenarios for Use Cases that are involved in the System of Systems.</td>
</tr>
<tr>
<td>Validation Viewpoint</td>
<td>The Validation Viewpoint provides the basis for demonstrating that Need can be met or complied with in some way. Views based on the Validation Viewpoint can be informal (such as Scenarios at various levels of abstraction) or may be formal (such as mathematical-based representation).</td>
</tr>
<tr>
<td>Viewpoint Context Viewpoint</td>
<td>The Viewpoint Context Viewpoint, that defines the Context for a particular Viewpoint. That is, it represents the Viewpoint Concerns in Context for a particular Viewpoint. It is derived from the AF Context Viewpoint.</td>
</tr>
<tr>
<td>Viewpoint Definition Viewpoint</td>
<td>The Viewpoint Definition Viewpoint defines a particular Viewpoint, showing the Viewpoint Elements (and hence the Ontology Elements) that appear on the Viewpoint.</td>
</tr>
<tr>
<td>Viewpoint Relationships Viewpoint</td>
<td>The Viewpoint Relationships Viewpoint shows the relationships between the Viewpoints that make up the Architectural Framework and groups them into Perspectives. It is derived from the Ontology Definition Viewpoint.</td>
</tr>
</tbody>
</table>

Table 9 - Descriptions of Viewpoints
11. Appendix III - Summary of the COMPASS Processes and Guidelines

The table below summarises the various Process and Guidelines identified in Section 2.3. For full details of each Process, refer to the original deliverable document.

<table>
<thead>
<tr>
<th>Process</th>
<th>Description (Summary)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SoS Requirements Processes</strong></td>
<td></td>
</tr>
<tr>
<td>System of Systems Requirements Engineering Process Group</td>
<td></td>
</tr>
<tr>
<td>SoS Requirements Development Process</td>
<td>The main aim of the SoS Requirements Development Process is to perform most of the Requirements engineering at the SoS level. This involves defining the Contexts at SoS and Constituent System (CS) level and identifying the relationships and interactions between them. This Process uses the Requirements Elicitation Process, the Context Process (at both SoS and CS levels), the Traceability Process and the Verification and Validation Definition Process (via the Context Process).</td>
</tr>
<tr>
<td>Requirements Elicitation Process</td>
<td>The main aim of the Requirements Elicitation Process is to take the Source Element Views and, from them, elicit the parts that are relevant and create the Requirement Description View.</td>
</tr>
<tr>
<td>Context Process</td>
<td>The main aim of the Context Process is to define a Context based on the Context Definition View. This is a generic Process that may be invoked from the SoS Requirements Development Process and may be applied at both the SoS and the CS level.</td>
</tr>
<tr>
<td>Verification and Validation Definition Process</td>
<td>The main aim of the Verification and Validation Definition Process is to define a number of Scenarios for each Use Case in a specific Context. These Scenarios may be either semi-formal (diagram-based) or formal (mathematical-based) and form the basis of the testing of the SoS. These Scenarios are defined for both verification (it works) and validation (it does what it is supposed to do) for the Use Cases.</td>
</tr>
<tr>
<td><strong>System of Systems Requirements Management Process Group</strong></td>
<td></td>
</tr>
<tr>
<td>Traceability Process</td>
<td>The overall aim of the Traceability Process is to enable traceability to be established between any elements in the model. This may be used for the Requirements model but may also trace to any elements in the wider SoS model or its CS models.</td>
</tr>
<tr>
<td>CS Process Analysis Process</td>
<td>The overall aim of the CS Process Analysis Process is to understand the Requirement management Process of the Constituent Systems (CSs) that make up the SoS. It is important to monitor the Requirements of the CSs so that any changes can be identified and evaluated. In order to do this there needs to be an understanding of the Requirement management Process of each of the CSs. This will be achieved by modelling each Requirement management Process and then mapping to the SoS Requirement management Process. Once this understanding has been achieved and mapped to the SoS Requirement management Process, then a number of control points can be set up that allow Requirements changes to be identified periodically. In the event that the Requirement management Processes of the SoS and its CSs are not compatible, then an exception is raised. This would then require action be taken to address the incompatibility between the Processes.</td>
</tr>
<tr>
<td>Requirement Control Process</td>
<td>The overall aim of the Requirement Control Process is:</td>
</tr>
<tr>
<td></td>
<td>- To ensure that all information contained in the Requirement model is communicated to the relevant Stakeholder Roles.</td>
</tr>
<tr>
<td></td>
<td>- To ensure that the Requirements model is reviewed and that a consensus is achieved between the relevant Stakeholder Roles.</td>
</tr>
<tr>
<td></td>
<td>- To obtain commitment from the Stakeholder Roles that the consensus is the most appropriate way forward and to allocate suitable resources to ensure that the Requirements are satisfied.</td>
</tr>
<tr>
<td>Requirements Change Process</td>
<td>The main aim of the Requirements Change Process is to identify any changes to Requirements, assess the impact and take appropriate actions. This process may be applied at both the SoS and the CSs level and can invoke another instance of itself.</td>
</tr>
</tbody>
</table>
### Requirements Monitor Process

The aim of the Requirements Monitor Process is:

- To allow Requirements from the CSs that make up the SoS to be monitored for change via Control Points.
- To allow Requirements from the SoS to be monitored.

Should any change occur in either the SoS or any of its CSs, then the Requirements Change Process will be invoked.

### Architectural Framework Definition Processes

#### AF Definition Process

The aim of this Process is to understand the underlying need for the Architectural Framework (AF). The Context of the AF that is under development is defined via the Context Process, so that the Needs that it is to address are clearly understood and any source Standards that the AF under development may need to comply with are identified.

Based on the AF Context View produced, the Ontology for the AF under development is now defined by invoking the Ontology Definition Process. This returns the Ontology Definition View.

Based on the Ontology and the AF Context, a number of Viewpoints are now identified along with the relationships between them. This results in the Viewpoint Relationship View.

#### Context Process

The main aim of this Process is to create a Context that can be used to create either an AF Context View or a Viewpoint Context View.

The Processes produces a Context that identifies Use Cases based on the Needs, identifies the System boundary, identifies Stakeholder Roles that are external to the System boundary, identifies relationships between the Use Cases and the external Stakeholder Roles and identifies relationships between Use Cases.

The Context is then analysed to ensure that all Use Cases, Stakeholder Roles and their relationships are understood. Any problems that may have been identified during the Context analysis are addressed.

#### Ontology Definition Process

The main aim of this Process is to identify and define the main concepts and terms used for the AF in the form of an Ontology.

The concepts that are relevant to the AF under development are identified and defined, with their provenance established against source material. Such concepts now become Ontology Elements. These Ontology Elements are then related one to another to become part of the Ontology, which is realised as the Ontology Definition View.

#### Viewpoint Definition Process

The main aim of this Process is to identify and define the key Viewpoints and to classify them into Perspectives.

Each Viewpoint from the Viewpoint Relationship View (produced by the AF Definition Process) is considered in turn. For the Viewpoint, its Context is created by invoking the Context Process. This produces the Viewpoint Context View for the selected Viewpoint.

Based on the Ontology Definition View, a subset of the Ontology is now identified that is judged to be relevant for the selected Viewpoint. This is used to define the Viewpoint Definition View for the selected Viewpoint.

Any relationships that exist between the Viewpoints are identified and added to the Viewpoint Relationship View. This is used as an input for defining Rules for ensuring consistency of the Viewpoints defined, captured as the Rules Definition View.

### Architecture Guidelines

#### Determine SoS Requirements and Stakeholders

This activity involves translation of business needs for the SoS into verifiable Requirements, along with identification of a preliminary set of Stakeholders for each Requirement. This is going to be refined by tracing Requirements further to CSs that are also associated with Stakeholders. At this time we do not take into account the CSs.

See [D21.5a 2013] Section 8.

#### Determine CSs and their Stakeholders

The understanding of the SoS is still incomplete. Understanding the CSs, their relationships and Stakeholders is one of the major objectives of this activity. This has far-reaching consequences on feasible sets of Requirements, degraded operation and on SoS integration.
### Process Description (Summary)

Two principle kinds of Stakeholders must be distinguished: technological and business. Many of the constraints seen in CSs are not technological in nature but related to business goals and interests. We collect the terms like societal, legal, i.e., “non-technological” in the one term: business. The characterisations of SoSs and CSs tend to be driven by business constraints. It is advisable to consider business interests of the Stakeholders first and defer technological considerations to a later point. Whereas for technological problems technological solutions are often not far away, business constraints may disallow such solutions.

This activity needs to be carried out continually while the Architecture of the SoS is developed and made gradually more specific.

See [D21.5a 2013] Section 9.

#### Outline SoS Architecture

The high complexity of a typical SoS makes it difficult to reason about its Architecture directly. In order to approach a complete model of the SoS we first make some simplifying assumptions: We assume that all features of all CSs are immediately accessible and all interfaces and protocols are compatible. Essentially, we assume that we are analysing an ordinary system. The SoS constraints are ignored. Using an outline SoS Architecture we argue that the Architecture satisfies the Requirements. This provides assurance that the CSs of our initial list are, in principle, sufficient to realise the SoS.

See [D21.5a 2013] Section 10.

#### Classify CSs and their components

The outline SoS Architecture shows what is required from each CS in terms of functionality, interfaces and protocols. For each CS it is necessary to determine how it could support all of these, provided the actual functionality, interfaces and protocols were compatible. We identify components with different groups of functionality, interfaces and protocols and classify them. We distinguish an SoS into three decomposition layers: SoS, CS, component. The component layer is associated with concrete functionality. The reason for considering components as subdivisions of CSs is that usually each CS has functionality with different degrees of openness. For instance, service discovery may be freely accessible whereas requesting diagnoses of a patient record stored on some device may not be.

A CS and its different components can be classified into the different categories forbidden, closed/legacy, extensible and open. Each category describes the degree of openness towards the integration into a given SoS.

See [D21.5a 2013] Section 12.

#### Improve outline SoS Architecture

The information on what the CSs really provide is added to the outline SoS Architecture. The CS classification determines how this can be done. Depending on how components and CSs are classified: the components can be modified or not; additional components or CSs may have to be included; access to a component may be allowed or disallowed.

During this activity, communication paradigms and architectural styles are established.

See [D21.5a 2013] Section 13.

#### Vary configurations and classify them as SoS

The purpose of this activity is to determine characteristics of the SoS in order to develop integration and verification strategies. An SoS may belong to several categories depending on the different configurations considered. The result of this step depends on the configurations actually analysed. Which configurations should be considered depends on which configurations are possible and what set of requirements they satisfy. Some configurations may be chosen for verification concerns, even if it would never actually occur. The result of this task includes an analysis of configurations versus Requirements providing insight into full performance with respect to Requirements, or degraded performance.

See [D21.5a 2013] Section 14.

#### Outline detailed SoS Architecture

A detailed Architecture of the SoS is produced. The Architecture must support all configurations. The different configurations should be indicated and their SoS classifications stated.

See [D21.5a 2013] Section 15.
<table>
<thead>
<tr>
<th>Process</th>
<th>Description (Summary)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System Integration Processes</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Original Deliverable</strong></td>
<td>D21.4 2013 Report on Guidelines for System Integration for SoS</td>
</tr>
<tr>
<td>Constituent System Identification Process</td>
<td>The Constituent System Identification Process identifies the SoS and its component Constituent Systems (CSs). It combines the Requirement Context Views for the CSs into the Context Interaction View for the SoS and uses this to help identify the Interfaces between CSs, producing Interface Identification Views for these Interfaces. All Artefacts produced are reviewed and are placed under formal configuration control.</td>
</tr>
<tr>
<td>Integration Process</td>
<td>The Integration Process takes each Interface identified in the Constituent System Identification Process and expands on the definition of these Interfaces by defining the Interface connectivity, where the connections between the System Elements of the SoS (i.e. between the CSs) are defined by creating a number of Interface Connectivity Views. The structure of each Interface is defined by creating the Interface Definition Views and the interactions between the System Elements (the CSs) and across the selected Interface are defined by creating the Interface Behaviour Views. Any Protocols, if required, are defined for each Interface by creating the Protocol Definition Views. All Artefacts produced are reviewed and are placed under formal configuration control.</td>
</tr>
<tr>
<td>Validation Process</td>
<td>In the Validation Process, the Validation Interaction Views for the SoS are produced or collected together. These form the main validation Scenarios for the SoS. The Interface Behaviour Views for the Interfaces (as produced by the Integration Process) are collected together and used to perform the actual validation by comparing the set of Interface Behaviour Views for the Interfaces to the Validation Interaction Views for the SoS. All Artefacts produced are reviewed and are placed under formal configuration control.</td>
</tr>
<tr>
<td>Integration Strategy Process</td>
<td>The Integration Strategy Process defines the Needs of the integration by creating a Requirement Context View from the point of view of the Integration. 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. The Processes that will be carried out in order to perform integration are identified by identifying an appropriate Process Content View. In the case of the COMPASS Project, this will be the Systems Integration Processes defined in D21.4 2013. These Processes identified are validated against the original Needs through a number of Process Instance views. Based on the above, an ‘Integration Plan’ for the integration activities is created. All Artefacts produced are reviewed and are placed under formal configuration control.</td>
</tr>
<tr>
<td><strong>Refinement Processes</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Original Deliverable</strong></td>
<td>THIS DOCUMENT – SEE SECTION 2.3.4</td>
</tr>
<tr>
<td>Refinement Point Process</td>
<td>The main aim of the Process is to identify and define the Refinement Points for a specific System Model. The Process begins with the System Modeller who, based on the System Model, identifies the Views that contain the Refinement Points. These Views are then used to identify a pair of Refinement Points and then this is repeated until all of the Refinement Points have been identified. Once all of the Refinement Points have been identified, the System Modeller then defines the Rules that apply to each Refinement Point pair, which is repeated until Rules have been defined for all Refinement Point pairs. The Reviewer then reviews all of the Process Artefacts to ensure that they are correct and, if the review is passed, then the Configuration Manager baselines all Process Artefacts. In the event that the review is failed, the control of the Process flow goes back to the identification of Views or definition of Rules, depending on whether the review</td>
</tr>
<tr>
<td>Process</td>
<td>Description (Summary)</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------</td>
</tr>
<tr>
<td><strong>Competency Processes</strong></td>
<td></td>
</tr>
<tr>
<td>Original Deliverable</td>
<td><strong>THIS DOCUMENT – SEE SECTION 4.2</strong></td>
</tr>
<tr>
<td>Bespoke Competency Definition Process</td>
<td>The main aim of the Process is to define the Competencies for a bespoke Competency Framework based on a source MBSE Ontology and Source Frameworks. The Process begins with the Competency Framework Definer who starts by identifying and, if necessary, modelling the source MBSE Ontology. Any Source Frameworks are then identified and, if necessary, modelled. With the MBSE Ontology and any Source Frameworks identified and modelled, the Competency Framework Definer next defines Competency Areas and identifies the Competencies that will be defined. These identified Competencies are then defined in detail, along with their Indicators and any descriptive material etc., through three parallel activities that look at concept-related, process-related and skill-related Competencies. All the material is then reviewed by the Reviewer and a review report produced. If the result of the review is a fail, then the Process is repeated, starting with the identification of Source Frameworks. If the review is a success, then the initial version of the bespoke Competency Framework is issued.</td>
</tr>
<tr>
<td>Bespoke Framework Definition Process</td>
<td>The main aim of the Process is to take the outline Competency Framework produced in the Bespoke Competency Definition Process and complete its definition. The Process begins with the Competency Framework Definer who starts by taking the initial version of the bespoke Competency Framework and analysing it in order to determine how best to define the Levels at which Competencies can be held. Having determined the appropriate Levels for each Competency, these Levels are then defined for each Competency. All the material is then reviewed by the Reviewer and a review report produced. If the result of the review is a fail, then the Process is repeated. If the review is a success, then the final version of the bespoke Competency Framework is issued.</td>
</tr>
<tr>
<td>Assessment Set-up Process</td>
<td>The main aim of the Process is to identify the needs and purpose of a planned Competency assessment (or set of assessments) and to define the Competency Scopes that will be used in the assessments. The Process begins with Human Resources who identify the Assessment Criteria (the needs and purpose) of the assessment that is to take place. These criteria, along with the bespoke Competency Framework, are then used to define the required Competency Scopes against which assessments will take place. All the material is then reviewed by the Reviewer and a review report produced. If the result of the review is a fail, then the Process is repeated. If the review is a success, then the Process ends.</td>
</tr>
<tr>
<td>Assessment Process</td>
<td>The main aim of the Process is to carry out the Competency assessment of a Person (the Assessee) against a Competency Scope that has been produced, from a Competency Framework, for a specific Stakeholder Role. The Process begins with the Assessor who gathers together any necessary material, such as Competency Scopes, the bespoke Competency Framework, guidance etc. He then selects a Competency from the Competency Scope to assess next, selects the Level for the Competency being assessed, from those Levels identified on the Competency Scope and perform an assessment of the Assessee against an Indicator for the selected Competency at the selected Level. This is repeated for all the Indicators in the Competency at the selected Level. If there are more Levels for the selected Competency then this selection of Level and assessment of Indicators until all the Levels for the selected Competency have been assessed. The Assessor then updates the Assessee’s Competency Profile with the results so far and all the above is repeated until all the Competencies in the Competency Scope have been assessed. The Assessor then discusses the resulting Competency Profile with the Assessee and the process ends.</td>
</tr>
</tbody>
</table>

**Table 10 - Summary of the COMPASS Processes and Guidelines**

157
12. Appendix IV - The Existing COMPASS Ontologies

This section contains the existing COMPASS Ontologies that are found in a number of other COMPASS documents. An Ontology Definition View (one of the Viewpoints defined in the COMPASS Architectural Framework Framework (CAFF), see [D21.5b 2014]) is given for each of the existing Ontologies. Each Ontology is presented without comment, except where changes have been made to the original version in order to address either consistency issues or in order to bring the Ontologies in line with research conclusions developed since publication of the originals.

Figure 96 shows the Ontology developed for SoS Requirements. Full details can be found in [D21.1 2012].

Figure 97 shows the Ontology developed for Processes and Competency. This is based on the one found in Appendix I of [D21.1 2012]. It has been expanded here to cover more concepts relating to Competency, namely ‘Competence’, ‘Competency Area’, ‘Competency’, ‘Level’ (and its types) and ‘Indicator’. For a discussion of these concepts see Section 4.
Figure 97 - Ontology Definition View focussing on Processes and Competency

Figure 98 shows the Ontology developed for Architectures and Architectural Frameworks. Full details can be found in [D21.5b 2014]. One small change has been made to the Ontology. The multiplicity on the association between the Perspective and Viewpoint blocks has been changed from ‘1’ to ‘1..*’ at the Perspective end. This has been done to reflect the re-use of Viewpoints in multiple Perspectives, as was discussed in Section 2.2.

Figure 98 - Ontology Definition View focussing on Architectures and Architectural Frameworks

Figure 99 shows the Ontology developed for SoS Integration. Full details can be found in [D21.4 2013].
Figure 99 - Ontology Definition View focussing on Integration

Figure 100 shows the Ontology developed for traceability. It is taken from the traceability pattern, one of the enabling patterns found in [D22.6 2014]. Although not presented as a separate Framework in [D22.6 2014], the pattern nevertheless defines a set of concepts that can (and should) be applied across all the other Frameworks.

Figure 100 - Ontology Definition View focussing on traceability

Figure 101 shows the Ontology developed for SoS refinement. Full details can be found in [D22.5 2013]. Note that the Ontology has been changed slightly from that found in [D22.5 2013]; the concept of the Refinable Element has been added to better indicate that refinement can take place at both the Architecture and Architectural Framework level.
The references for these Ontologies are summarised below:

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Original Deliverable</th>
</tr>
</thead>
</table>

Table 11 - Original sources for Ontologies
13. Appendix V - The Existing COMPASS Frameworks

This section contains the existing COMPASS Frameworks that are found in a number of other COMPASS documents and that are based on the Ontologies presented in Section 12. A Viewpoint Relationships View (one of the Viewpoints defined in the CAFF, see [D21.5b 2014]) is given for each of the existing Frameworks, with the Framework represented as a Perspective through the use of a SysML package. Each Framework is presented without comment, except where changes have been made to the original version in order to address either consistency issues that have been discovered since the original documents were issued or in order to bring the Frameworks in line with research conclusions developed since publication of the originals.

Figure 102 shows the Framework developed for SoS Requirements. Full details can be found in [D21.1 2012]. Note that in [D21.1 2012] each of the Viewpoints are named as Views. For example, in [D21.1 2012] the ‘Validation Viewpoint’ is named the ‘Validation View’. The renaming from View to Viewpoint has been done here to bring the concepts in line with those found in the CAFF where a Viewpoint represents the definition and a View represents an instance of a Viewpoint. As can be seen in Figure 98 (and Figure 2) an Architectural Framework is made up of Viewpoints, whereas an Architecture which conforms to an Architectural Framework is made up of Views.

Figure 103 shows the Framework developed for Process modelling. This is based on the one found in Appendix I of [D21.1 2012]. Again, to bring terminology in line with the CAFF, all the Views found in [D21.1 2012] have been renamed as Viewpoints. Note also that although the Ontology on which this Framework is based (see Figure 97) includes some concepts relating to Competency, no Viewpoints that cover these concepts are found in [D21.1 2012]. A Competency Perspective can be found in Section 4.1 above.
Figure 103 - Viewpoint Relationships View for the “Seven Views” Process Framework

Figure 104 shows the Framework developed for Architectures and Architectural Frameworks – the CAFF, based on the Ontology in Figure 98. Full details can be found in [D21.5b 2014].

Figure 104 - Viewpoint Relationships View for the COMPASS Architectural Framework Framework (CAFF)

Figure 105 shows the Framework developed for SoS Integration. Full details can be found in [D21.4 2013].
Figure 105 - Viewpoint Relationships View for the Integration Framework

Figure 106 shows the Framework developed for traceability. It is taken from the traceability pattern, one of the enabling patterns found in [D22.3 2013]. Although not presented as a separate Framework in [D22.3 2013], the pattern nevertheless defines a set of concepts that can (and should) be applied across all the other Frameworks.

Figure 106 - Viewpoint Relationships View for the Traceability Framework

Again, to bring terminology in line with the CAFF, all the traceability Views found in [D22.3 2013] have been renamed as Viewpoints.

Figure 107 shows the Framework developed for SoS Refinement. Full details can be found in [D22.5 2013].
The references for these Frameworks are summarised below:

<table>
<thead>
<tr>
<th>Framework</th>
<th>Original Deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process modelling</td>
<td></td>
</tr>
<tr>
<td>Frameworks</td>
<td></td>
</tr>
</tbody>
</table>

Table 12 - Original sources for Frameworks