## Concept Alignment Example description

**D_3.3**

<table>
<thead>
<tr>
<th>Deliverable Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nature</strong></td>
</tr>
<tr>
<td><strong>Project</strong></td>
</tr>
<tr>
<td><strong>Deliverable ID</strong></td>
</tr>
<tr>
<td><strong>Status</strong></td>
</tr>
<tr>
<td><strong>Contact Person</strong></td>
</tr>
<tr>
<td><strong>Phone</strong></td>
</tr>
<tr>
<td><strong>Associated Materials</strong></td>
</tr>
</tbody>
</table>
AUTHORS TABLE

<table>
<thead>
<tr>
<th>Name</th>
<th>Company</th>
<th>E-Mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim Lochow</td>
<td>EADS DE</td>
<td><a href="mailto:Tim.lochow@eads.net">Tim.lochow@eads.net</a></td>
</tr>
<tr>
<td>Imad Sanduka</td>
<td>EADS DE</td>
<td><a href="mailto:Imad.Sanduka@eads.net">Imad.Sanduka@eads.net</a></td>
</tr>
<tr>
<td>Rene Bullinga</td>
<td>EADS DE</td>
<td><a href="mailto:Rene.Bullinga@cassidian.com">Rene.Bullinga@cassidian.com</a></td>
</tr>
<tr>
<td>Alexandre Arnold</td>
<td>EADS FR</td>
<td><a href="mailto:Alexandre.Arnold@eads.net">Alexandre.Arnold@eads.net</a></td>
</tr>
<tr>
<td>Roy Kalawsy</td>
<td>LU</td>
<td><a href="mailto:R.S.Kalawsky@lboro.ac.uk">R.S.Kalawsky@lboro.ac.uk</a></td>
</tr>
<tr>
<td>Gerard Cristau</td>
<td>THALES</td>
<td><a href="mailto:gerard.cristau@thalesgroup.com">gerard.cristau@thalesgroup.com</a></td>
</tr>
<tr>
<td>Martin Jung</td>
<td>CARMEQ</td>
<td><a href="mailto:martin.jung@carmeq.com">martin.jung@carmeq.com</a></td>
</tr>
<tr>
<td>Christoph Etzien</td>
<td>OFFIS</td>
<td><a href="mailto:christoph.etzien@offis.de">christoph.etzien@offis.de</a></td>
</tr>
<tr>
<td>Eric Honour</td>
<td>HCODE</td>
<td><a href="mailto:ehonour@hcode.com">ehonour@hcode.com</a></td>
</tr>
</tbody>
</table>

CHANGE HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Reason for Change</th>
<th>Pages Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>20/08/2012</td>
<td>1st Draft, List of Contents</td>
<td>All</td>
</tr>
<tr>
<td>0.2</td>
<td>14/09/2012</td>
<td>2nd Draft, List of Content, identification of contributors</td>
<td>All</td>
</tr>
<tr>
<td>0.2</td>
<td>18/09/2012</td>
<td>3rd draft, List of contents</td>
<td>All</td>
</tr>
<tr>
<td>0.3</td>
<td>16/10/2012</td>
<td>Content Chapter 1-6</td>
<td>All</td>
</tr>
<tr>
<td>0.4</td>
<td>26/10/2012</td>
<td>Contributions from LU, OFFIS, Thales, Editing of chapters</td>
<td>All</td>
</tr>
<tr>
<td>1.0</td>
<td>31/10/2012</td>
<td>Release of Intermediate Version 1.0 (M12) Including review remarks</td>
<td>All</td>
</tr>
<tr>
<td>1.1</td>
<td>18/04/2013</td>
<td>Collect updates contribution from OFFICE,EADS</td>
<td>All</td>
</tr>
<tr>
<td>2.0</td>
<td>25/04/2013</td>
<td>Release of final version</td>
<td></td>
</tr>
</tbody>
</table>

Version | Status     | Date       | Page |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>Final</td>
<td>2013-04-25</td>
<td>2</td>
</tr>
</tbody>
</table>
INTRODUCTION

1.1 Purpose of the document ................................................................. 8
1.2 Inputs used for the work ............................................................... 8
1.3 Addressed readers ..................................................................... 8
1.4 Structure of the document .......................................................... 9

CONCEPT ALIGNMENT EXAMPLE (CAE) DESCRIPTION ................................................. 10

2.1 CAE Overview ........................................................................... 10
2.1.1 The Emergency Response System of Systems ...................... 10
2.1.2 CAE Models Overview .......................................................... 12
2.2 CAE Versus DANSE Requirements ........................................... 13

CAE – MODELLING THE SOS ARCHITECTURE......................................................... 15

3.1 Challenges of an Exemplified Emergency Response SoS .................. 15
3.2 Conflicting Goals ...................................................................... 16
3.3 SoS Modelling using SoS Architecture Frameworks ................... 18
3.3.1 Define the modelling purpose .............................................. 18
3.3.2 Define and analyse the SoS constituent systems ................... 18
3.3.3 Choose Modelling Language ............................................... 21
3.3.4 Build the SoS model ............................................................ 21
3.4 Collaboration in SoS Modelling .................................................. 28

CAE – MODELLING SOS ARCHITECTURE ALTERNATIVES FOR DESIGN SPACE EXPLORATION .................................................................................. 30

4.1 SoS Architectural Patterns ............................................................. 30
4.1.1 Centralized and decentralized Architecture ............................ 31
4.2 SoS Architecture Optimisation ...................................................... 34
4.2.1 Optimization Concept ......................................................... 34
4.2.2 ER SoS - Communication System Optimization .................. 35
4.2.3 Optimization Goals ............................................................. 36
4.2.4 Mathematical description .................................................... 36
4.2.5 Matlab Model .................................................................... 37
4.3 Modelling SoS Architectural Dynamicity .................................. 38
4.3.1 Modelling Concepts ............................................................ 38
4.3.2 Implementation ................................................................. 39
4.3.3 Dynamicity applied to the CAE ......................................... 40

CAE – SOS BEHAVIOURAL MODELLING FOR RUN TIME ANALYSIS ......................................................... 42

5.1 Constituent Systems Behaviour ................................................... 42
5.2 SoS Stochastic/Statistical Modelling .......................................... 52
5.3 Human Behaviour Modelling ..................................................... 55
5.3.1 Objectives ........................................................................ 55
5.3.2 Relevance of Human Behaviour (HB) in SoS: ....................... 55
5.4 SoS Goals Formalisation ............................................................ 57
5.5 Analysis of Alternatives Performance ....................................... 58
5.6 Emergent Behaviour Analysis .................................................... 58
5.6.1 CAE Examples of Emergent Behaviour .............................. 58
5.6.2 DANSE Discovery and Control of Emergent Behaviours ....... 59

CAE- DANSE TECHNOLOGY INTEGRATION ................................................................................. 61
6.1 INTEGRATION MODEL ........................................................................................................................................ 61
  6.1.1 Detailed UPDM Model .................................................................................................................................. 61
  6.1.2 Intergradation Model Scenario ....................................................................................................................... 62
6.2 TECHNOLOGY INTEGRATION FLOW ...................................................................................................................... 64
6.3 INTERGRADATION MODEL EXTENSIONS ........................................................................................................ 65
6.4 PREDICTIVE RUN TIME OPTIMIZATION ........................................................................................................... 69

7 CONCLUSION ......................................................................................................................................................... 71

8 REFERENCES .......................................................................................................................................................... 72
List of Figures

Figure 2-1: Emergency Response SoS Dynamicity aspects and SoSE Challenges ........................................ 11
Figure 2-2: CAE breakdown Overview ........................................................................................................ 12
Figure 2-3: DANSE Requirements ............................................................................................................... 13
Figure 3-1: Process for building the ER SoS UPDM model ........................................................................ 22
Figure 3-2: ER SoS High level operational view OV-1 ................................................................................. 23
Figure 3-3: House Fire Scenario OV-5 ......................................................................................................... 24
Figure 3-4: CAE Operational node connectivity description OV-2 ............................................................... 25
Figure 3-5: CAE Operational Information Requirement OV-3 ................................................................. 26
Figure 3-6: CAE System Functionality View SV-4 (Dispatching) ............................................................... 27
Figure 3-7: CAE System View SV-1 ........................................................................................................... 28
Figure 3-8: Model Collaboration in CAE model ......................................................................................... 29
Figure 4-1: Hierarchy of SoS Patterns .......................................................................................................... 31
Figure 4-2: Centralized Architecture patterns example .............................................................................. 31
Figure 4-3 Centralized System .................................................................................................................... 32
Figure 4-4: Decentralized System ................................................................................................................ 32
Figure 4-5: Architecture pattern for the JESCC .......................................................................................... 33
Figure 4-6: Level 0 CAE network coverage optimization .......................................................................... 34
Figure 4-7: CAE optimization concept ...................................................................................................... 35
Figure 4-8: Communication coverage optimization case ............................................................................ 35
Figure 4-9: Matlab model results ................................................................................................................ 37
Figure 4-10 Change communication ......................................................................................................... 39
Figure 4-11 New Region .............................................................................................................................. 40
Figure 4-12: Add relation to district ............................................................................................................ 41
Figure 5-1: SV-1 BDD of the CAE behavioural model ................................................................................. 42
Figure 5-2: City Districts .............................................................................................................................. 43
Figure 5-3: SV-1 of the SoS subset ............................................................................................................... 44
Figure 5-4: District behaviour (SV-10b statechart) .................................................................................... 45
Figure 5-5: Simplified behaviour of Fire Headquarter (SV-10b statechart) ................................................. 46
Figure 5-6: Simplified behaviour of Fire Station (SV-10b statechart) ......................................................... 47
Figure 5-7: Simplified behaviour of Fire Fighting Car (SV-10b statechart) ................................................. 49
Figure 5-8: Modelica code of a fireman ...................................................................................................... 51
Figure 5-9: Simulation of one fireman inside OMEedit - model ................................................................. 51
Figure 5-10: Simulation of one fireman inside OMEedit - result ............................................................... 51
Figure 5-11: (SysML) Stereotypes to represent stochastic models ............................................................. 52
Figure 5-12: DANSE Stochastic profile ..................................................................................................... 54
Figure 6-1 Command and Control Center Operational Node representation ........................................... 61
Figure 6-2 OV-5 CCC Operational activity flow example ......................................................................... 62
Figure 6-3 Integration Model Scenario 1 .................................................................................................... 63
Figure 6-4 Integration Model Scenario 2 .................................................................................................... 63
Figure 6-5 Integration Model Scenario 3 .................................................................................................... 64
Figure 6-6 Technologies integration flow .................................................................................................. 65
Figure 6-7 Operational activities to functions mapping ............................................................................ 66
Figure 6-8 Operational activities to functions mapping matrix ............................................................... 66
Figure 6-9 Integration Model functional flow diagram ............................................................................. 67
Figure 6-10 Integration Model constituent systems to functions mapping diagram .................................. 67
Figure 6-11 Integration Model Constituent systems to functions mapping table ...................................... 68
Figure 6-12 Integration Model pattern definition example ....................................................................... 68
Figure 6-13 Predictive Run Time Optimization for System of Systems Model ........................................... 69
Figure 6-14 Predictive Run Time Optimization Incubator .......................................................................... 70

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Date</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>Final</td>
<td>2013-04-25</td>
<td>5 of 72</td>
</tr>
</tbody>
</table>
List of Tables

Table 3-1: CAE Constituent Systems ................................................................. 21
Table 5-1: City District Attributes .................................................................... 44
Table 5-2: Fire Station attributes ...................................................................... 46
Table 5-3: Fire Fighting Car attributes .............................................................. 47
Table 5-4: Fireman model parameters ............................................................... 50
Table 5-5 Information Quality Attributes ........................................................ 56
Table 5-6: Goal formalisation using GCSSL approach ....................................... 57
## Abbreviations and Definitions

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>Command and Control</td>
</tr>
<tr>
<td>C4I</td>
<td>Command, Control, Communications, Computer and Intelligence (also: Information)</td>
</tr>
<tr>
<td>CAE</td>
<td>Concept Alignment Example</td>
</tr>
<tr>
<td>CS</td>
<td>Constituent System</td>
</tr>
<tr>
<td>DANSE</td>
<td>Designing for Adaptability and evolution in System of systems</td>
</tr>
<tr>
<td>DODAF</td>
<td>Department of Defence Architecture Framework</td>
</tr>
<tr>
<td>ER</td>
<td>Emergency Response</td>
</tr>
<tr>
<td>GCSL</td>
<td>Goal and Contract Specification Language</td>
</tr>
<tr>
<td>Groove</td>
<td></td>
</tr>
<tr>
<td>LSI</td>
<td>Large System Integrator</td>
</tr>
<tr>
<td>LTE</td>
<td>Long Term Evolution</td>
</tr>
<tr>
<td>MODAF</td>
<td>Ministry of Defence Architecture Framework</td>
</tr>
<tr>
<td>NAF</td>
<td>NATO Architecture Framework</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Modelling Group</td>
</tr>
<tr>
<td>OV</td>
<td>Operational View</td>
</tr>
<tr>
<td>SoS</td>
<td>System of Systems</td>
</tr>
<tr>
<td>SoSE</td>
<td>System of Systems Engineering</td>
</tr>
<tr>
<td>SV</td>
<td>System View</td>
</tr>
<tr>
<td>SysML</td>
<td>System Modelling Language</td>
</tr>
<tr>
<td>TETRA</td>
<td>Terrestrial Trunked Radio</td>
</tr>
<tr>
<td>UPDM</td>
<td>Unified Profile for DODAF and MODAF</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Purpose of the document

The purpose of the document is to provide a detailed description of the Concept Alignment Example (CAE) for the DANSE project.

The CAE shall be a playground to demonstrate new methods and models for analysis and visualization of SoS design. It shall allow featuring existing "views", e.g. using the Unified Profile for DoDAF and MODAF (UPDM), but also evolving "views" (as a result of proposed new methods).

Furthermore, the CAE shall offer an opportunity to analyse and visualize existing behaviours in a SoS as well as to evaluate (predict) emergent behaviours related to evolving developments in the operational, human and technical context. In general the concept validation example shall be used to illustrate the integration of heterogeneous legacy systems as well as evolving systems and stakeholders that need to work together according to an "SoS" architecture.

For the development of the CAE it was envisaged by the project partners to focus on a reference SoS that helps to better illustrate the business needs and the challenges to overcome in present SoS Engineering methods, practices and technologies. The DANSE Consortium agreed to choose an exemplified Emergency Response SoS to serve as the “playground” for the development of new SoSE methods and technologies. Accordingly this deliverable documents in brief relevant examples, models, views and data of the Emergency Response SoS, which shall serve as a direct input to DANSE technology partners (WP4-WP8) in order to apply the DANSE methods and technologies and use it for illustration and demonstrations.

1.2 Inputs used for the work

The Emergency Response SoS has been a frequently investigated area in the SoS context. Therefore, many public references exist, e.g. RD2, which serve as input to the project and to the definition of the CAE. Nevertheless, the presented examples were still also highly influenced by interviews with internal and external experts by DANSE industrial partners. Most focus was put on the attempt to describe examples and issues, which stand in direct relation to current DANSE SoS Engineering Challenges and envisaged new SoSE methods and technologies. At the same time the selected examples should be used in representation of additional SoS Engineering challenges that could be found in other industrial application domains such as Air Traffic Management, Integrated Water Management and Supply, or Autonomous Ground Control (DANSE Test Cases).

As a consequence the examples given and models developed do not claim to be completely realistic or to reflect a real SoS Engineering Case of an Emergency Response SoS. Nevertheless, expert interviews e.g. with DANSE partners’ industrial experts experienced in the field of C4I and Communication Systems have provided a significant basis for the work presented.

1.3 Addressed Readers

The addressed readers of this document are system designers, modellers, solutions architects and managers that are interested in SoS examples used in the DANSE project to investigate new methods and technologies for SoS Architecture Definition, Analyses and Simulation.

Internally to the DANSE project, this document shall be the reference for WP4-WP8 applying DANSE methods and technologies on the presented examples.
1.4 Structure of the document

The document is structured into 6 main chapters:

Following the introduction, Chapter 2 presents an overview of the CAE scope and work performed in terms of CAE modelling; the relation to the DANSE requirements is explained as well.

Chapter 3 describes concretely exemplified SoS modelling challenges and provides a detailed summary of SoS Architecture and models that have been built using a classical Architecture Framework approach.

Chapter 4 specifically describes challenges and examples to support the design exploration on SoS level. Different aspects and views of how to explore Architecture Alternatives are presented.

Complementary to a rather static representation and analyses of SoS architectures, Chapter 5 investigates more deeply on examples targeting the analyses and simulation of the behaviour of the SoS at run time.

DANSE Technologies integration and runtime complementary approach are presented in Chapter 6. The technologies flow is implemented through a detailed CAE UPDM model enhanced with technical extensions.

Chapter 7 provides a conclusion for the definition of the examples given in this document and provides an outlook on the further use or evolvement of the presented material.
2 Concept Alignment Example (CAE) description

The objective of the CAE in DANSE is to help the DANSE research partners focus on the multi-lateral elements of SoS Engineering, starting at the very beginning of the project as a support for requirement elicitation for DANSE, then as a platform to focus all technology developing partners in a coordinated direction and later as a demonstrator for public dissemination of the project results in meetings, conferences and wide reaching publications.

In the following sections, a principle overview of the reference SoS will be presented that has driven the development of different models as input to the DANSE methods and technologies. Furthermore, the relationship of the CAE to the DANSE requirements will be introduced. Yet it shall be noted that the actual DANSE Requirements will not be presented as they are documented within a DANSE internal deliverable.

2.1 CAE Overview

In order to develop DANSE methods and technologies in the context of a realistic SoS with realistic SoS Challenges, the DANSE Consortium has chosen to consider a widely used reference example of SoS, namely the Emergency Response System of Systems (ER SoS). As there are different methods, practices and technologies existing to support SoS Engineering it also became apparent early in the project, that the CAE would need to be supported by various models, build upon various modelling languages and standards.

In the following sub-sections the driving context of the ER SoS and its associated models produced for the first validation iteration of DANSE methods and technologies are presented.

2.1.1 The Emergency Response System of Systems

The Emergency Response SoS can be viewed from various perspectives. It has been chosen for the DANSE project as all of the DANSE challenges can be addressed in the context of this SoS.

The rationale for the ER System considered as being an SoS can be explained by the following characteristics:

The ER SoS consists of a non-specified number of individual Constituent Systems, all having their own life cycle, partially with managerial independence. Constituent System can be understood as the classical definition of a system as defined by e.g. INCOSE\(^1\). In the ER context constituent systems could be represented by organizations (Government, Fire Brigade, Police, etc.) or technical systems (C4I System, Communication System, Fire Cars, etc.) and humans (operators, decision-makers, fire fighters, etc.).

In addition, the ER SoS is subject to dynamic aspects and evolution over different time scales. In order to provide a starting point for DANSE we take the following assumptions about the context of an ER SoS in an exemplified large city in Western Europe:

- The population of a prosperous city increases over several decades

As a consequence:

- The city infrastructure needs to be adapted to city growth
  - New buildings, roads and crossroads are created
- City road traffic increases
- New public services for emergency response are created
  - New fire, police and health care department stations are built or moved (more stations in order to serve smaller city areas)
  - More fire, police and health care department units are allocated
  - New C4I command & control organization & communication system

---

\(^1\) INCOSE – International Council on Systems Engineering (www.incose.org)
Improved Emergency response performance in terms of response time to emergency call and situational awareness

The following illustration highlights that the indicated aspects influence the ER SoS at different time scales. The evolution and dynamicity of the ER SoS over different time scales causes significant SoS Engineering Challenges, which the DANSE project addresses.

![Figure 2-1: Emergency Response SoS Dynamicity aspects and SoSE Challenges](image)

The Challenges in SoS Engineering can be broken down into the difficulty to effectively 'model' the SoS considering the indicated dynamic and evolutionary aspects. The modelling issue is directly followed by the challenge to explore and select the best SoS Architecture among the large number of possible alternative configurations of the SoS. Complementary to the Design Exploration Challenge there is also a need to understand the (emergent) behaviour of the SoS at run time, as some effects of constituent system interaction and total SoS performance can only be analysed as the SoS operates. DANSE methods and technologies will support this aspect e.g. with innovative capabilities to simulate the SoS Architecture.
2.1.2 CAE Models Overview

As stated in the previous chapter, the SoSE challenges as well as the proposed DANSE methods and technologies required an examination of the current SoS models and data, necessary to elaborate further on a new SoSE Methodology.

Concretely it was found that as a starting point the CAE would need to provide a reference model developed based on a widely used SoS Architecting approach using an SoS Architecture Framework. For DANSE we have chosen to use the Unified Profile of DoDAF and MoDAF (UPDM) by the OMG as it presents a Standard in this area. In addition, we had to make a choice for a tool to support the modelling in order to make these models accessible and deployable by other DANSE technologies. IBM, as a core DANSE project partner, provided Rational Rhapsody v8 as the reference modelling tool to support the development of that model. Thereby, it was possible to provide a first reference of "as-is" practices in SoS Modelling and associated Architecture Frameworks. Accordingly several views of the ER SoS could be provided and are documented in more detail in Chapter 3.3.

In the course of several workshops with DANSE Technology partners, it soon became very clear that the today's challenges in SoS Engineering and the DANSE new methods and technologies could not directly be addressed and neither illustrated by keeping only one modelling profile. A major reason resides in the fact that the change or enhancement of existing standard profiles such as UPDM, and associated implementations in a commercial tool, would be a too great challenge for a research project as DANSE for the first year. Therefore, a decision was made to also develop models based on other tool profiles such as e.g. the pure SysML profile used within IBM Rational Rhapsody, as there seemed to be more extensions and capabilities available in this area to support a preliminary illustration of DANSE methods and technologies. The remaining effort to be taken in the project is to understand how to apply methods and technologies also on classical SoS Architecture Frameworks and profiles.

The Figure 2-2 illustrates the breakdown of CAE models that were developed to support various activities to develop the DANSE methodology. It can be seen that starting from classical SoS modelling using UPDM, several other models have been developed e.g. in pure SysML, or Graph Grammar environments (GROOVE) to investigate more closer on Architecture Optimisation Techniques (also see Chapter 4.2), Design Patterns (see Chapter 4.1) and models to support predicted run time analysis. The idea of the CAE behavioural Model was to focus on a subset of the ER SoS in order to provide a small set of models that provide e.g. a detailed characterisation of the constituent systems behaviour and their abstraction in order to serve as effective input to the DANSE technology development. One major reason behind is the fact that the development of a large set of detailed models of the whole ER SoS would simply require too many resources and would significantly slow down the innovation process within DANSE.

![Figure 2-2: CAE breakdown Overview](image-url)
2.2 CAE versus DANSE Requirements

The Concept Alignment Example contributes to the elicitation of end user needs and expectations regarding Systems of Systems engineering. In the DANSE project, end-users of engineering methods and tools are represented by EADS, IAI, Carmeq and Thales. Their expectations are formally expressed in DANSE through a specific document: DANSE Deliverable D3.1 – Project Requirements (RD4).

The methodology adopted for requirements management in DANSE is inspired by the approach generally adopted in the industry:

- A set of user needs is initially identified. In DANSE, this is expressed in the project Statement of Work, and has been updated right from the DANSE kick-off meeting. User needs are not expressed in terms of technical solutions, hence are not readily usable by technology providers.
- Consistently with these user needs, we need to derive an expression which has an added value for the relevant technology providers, in terms of orientation of research and technology development.
- Common to all requirements, an architectural view of the system to be developed is supposed to be agreed among stakeholders. In DANSE, the “system to be developed” is the engineering environment (methods and tools) to develop Systems of Systems. Its components, which are the topic for the requirements, are DANSE technical solutions.
- Each requirement should refer to the engineering environment as a whole, or to one of its functional or technological components. For example, requirements may be of the form
  - “DANSE simulation tools shall be able to ....” or
  - “DANSE methodology shall support ....” or
  - “DANSE SoS metamodel shall include ...”

The figure below summarizes the role of Requirements, as a link between user needs and DANSE technical solutions.

![Diagram](image)

Figure 2-3: DANSE Requirements

There is not at the moment an agreed view on all SoS concepts. This was recognized since the beginning of DANSE as a potential difficulty for this “top-down” requirements elicitation process, as it would not make sense to be very rigorous in the requirement wording if some terminology remains ambiguous. Also, not all technology providers are familiar with the terminology used in systems engineering.

It was therefore strongly recommended that some requirements are illustrated by examples, when needed. These examples can be taken from publicly available data, from the background of each company or from the joint open example presented in this document.

At the time this final version of D3.3 is produced, DANSE requirements database includes 76 requirements. Most of these requirements should ultimately benefit from illustrations through examples.
At the moment, about 63% of these are actually referring to the CAE as concrete illustration (see RD11). Lack of coverage of requirements by the CAE is motivating improvements of the CAE. Conversely, elaborating the CAE may suggest complementary requirements to be formulated. Illustrations through examples are a major contribution to the understanding of user needs by technology providers. Some may also suggest a means to verify the effectiveness of DANSE solutions. It must be noted though that not all Requirements may be able to be illustrated by the CAE alone as the CAE is focussed on providing an example of applying DANSE methods and technologies on the Emergency Response SoS. E.g. Requirements towards new capabilities in SoS Engineering and Modelling may also be directly addressed by the DANSE Methodology (WP4).

In DANSE, IAI, Carmeq and Thales are bringing test cases from their respective business areas. These test cases may contribute to the illustrations and examples. However, illustration through the CAE is preferred when it would be a problem to disclose some specificity of the test cases – while the CAE is shared.

A specific role of the test cases is to put DANSE solutions to the test, give feedback to the technology providers, and ultimately assess DANSE solutions. This is one primary goal of the test cases. In addition, the CAE may contribute to training end users and to putting DANSE tools to the test (before deployment on the use case) and may be a place for investigating issues – hence contributing to the improvement of tools, and not only to illustrations and demonstrations.

DANSE requirements document is elaborated in a phased approach. The final version will include updated references to the CAE for illustrations, and will make explicit, for each requirement, which test case(s) is (are) involved in assessing the implementation of this requirement.
3 CAE – Modelling the SoS Architecture

This chapter elaborates on the issues and examples concerning the challenges in modelling the context and structural aspects of an SoS (in an evolving environment).

3.1 Challenges of an exemplified Emergency Response SoS

Based on the initial assumption for the ER SoS, the following mission definition and desired changes to the ER SoS shall represent an exemplified set of capabilities and objectives for the ER SoS of our reference city. Please note that the missions as well as the desired changes are - in real life – motivated by several stakeholders of the ER SoS. This means, that on a lower level, these objectives and the individual interpretation of the required capabilities or services could differ significantly. DANSE methods and technologies are aiming to provide capabilities to resolve this issue.

The Emergency Response SoS Mission Definition:

- Provide ER Services that meet the challenges of the next decade
  - Increasing density of population in urban (multi-cultural) areas
  - Increasing city road traffic
  - Provide the ability to quickly respond to complex, unforeseen and ambiguous threats (e.g. by terrorism or natural catastrophes)
- Maintain ER SoS at highest level of available technology balancing cost, reliability & effectiveness
- Manage the evolution and integration of new ER technologies

Desired Changes to improve the ER SoS in the short/mid/long term future:

- Increase overall effectiveness of Emergency Response services
  - Provide enhanced situational awareness in less time
  - Reduce operational Response Time to emergencies
  - Ensure ER communication network coverage in urban and rural areas
  - Ensure optimal distribution and availability of ER Resources
  - Ensure reliability and survivability of ER SoS, reducing negative emergent behaviour of the total system (early detect emergent behaviour)
- Provide adaptive ER infrastructure to integrate e.g.
  - Next generation of C4I Systems,
  - New Communication Networks (LTE)
  - New Fire Fighting Equipment, etc.
  - Improve collaboration of industry/ technology partners
- Ensure interoperability of ER Systems

These examples will be used to drive further elaboration of e.g. individual constituent system goals, formalization techniques, architecture optimization and trade off objectives and simulation constraints & scenarios.
3.2 Conflicting Goals

In general, each system has its own individual goals to fulfil. Most SoS are constructed as combination of pre-existing autonomously operating constituent systems that result in one system with a defined set of higher level goals and strategies. Once a constituent system decides to join the SoS, it may delegates some of its authority to the SoS. This process may require from the constituent system to merge some of its goals with the SoS global goals to serve the SoS overall functionality. As a consequence, these goals of the constituent system that can be considered as common goals need to be identified, plus the goals that must be adjusted to match the SoS global goals, and, finally, the goals that could create conflict.

Dealing with an SoS environment makes the goals definition and understanding more complex for the following reasons:

- Different stakeholders for different constituent systems: each constituent system has its own stakeholders that differ from the other systems according to its scope of work and applications, these stakeholders have various interests that are reflected in the formation of the system goals, which ends up in a divergence of the goals variability within the overall SoS goals environment.
- Systems life cycle and periodic goals: the SoS and its constituent systems can be viewed from different timing scales, each having its own periodic strategies and goals. The strategic planning can be different when focussing on e.g. decades, years, hours or minutes. As the constituent systems are from diverse disciplines and developed for different purposes, their life cycles and strategies are distinct. E.g. organisational goals may focus on long term goals, whereas technical goals may focus on mid-term strategies, as their stakeholders e.g. the government or respectively the technology provider have different goals and interest.
- Systems evolution: the systems evolve over time as well as their environment, and their objectives may be changed accordingly. This dynamism affects the goals definition; some goals will be changed, new goals will be defined, and others will be eliminated.
- Various perspectives: each of the constituent systems contributes to the overall SoS goals from different viewpoints, e.g. operational, technological, economical, etc.

Some examples of ER SoS stakeholders and their individual goals are presented hereafter:

- Local Government:
  - “Minimize mean number of injured people due to fire per year” (operational/ long term)
  - “Minimize cost for ER infrastructure, while achieving the operational objectives” (economical vs. operational/ mid-term)
  - “Ensure survivability of ER SoS” (operational & technical/ short term)

- Fire Brigade:
  - “Minimize mean time for Fire HQ orders to reach their fire station” (operational and/or technical/ short term)
  - “Minimize mean time of a district staying under fire” (operational/ short term)
  - “Minimize mean time for the fire fighting cars to reach a district, provided that no fire station is empty” (operational/ short term)

- Technology Provider:
  - “Develop next generation best in class C4I System” (technical & economical/ mid-term)
  - “Become Market leader for Digital Communication System” (technical & economical/ mid-term)

It could be easy to verify constituent systems’ goals if they would work alone but this is not the case in an SoS context. The emergent behaviour of the SoS plays a great role in the SoS behaviour as the emergent behaviour cannot be expected and can be good or bad; this makes the verification process for the SoS goals more complicated.
The concept of conflicting goals is about the opposition that can appear between global goals of the SoS and local goals at constituent system level. Most global SoS goals are likely to be behavioural, in the sense that they will require simulation to verify their achievement. As it will be explained in detail in chapter 5, simulation requires that a behavioural model is available for each involved constituent system. It is not necessary to express the local goals explicitly to run simulation and verify the global goals, since the constituent system behaviours should already reflect their own goals. In the frame of the previous example, this means that the behavioural model of the fireman will typically include a small probability for him to disobey an order when he thinks that his life is too much at risk, maybe resulting in the failure of the global goal of minimizing the total number of deaths (among firemen and people in danger).

Eliciting the local goals can still be valuable information to make sure that the constituent system behaviours are indeed in-line with their goal. So, when using simulation (possibly combined with statistical Model Checking) to verify the local/global goals:

- If local and global goals are all verified, then the whole SoS behaves as expected
- If only local goals are not verified, then the behavioural modelling of the constituent systems must be corrected to match their goal as they should
- If only global goals are not verified, then the architecture of the SoS must be improved until they are (e.g. replacing/adding/removing constituent systems or the connections between them)
- If some local and global goals are not verified, it is a mix of the two previous problems

In chapter about SoS optimization (chapter 4) we describe a network coverage scenario, highlighting that the conflicting goals have great effect on the optimization process. Both global and local goals are considered during the optimization process. The developer has to arrange the priority of goals that must be considered and he must define their combination formula.

Conflicting goals could be also detected among the constituent systems’ local goals themselves. The problem in this case is that the systems could work against each other, which results in bad effects on the SoS overall behaviour. For such a problem the SoS management has to consider the goals for each constituent system in its SoS planning and needs to point out the conflicting goals that need special attention.

In the CAE we have global goals for the Emergency Response SoS, and local goals for constituent systems, for example one of the overall goals for the SoS is to “Minimize mean number of injured people due to fire per year”, implicating that the firemen are committed to save these people. This matches the SoS goals but not the fireman’s individual goal “protect own life”. This case can be monitored and controlled in a better way in a simulation process where conflicting goals can be discovered according to the systems behaviour during the simulation. The CAE behavioural part as introduced in chapter 2.1.2 defines such behaviour in order to detect the emergent behaviour of such conflict.

In defining the optimization goals for the CAE, different conflicting goals could be described. For the ER SoS, the overall goal is to “Minimize cost for ER infrastructure, while achieving the operational objectives", which implies that we have always to look for the minimum cost for ER infrastructure that includes the communication network. However, for technical and service providing systems the goals are (as mentioned above):

- “Develop next generation best in class C4I System”
- “Become Market leader for Digital Communication System”, which means:
  o Achieve the best coverage area that serve the emergency response system
  o Lower the cost: Procurement, maintenance and running cost
  o Provide the best communication quality

In order to achieve the previous mention goals the technical and services systems may require changing the communication system to a new system. This implies a change of the communication infrastructure, which again adds new cost on the ER SoS, in contradiction to the SoS overall goals of minimizing the cost. For such a case in the optimization process, all the goals are defined and the optimization is directed according to the priority definition of SoS goals and the SoS leader interest.
3.3 SoS Modelling using SoS Architecture Frameworks

The SoS modelling process can be separated into the following steps:

- Define the modelling purpose
- Define and analyse the SoS constituent systems
- Chose the Modelling Language
- Build the SoS model

3.3.1 Define the modelling purpose

The first step in the modelling of an SoS starts with answering the questions:

- Why do we want to build the model?
- What do we want to investigate?
- What information do we want to share between stakeholders?

The purpose of modelling the CAE is to meet the modelling challenges listed in section 3.1. It is needed to have a model that depicts the core goals of the ER SoS and shares the same information about its constituent systems and their behaviour. It forms the basic model at which the stakeholders’ technologies can be applied. Therefore, the model is built in a way that explicitly defines all the information regarding the constituent system, their connections, operations, and functions. It also defines the communication network between these systems.

3.3.2 Define and analyse the SoS constituent systems

After defining the modelling purpose, the next step is to analyse the current SoS (if already existing) and its constituent systems, which can be done by analysing:

- the scope of the chosen SoS, its boundaries and fields of operation
- the type of the investigated system (i.e. directed, acknowledged or collaborative)
- SoS constituent systems
- SOS constituent systems types (i.e. human systems, technical systems, services system)
- the scope for each constituent system
- constituent systems components and their connections
- constituent systems operations
- the operations flow
- different connection possibilities between constituent systems
- constituent systems functions
- constituent systems communication channels
- services systems and services taxonomy
- human systems organizations.

Table 3-1 shows the CAE constituent systems and their role.

Abbreviations:

- PCC= Police Command and Control
- C2= Command and Control
- LSI= Local Surveillance Infrastructure
- LCI= Local Communication Infrastructure
- FCC= Fire Command and Control
- WaterCC=Water Command and Control
- GO = Governmental Organization
- EX = External Organization

<table>
<thead>
<tr>
<th>Component System</th>
<th>Owner</th>
<th>Controlled by</th>
<th>Description</th>
<th>Objective</th>
<th>Services</th>
<th>Status</th>
<th>Date</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Police Resource tracking System</td>
<td>PCC</td>
<td>LSI</td>
<td>The system provides all the facilities and means used to track the police resources which facilitates dispatching and enhances the response time.</td>
<td>Define exact position for each resource connected to the system.</td>
<td>• Provide the exact position on the map.</td>
<td>2.0</td>
<td>Final</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Define the drive direction</td>
<td></td>
<td>2013-04-25</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Define the surroundings</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Police Communication System</td>
<td>PCC</td>
<td>LCI</td>
<td>The system provides equipment and means that serves the different communication channels used by the Police members to achieve their tasks.</td>
<td>Enhance data exchange between different police systems parts, and police system parts with external organizations.</td>
<td>• Data information Exchange.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Voice communication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Transfer orders</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traffic Surveillance System</td>
<td>PCC</td>
<td>LSI</td>
<td>This system monitors different parts of traffic system.</td>
<td>Help and Direct police members to the best traffic plan to achieve their tasks.</td>
<td>• Point traffic problems</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Report traffic status</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• show best traffic usability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Police Resources</td>
<td>PCC</td>
<td>C2</td>
<td>The system responsible for managing and provide the required resources and tasks implementation.</td>
<td>Provide best resource supply services and task achievements</td>
<td>• Manage different police resources.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Implement dispatch orders</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Report resource availability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Achieve required tasks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fire Brigade Communication System</td>
<td>FCC</td>
<td>LCI</td>
<td>The system provides equipment and means that serves different communication channels used by the fire brigade members to achieve their tasks.</td>
<td>Enhance data exchange between different fire systems parts, and fire system parts with external organizations.</td>
<td>• Data information Exchange.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Voice communication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Transfer orders</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fire Brigade Resources</td>
<td>FCC</td>
<td>C2</td>
<td>The system responsible for managing and provide the required resources and tasks implementation.</td>
<td>Provide best resource supply services and task achievements</td>
<td>• Manage different police resources.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Implement dispatch orders</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Report resource availability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Achieve required tasks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Site Monitoring System</td>
<td>C2</td>
<td>LSI</td>
<td>The system collects information about the emergency site and reports for the required additional resources as well as required actions.</td>
<td>Supporting emergency response and controlling the site operations.</td>
<td>• Collect and distribute site information</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Perform additional dispatches</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Direct different resources to the required actions.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Threat Detection and Allocate System</td>
<td>C2</td>
<td>C2</td>
<td>The system responsible for detect threat possibilities and define its locations and sources.</td>
<td>Prevent or mitigate emergency propagation by provide threat analysis and earlier detection.</td>
<td>• Current situation analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Alert possible threats</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Recommend actions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource Dispatching System</td>
<td>C2</td>
<td>C2</td>
<td>The system responsible for the first response dispatch and resource allocation.</td>
<td>Reduce emergency effect by providing planned activities with the required resources</td>
<td>• Define the emergency classification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Assign the required activities</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Assign the required resources and their availability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component System</td>
<td>Owner</td>
<td>Controlled by</td>
<td>Description</td>
<td>Objective</td>
<td>Services</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Call handling system</td>
<td>C2</td>
<td>C2</td>
<td>The system collects and provide the available information from the emergency reporters</td>
<td>Organize and supply the emergency cases information</td>
<td>● Issue dispatch orders</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2 Communication System</td>
<td>C2</td>
<td>LCI</td>
<td>The system provides equipment and means that serve different communication channels used by the C2 team members to achieve their tasks.</td>
<td>Enhance data exchange between different C2 systems parts, and C2 system parts with external organizations and emergency response members</td>
<td>● Data and information Exchange.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Water Monitoring system</td>
<td>WCC</td>
<td>LSI</td>
<td>The system responsible for monitoring water resources and possible water pollution threats</td>
<td>Identify and notify water pollution case</td>
<td>● Water resources analysis.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TettraPol Communication</td>
<td>LCI</td>
<td>LCI</td>
<td>The system supplies Tetra Communication services for different communication systems</td>
<td>Facilitate data and information exchange between different parties</td>
<td>● Speech Transfer (Direct Mode, Late Entry, Priority Call, open Channel, Include Call, Emergency Call, Broadcast Call, individual Call and Group Call)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VHF/UHF Communication</td>
<td>LCI</td>
<td>LCI</td>
<td>The system consists of technical facilities that provide the UHF/VHF communication services</td>
<td>Facilitate system parts communication</td>
<td>● Radio Communication.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cell Phone</td>
<td>LCI</td>
<td>LCI</td>
<td>Communication stations and towers that provide cell phone services</td>
<td>Facilitate system parts communication</td>
<td>● Speech Communication.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Land Line Phone</td>
<td>LCI</td>
<td>LCI</td>
<td>Communication stations that supply the land phone services</td>
<td>Facilitate system parts communication and different systems communication</td>
<td>● Speech communication.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Satellites</td>
<td>LCI</td>
<td>LCI</td>
<td>The system consists of technical facilities and stations that provides surveillance services</td>
<td>Provide live view and information of targeted object</td>
<td>● Pictures.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Navigation</td>
<td>LCI</td>
<td>LCI</td>
<td>The system consists of technical facilities and stations that provides navigation services</td>
<td>Provide information about targeted object position</td>
<td>● Define position.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Communication</td>
<td>EO</td>
<td>LCI</td>
<td>The system responsible of communication services in the external organization</td>
<td>Facilitate Data and information exchange between the system and other systems</td>
<td>● Speech communication.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Catastrophe and Emergency center</td>
<td>GO</td>
<td>GO</td>
<td>The system responsible for the whole Emergency response SOS</td>
<td>Direct and Control the Emergency Response SOS</td>
<td>● Control the ER Operations.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Version | Status | Date     | Page |
---------|--------|----------|------|
2.0      | Final  | 2013-04-25 | 20 of 72 |
### 3.3.3 Choose Modelling Language

The “language” and associated Architecture Framework used for SoS modelling for the CAE is the Unified Profile for DoDAF and MoDAF. For the CAE it was decided to use the UPDM DoDAF implementation in IBM Rational Rhapsody V8.0. UPDM was developed by the Object Management Group (OMG) to support the USA Department of Defence Architecture Framework (DoDAF) and the UK Ministry of Defence Architecture Framework (MODAF).

With UPDM it is possible to create different views that show different aspects of the SoS. The views serve the scope of interest of different stakeholders of the SoS and try to depict the overall SoS at an abstracted level.

### 3.3.4 Build the SoS model

To build the UPDM models of the ER SoS we have to follow the flow of the Architecture Framework views and their relations. Figure 3-1 illustrates the workflow for building the SoS model.

The process starts with understanding the main concept of the SoS and its purpose, which is done by building the ER SoS Operational View OV-1, that depicts the overall picture of the Emergency Response operational context. It includes all the operational nodes in the ER SoS and a general description of their contribution in the SoS.

After defining the main concept of the ER SoS, the second step is to build the Operational Activity Model (OV-5). From OV-1, the major Emergency Response operational nodes can be defined, which will be used in OV-5 that describes the operational activities flow between these nodes.

There are two approaches to build the OV-5:

- In the *scenario-based* approach a certain scenario is created, the OV-5 describes the activities flow of such a scenario.
- In the *general* approach the general activities of the operations are described and their relations.

Usually SoS Architects may need multiple OV-5 views to show different operational flows within the SoS operations. In our model we have used both the scenario based and the general approach to describe the ER SoS operational activities.

Based on the OV-5 view the connections between the nodes can be defined, and thus, the Operational node Connectivity Description view (OV-2) can be built. OV-2 describes the connections and data exchange between the ER SoS operational nodes. After this step, the Operational Information Requirements (OV-3) diagram that contains all the information about the connections described in OV-2 can often automatically be generated in existing modelling tools as e.g. Rhapsody.

Parallel to the previous step, the ER Systems Functionality (SV-4) view could now be constructed. It gives a clear description to data flow and functional allocation of the ER Systems. This view forms the basis for constructing the System view (SV-1). SV-1 represents the description of the SoS as a collection of constituent systems and their connections, relations, communications channels, interfaces, dependencies and other aspects.
The views mentioned above present the main baseline of constructed UPDM views for our ER SoS; other views could be helpful of course and would need to be built depending on the SoS aspects that shall be highlighted. For the DANSE CAE, presently these UPDM views were considered to be sufficient to support further work on the DANSE methods and technologies.

![Figure 3-1: Process for building the ER SoS UPDM model](image)

In the following sub-sections a general illustration for each UPDM view of the ER SoS, its contents and links will be shown.
3.3.4.1 High level Operational View Description (OV-1)

Figure 3-2 depicts the overall operational concept and mission of our Emergency Response SoS. The operational nodes that contribute to the ER SoS are defined and a general view about their principal operations is provided. As mentioned previously it is the first step for building the model. The figure presents a high level operational scenario in which a major European City (e.g. Berlin) is exposed to a major emergency/incident. It is shown that different operational emergency nodes are working together during the emergency case:

The Fire Brigades are distributed across the city and work jointly with the police stations. Each of them may be controlled by individual Command and Control Centres (CCC) in order to mitigate the emergency/incident. Of course a lot of other ER Resources are also involved, e.g. the hospitals providing ambulances that try to evacuate and rescue injured people, the Water Command and Control that controls the local water resources, the surveillance systems that send information about the emergency area and overall situation to the CCCs, mobile nodes such as a Threat detection and Alert System that detects further information or possible danger close to the emergency scene and finally different types of communication systems that are involved to facilitate the communication between different operational nodes in the ER SoS.

3.3.4.2 Operation activity model (OV-5)

This Operation activity model presents the major operational activities, their flow and the responsibility of different operational nodes. For our alignment example we have built multiple OV-5 models that show different perspectives of the ER SoS:

- Manage Incident
  The manage incident view describes the required operational activities and their flow, which is followed for dealing with emergency case.

- Manage Operational Data
  The manage operation data view illustrates the data management activities for the emergency case.

- Receive Emergency Call
  The receive emergency call depicts the activity flow for dealing with emergency call.
- Resources Management
  The Resource Management shows activity connections that are used to manage the resources from different systems.
- House Fire Scenario
  The House Fire Scenario was built in order to present the flow of general emergency systems activities for a specific scenario.

We will consider the House Fire Scenario as an example of OV-5 views, as shown in Figure 3-3.

![Figure 3-3: House Fire Scenario OV-5](image-url)
The operational nodes are distributed at the head of the diagram, and the activities are arranged in the swim lanes of each operational node. The links represent the flow of the activities and their arrangement and dependencies. It also shows the data exchange between different nodes. Each rounded rectangular is an operational activity, the diamond is a decision gate at which a decision must be taken. According to that decision the flow will continue either to the right or left side. The grey circle is the starting point while the blue circle presents the end point of the flow.

### 3.3.4.3 Operational node connectivity description (OV-2)

The operational node connectivity view is used to describe the connections between the nodes that were indicated in OV-5. As shown in Figure 3-4 each node is represented as a rectangular box. The dashed line indicates the information exchange between two different nodes, which means that both nodes are connected to each other and they exchange data and/ or information. The connections also can include “needlines” that represent the dependencies between the nodes.

![Figure 3-4: CAE Operational node connectivity description OV-2](image-url)
3.3.4.4 Operational Information Requirements View (OV-3)

This Operational Information Requirements View is automatically generated after building the OV-2 (see Figure 3-5).

![Figure 3-5: CAE Operational Information Requirement OV-3](image)

It majorly shows a table that summarizes the information exchange between different nodes, who will be in contact with whom and which kind of elements are exchanged.

3.3.4.5 System Functionality view (SV-4)

The System Functionality View is the basis for developing the System view SV-1 (sub-section 3.3.4.6). It contains the relations between the operational nodes and the constituent systems, and relates the functions required for the ER SoS to the constituent systems.

There can be multiple views that describe the functional flow of the ER SoS. In our example we have built the views that we believe help us to identify the constituent systems and to better understand the ER SoS.

These views are:

- Receiving Call
- External Resource Despatching
- Monitoring Emergency Case
- Dispatching

As an example, we will proceed with the dispatching view (Figure 3-6) to show the important elements. It can be seen that the system nodes are on the top level of the diagram. The system node is a node where a group of systems work together in order to achieve a common goal. They are related at the same time to other systems that control and manage their operations. Each system node contains one or more systems, for example, the CCC node contains the call handling system, the dispatching system and others.

In SV-4 the system function is represented as a rectangular box and the links between the functions show the functional flow between the system nodes. The diamond is used to indicate a decision node where a decision has to be made in order to proceed with the functional flow, and the result will be either to proceed to the right of or the left link.

The blue rectangle is used to show the branches of the functional flow, in other words, the functions after the blue rectangular will be implemented in parallel.

The grey circle is the start of the flow, the blue circle indicates the end of the flow.
3.3.4.6 System View (SV-1)

The System View SV-1 is one of the most important views that illustrate the ER SoS and its constituent systems. It provides an overall picture of the ER SoS from a “system” perspective. It shows not only the constituent systems but also their connections, interactions, main components and interfaces.

This view was constructed after analysing the previous views. The constituent systems were classified according to their type:

- Technical Systems
- Human Systems
- Services Systems

Accordingly, for each type of constituent systems the way of representation and connections are different. For a detailed view the model would be very large and chaotic, and it would be difficult to understand. For this purpose, a high level abstraction was used to represent the constituent systems.

As defined before, the term “system node” has been used to indicate a group of constituent systems that work together for a certain purpose. The system node may have the full or partial control on its components or in some cases it may just uses the service provided through its components by another system node.

The system nodes are represented in a rectangular shape and denoted as ‘SystemPart, SystemNode’, while the systems are represented by rectangles denoted by ‘SystemPart, System’. Each of the systems has interfaces that link the system to other systems while the notations attached to the interface indicate to which system it will be linked. The systems are connected to each other through a link with the notation ‘DataExchange’ to indicate that these systems communicate and exchange data between each other. The arrows attached to the links define the direction of the data flow, which could be in one or both directions.

The links between services systems (i.e. Communication, Surveillance) are in pink and mean that a system uses the service from the service provider. The services connect to the systems through service interfaces that end up with half circle when the system is a client and circle when the system is a service provider. The notations next to the services interface indicate which services are connected to the interface, either provided or used by the system.
3.4 Collaboration in SoS Modelling

For our example of engineering an Emergency Response SoS, we also consider that the development of an SoS is a joint result of several technology providers working collaboratively on a common solution. Lead System Integrators often face the issue that collaboration with partners or suppliers becomes quite challenging as there are only limited available technologies to exchange information about their constituent system while securing intellectual property.

In fact, there are several reasons why models of the individual constituent systems cannot be joined together to construct a unified SoS model:

- Systems are from different disciplines
- Systems are separately developed
- Developers are from different contractors
- Developers use different modelling methodologies
- Not unified level of abstraction for systems models
- Different modelling purposes

---

Figure 3-7: CAE System View SV-1

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Date</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>Final</td>
<td>2013-04-25</td>
<td>28</td>
</tr>
</tbody>
</table>
Different modelling tools and environment.
Thus, in order to be able to construct the SoS models using the pre-existing constituent systems models while securing IP and without the need to re-model the constituent system again, there must be a methodology that allows to make an abstraction for these systems, and a flexible technology that allows the exchange of models developed in different tools.

In the ER SoS we have various systems with totally different purposes and disciplines. There are systems that depend on human efforts like the Police and the Fire brigade working together with technical tools to achieve their mission. For such systems we need human models that abstract their work and describe their behaviour and other models for the technical systems to describe their functions, inputs, outputs, and their behaviour. Other systems are the services systems that provide the communications and surveillance. These systems are modelled to define their technical aspects. The figure below illustrates the collaboration model:

Figure 3-8: Model Collaboration in CAE model

Figure 3-8 illustrates schematically the need to exchange, import or export various sub-sets of constituent system models as these could be owned by different stakeholders of the ER SoS. Within the frame of DANSE it is expected that future SoS tooling will provide capabilities to support these issues.
4 CAE – Modelling SoS Architecture Alternatives for Design Space Exploration

4.1 SoS Architectural Patterns

SoS architectural requirements typically have broad cross-functional implications. Such requirements are usually extracted and specified using quality sensitive scenario (RD5). When used appropriately, scenarios are very effective for specifying SoS architectural requirements because they are very flexible. For example, we can even use scenarios to represent i) failure modes in order to examine availability and reliability, ii) change requests to analyse modifiability, iii) threats to analyse security, or iv) ease of use to analyse usability. Consequently, SoS architecture design decisions are principally inspired by architecture requirements, which provide the criteria used to reason about and justify the architectural choices (RD6). SoS architects will typically consider several design options, which can potentially satisfy different non-functional requirements. From this, a decision is made based on several trade-off options from the available design options in order to satisfy all (or most) of the desired non-functional requirements. The architecture for the SoS is central to maintaining a persistent technical framework to guide the evolution of the SoS because it provides an integrated view of the ensemble of systems within the SoS. Approaches to the analysis of a SoS present very unique challenges because of the characteristics of the SoS. The SoS is in many ways a loose aggregation of other independent systems and there often is no overarching organization overseeing the system as a whole (RD7). To this end models are built that “define the critical acceptances requirements of the client and the overall structure of the system” (RD8). This means that highly detailed and abstract models are required for components on which success depends.

In this sense, a model does not have to be a software component, but can be any kind of technical or non-technical entity. Expanding on this definition, we use the term architecture pattern as an expression of the architectural structure as opposed to the traditional design patterns encountered in the software engineering community. The notion of the form of a pattern is essentially defined through its representation as a set of interacting components and their relationships. In other words, the form of a pattern consists of a finite number of visible and distinguishable components and their relationships meaning that a pattern has both structural and dynamic properties.

An architectural pattern can be considered as a framework in the sense that it provides a template for the structure and behaviour of an entire system within a domain (RD9). The system’s architecture model encapsulates decisions about the systems requirements, its logical elements and its physical elements. System in this context does not refer to just computer hardware and software (unlike earlier definitions), instead system is now defined in the general sense of an assembly of components that perform activities (these can be human, machine or services) and can be interacting or interdependent. Analysis of most system architectures will reveal many patterns. When applied correctly patterns play a huge part in architecting complex systems, they can be used to express common elements in a system design in a way that makes implementation easier. Also well-constructed patterns can be re-used in other places and make it easier to document and maintain existing systems through the use of a library or catalogue of patterns.

Any particular SoS can be represented as an amalgamation of models across these three layers (Figure 4-1) depending on the specific context of interest. An important approach to representing a SoS for analysis purposes is to consider expressing the SoS at systems architectural and systems design levels. The use of three pattern categories: Architectural, Interaction and Design (relating to the three-layered stack as shown in Figure 4-1) provides a clearer structure for any subsequent analysis of the SoS. In describing patterns it is important to recognise that patterns exist at different levels of abstraction. At the highest level (level 0) these patterns essentially describe a pattern category.
4.1.1 Centralized and decentralized Architecture

In our CAE model, we defined different kinds of systems nodes that contain different constituent systems, the way of controlling and managing the emergency case can be implemented using two different structures, Centralized and decentralized. Figure 4-2 shows the level 0 pattern for a generic command – response system. Although this describes a command-response system at a high level it does not yield sufficient information for a subsequent analysis of system performance trade-off. It will be seen that this high level pattern can be satisfied by various configurations (for illustrative purposes in the case of the figure a centralized architecture).

Applying context to this pattern we can derive a pattern for a centralised command - response system (ie Centralised command, control), comprising the constituent systems shown in Figure 4-3. In the centralized system we have Command and Control Centre (CCC) node that are connected to the other nodes and control their operations. The nodes delegate some of their authorities to the CCC node. CCC will be
responsible for managing the emergency notification, issue first response plan, issue and distributing the dispatch plan, and monitoring the emergency case.

Figure 4-3 Centralized System

Figure 4-3 shows an example of the generalized patterns for the centralized architecture.

In the case of decentralised system architecture the contextualised pattern is shown in Figure 4-4. It should be noted that there are many different alternatives to the centralised and decentralised architecture patterns.

In the decentralized system, each of the nodes has its full control, there is no CCC node and the nodes work together and exchange the report and information about the emergency case.

Each of the two structures has its own architecture description, for our CAE we built two models that represent the centralized and decentralized systems. Some of the views can be common for both of the architecture description (e.g. Service View, Operational View) and others are different (e.g. System view, Operation Activity Model view).

Figure 4-4: Decentralized System
**Example London Metropolitan Police**

The Major Incident response SoS for London, known as ‘The London Emergency Services Liaison Panel (LESLP)’ was formed in 1973 and consists of representatives from the Metropolitan Police Service, City of London Police, British Transport Police, the London Fire Brigade, the London Ambulance Service and local authorities. The Port of London Authority (PLA), Marine Coastguard, RAF, Military and voluntary sector are also represented. A recently published update to the London Emergency Planning Procedure (RD10) has been mined used to illustrate how patterns can be used to address interactions within this critical SoS. The architectural patterns show clearly how the different emergency services are supposed to work together and the sorts of interfaces that exist between the systems and operatives at a command and control level. Once these patterns have been mined it is then possible to look at alternative ways of providing a more efficient emergency response service (effectiveness in this case refers to reduced time to respond, decreased cost, greater availability, planning for future population/urban growth etc.). Scene management is clearly important and the general pattern for achieving this is shown in Figure 4-5 where carefully controlled cordon are established to protect the general public and allow the emergency services to operate without hindrance. At the heart of the operations is a Joint Emergency Services Control Centre (JESCC), which forms the focus from which the entire operations are managed. It has been possible to extract a pattern from (RD10) which provides details of the details of how a specific agency (emergency service) operates in such a scenario. The extracted architecture pattern (Command Relationships View OV-4) for the JESCC is shown in Figure 4-5.

![Figure 4-5: Architecture pattern for the JESCC](image)

In the diagram the term ‘agency’ refers to one or more of the emergency services. In the UK the ‘agency silver’ is responsible at the scene for all of that agency’s resources (tactical operations). It should be noted that the primary emergency service functions (fire, ambulance and police) are usually based at different geographical locations.

**Example ER Base Stations**

In SoS optimization patterns can be also used to describe different functional connections for the optimization problem. As shown in Figure 4-6, the 0 level patterns were developed to describe the optimization functional connection. The process starts with sending a message from a system and ends up in receiving the message within another system.
4.2 SoS Architecture Optimisation

SoS Architecture optimization is based on choosing the optimum combination of constituent systems components that achieve the optimization goals. The primary objectives of applying system optimization in the ER SoS context are:

- To choose the best combination of constituent system components that optimized against
  - Emergency response time
  - Emergency response quality
  - Cost of emergency operations
- To apply the system optimization to different CAE architecture descriptions and make a comparison between them.
- To plan for systems evolution through testing new components effect
- To plan for services system selection and compare the effect of use different services

4.2.1 Optimization Concept

In the ER SoS there are two architecture descriptions i.e. centralized and decentralized. We consider these two models as patterns for functional and operational flows and feed them to an optimization engine, together with the optimization goals and component systems specifications and properties to get at the end a proposal for an optimized system architecture candidate. Figure 4-7 illustrates the main concept of the ER SoS optimization:

1. Define the systems functions, and components that implement these functions with their different alternatives according to the optimization goals; some of the component's properties are needed (e.g. geographical distribution).
2. The next step is to set the functional flow patterns and to describe functional dependencies.
3. List the optimization goals that the architect needs to consider for the optimization. For example, one of the goals could be to minimize the overall cost, etc. Multiple goals could be defined with different constraints.
4.2.2 ER SoS - Communication System Optimization

As an optimization case, the communication coverage and network were considered as part of the overall ER SoS. The data exchange between the constituent systems depends on the communication links between them. Thus the communication services and quality affects the performance of the ER constituent systems and so the overall ER SoS performance. The quality of the communication depends on the type of the systems that are used, the service provider, and the communication coverage.

For our example we will consider the optimization of the communication coverage for the ER SoS of Berlin City.

Figure 4-7: CAE optimization concept

Figure 4-8: Communication coverage optimization case
As shown in Figure 4-8, the emergency system works at different locations. The communication antennas must be distributed in different geographical places across the city in order to achieve the best coverage area. At the same time the components of the systems are not in fixed places and they change their places according to the emergency case requirements. Beside the geographical place of the antenna, the communication coverage also depends on:

- Location
- Radiation power level
- Antenna height
- Antenna Patterns
- Polarization
- Frequencies
- Service dependent parameters

The previous points can be used as optimization parameters that need to be manipulated or optimized, as well as the geographical places of the constituent systems, that will use the communication system.

### 4.2.3 Optimization Goals

In order to proceed with the optimization process, the optimization goals have to be defined. In our communication network coverage case, the optimization goals are to:

- Achieve the best coverage area that serves the emergency response system
- Lower the cost: Using antennas to achieve the best coverage area is constrained by the cost. We need to lower the cost that distributed over:
  - Procurement cost
  - Maintenance cost
  - Running cost
- Provide the best quality: Not just the best coverage with a minimum cost is needed, we also need a good communication quality and services that can be supplied from the communications systems. This can be achieved by comparing different types of communication systems.

The global and local goals concept must be considered for the optimization, the architects have to decide the priority of goals that the system is optimized for. For example, lowering the cost of the overall SoS may resulting in increasing the local cost for a certain system.

### 4.2.4 Mathematical description

The antenna coverage can be calculated according to [1] using the following equations:

\[
PR_x(dBm) = E_RPTx - LMASK(\theta, \varphi) - L_p
\]

where:

- \( PR_x(dBm) \) is the received power in dBm.
- \( E_RPTx \) is the maximum Effective Isotropic Radiated Power of the cell in dBm (that is, at the peak gain point of the antenna).
- \( LMASK(\theta, \varphi) \) is the antenna mask loss value for azimuth and elevation angles respectively in the direction of the path being calculated in dB. When the received signal is directly on the main beam of the antenna, this value will be zero.
- \( L_p \) is the path loss in dB.

In order to calculate the path loss, the following equation is used:

\[
Path \ Loss \ (dB) = k_1 + k_2 \log (d) + k_3 (Hms) + k_4 \log (Hms) + k_5 \log (Heff) + k_6 \log (Heff) \log (d) + k_7 \text{Diffn} + C_loss
\]
Where:
- \( d \) = Distance from the base station to the mobile station (km).
- \( H_{\text{ms}} \) = Height of the mobile station above ground (m). This figure may be specified either globally or for individual clutter categories.
- \( H_{\text{eff}} \) = Effective base station antenna height (m).
- \( \text{Diffn} \) = Diffraction loss calculated using Epstein, Peterson, Deygout or Bullington equivalent knife edge methods.
- \( k_1 \) and \( k_2 \) Intercept and Slope. These factors correspond to a constant offset (in dBm) and a multiplying factor for the log of the distance between the base station and mobile.
- \( k_3 \) = Mobile Antenna Height Factor. Correction factor used to take into account the effective mobile antenna height.
- \( k_4 \) = Okumura-Hata multiplying factor for \( H_{\text{ms}} \).
- \( k_5 \) = Effective Antenna Height Gain. This is the multiplying factor for the log of the effective antenna height.
- \( k_6 \) = This is the Okumura-Hata type multiplying factor for \( \log (H_{\text{eff}}) \log (d) \).
- \( k_7 \) = Diffraction. This is a multiplying factor for diffraction loss calculations.
- \( C_{\text{loss}} \) = Clutter specifications such as heights and separation are also taken into account in the calculation.

### 4.2.5 Matlab Model

In order to calculate the coverage area for every antenna position, a Matlab model was built that distributes the investigated area on grid points.

![Matlab model results](image-url)
In Figure 4-9 the map is divided into grid points in order to simplify the coverage calculation. The yellow, red and blue colours indicate the coverage of antennas that are fixed in pre-defined points. From this model, the points covered by each antenna can be calculated.

The results from the matlab model will be used to feed the optimization engine with the data required for the calculations. The overall covered area can be calculated by the sum of the covered points without redundancy and according to their coverage percentage.

### 4.3 Modelling SoS Architectural Dynamicity

One of the major challenges in the domain of Systems of Systems (SoS) is the dynamically changing architecture of the SoS. This means that the participating Constituent System (CS) as well as their relations to each other change over time. This change can take place at different time scales. In this section we focus on long term changes for example changes taken place over years or decades. Changes during the operational time scale like one emergency operation are not part of this focus. For such short term changes well known methods and tools exist and e.g. UML covers this kind of dynamicity.

Long term changes come along with new functions and capabilities of individual CS and imply therefore also changes of the global (SoS) functions and capabilities. Similar questions arise during the design space exploration where alternative architectures are compared against each other. The same modelling mechanism can be applied to describe possible architectures because dynamic change impacts directly the architecture.

Dynamicity is modelled using graph rewriting rules which represent changes of the architecture. These rules are based on architectural patterns and can also be utilized to generate new architectures. One example of the emergency response (ER) scenario is the allocation of fire fighting cars to fire stations. This allocation is not statically over time but adaptive to for example the change of the size of the city. A local change (a local allocation of one fire fighting car to one fire station) changes the architecture and will be applied several times during the SoS lifecycle. The possible functions and capabilities of the ER SoS depend on the number and order of applications of such change.

By modelling the evolution of the SoS, questions concerning sets of architectures can be answered. To enable reasoning about these architectural alternatives the generation of these alternatives is required. The investigation if a certain property is fulfilled by a set of architectures can be done with and without generating all alternatives. Without the generation of alternatives only structural and static properties can be taken into account even in infinite sets of alternatives. The drawback is that no behaviour related dynamic properties of the alternatives can be investigated. The approach for DANSE is therefore to generate a small set of alternatives and verify the goals and contracts of the SoS as well as of the CSs by simulation and statistical model checking. This supports the identification of unexpected emergent behaviour on the simulation level.

The drawback is obviously the high computational complexity. Therefore we propose to apply this technology only to the (abstract) SoS level and not to (detailed) system level as well as to model the evolution rather than modelling the operational behaviour which also may come along with architectural changes.

#### 4.3.1 Modelling Concepts

The evolution is modelled by defining so called rules. These rules describe how two architectural patterns change one subset of architecture to another.

A rule contains a left-hand side (LHS) and a right-hand side (RHS) pattern. In general a rule can be applied to a graph if the LHS is a sub-graph of this graph and this sub-graph cannot be extended by embargo elements. Embargos are negative pattern elements to complement the expressiveness of the rule by a “not”. The graph is in this context the UPDM/SysML model and each application of a rule implies a new UPDM/SysML model. Such an application of the rule changes the sub-graph matched by the LHS to the RHS. Note that all not matched elements are changed. To model a rule the following roles of model artefacts for a rule are required:

- **Reader**: Elements marked in the rule as reader have to be matched but are not changed. They appear in the LHS and RHS.
- **Eraser**: LHS elements not element of the RHS and therefore removed by the application of the rule.
- **Creator**: RHS elements marked as creators are added by the rule to the graph.
- **Embargo**: Since the matching of the LHS defines only the required element one cannot restrict a match without embargos.
For example: If a component in a model is not connected, a rule could add a connection to another component having a port free to connect to. Between the ports and the components there is an “own” or “containment” relation. The LHS of the rule would require to component (or some more detailed type) to have at least one port. The match would be restricted to not have a relation from each of the ports to a connector using the embargo role. The RHS would consist of a new connector with role creator and with relations to the two ports also in the role creator. The set of rules can be applied to a (e.g. incomplete or initial) architecture and the matches as well as the order of application may generate different new architectures which can afterwards be investigated. In general a rule may match several times in the same architecture and therefore each different match results in a new architecture.

4.3.2 Implementation

In DANSE the UPDM is used as main modelling language. This profile enables SoS modelling with the tool Rhapsody which is based on the SysML modelling language. The UPDM has been analysed and one result the identification of the gap to model dynamicity in an appropriate way. To enable this important modelling feature a profile was developed which adds the required element to UPDM. The basic elements are like described above the different roles of model element within a rule. To model a rule a new diagram type is included in the dynamicity add-on profile and allows specifying rules directly in Rhapsody. Due to the nature of the roles, a single rule can be modelled in one diagram. Such diagrams are called “StoryCharts” and contain the LHS and the RHS. This is possible because the embargo and removers only appear in the LHS, the creator only in the RHS and the reader in both. To model a rule graphically some annotations are made to the model elements in order to identify the different roles. From technical point of view the elements in the graph rule diagrams are only representatives of the actual model element to avoid multiple annotations if one model element is used in several rule diagrams. For the user this difference is kept as transparent as possible.

Figure 4-10 and Figure 4-11 illustrate two rules for the CAE behavioural model. Note that the blue lines represent the eraser elements and the green ones the creator elements. The first rule models that a fire fighting car changes its communication relation from one fire station to another. The second rule models the creation of a new region between two already existing ones.
4.3.3 Dynamicity applied to the CAE

The CAE behavioural part as introduced in chapter 2.1.2 presents the behaviour of one subset of the CAE ER SoS that mainly consists of ten districts of a city, several fire fighting cars, four fire stations and one fire head quarter. The dynamicity in this setting can be located at the assignment of fire stations to the regions, the allocation of fire fighting cars to fire station, an additional fire head quarter or the growth of the city in term of an increasing number of residents and/or additional districts. The concept of graph rewriting is used to describe what are the possible architectures of the SoS. This concept can be used during the design exploration phase where different possible architectures are identified and compared against each other. The other purpose is to anticipate possible future architectures which result from evolutionary change of the SoS.

In the CAE behavioural part, a design exploration scenario is to assign the responsibility of the fire stations to the districts. This assignment goes along with the challenge to ensure that the fire fighting cars can reach each district within a certain time bound. One intuitive first allocation is therefore that the district where a fire station is located is always assigned to this fire station. For all other districts the assignment must ensure that the distance between district and fire station is not too far. The exploration scenario is defined as if the fire stations are already deployed (which is naturally the case in existing cities) and the number of fire fighting cars shall be minimized by ensuring the maximal response time limit. The modelling of dynamicity allows specifying the allocation of fire fighting cars to fire stations. The response time could be abstracted by the number of districts between a district and the responsible fire station.

This example will be used to describe the methodology in detail in this paragraph. Let assume that the fire stations are located as indicated in Figure 4-11. The challenge is to assign the responsibility of the fire stations to the remaining districts. This is modelled by a communication relation between the fire station and a fire fighting car and a communication relation between the fire fighting car and the district. Using the concepts of the SoS modelling formalism the communication relation between the fire station and the fire fighting car is not only a communication relation it is actually a coordination and/or ownership relation because the fire station uses the fire fighting car like a sub-system. The communication relation between the fire fighting car and the district is an interaction relation because the fire fighting car directly interacts with the district for example by deletion of a fire which is an intrinsic value of the district in this model. There are two rules which describe that a fire station gets responsible for a district.

1. The fire fighting car is already connected to the fire station. A coordination relation between the fire fighting car and a district is added.
2. (At least) One fire fighting car is not allocated to a fire station. Like in (1) a coordination relation between the “free” fire fighting car is added as well as an interaction relation to the district. Both rules only to districts not connected to any fire fighting car and depending on the abstraction of the response time only if the district is “near” to the district hosting the fire station. Figure 4-12 illustrates the (1) rule in Rhapsody without the details for the “near” relation which could be only direct neighbours.

![Figure 4-12: Add relation to district](image)

In addition also limitations on the connections between the fire fighting cars and the districts should be taken into account to prevent a high number of districts related to only one fire fighting car. If this number is unknown or depends on the individual districts (number of residents, size of district), then the rule could get quite complex and/or it would be more easy to analyse this property of the resulting model using other tools. The challenge is to decide which properties shall be taken into account by the rule and therefore also have to be part of the model. If one may want to take public needs into account the impact of change is maybe very complex and could exceed the dynamicity modelling in terms of usability. In such cases one could apply a rule to the model and analyse the need property based on sophisticated analysis and/or simulation methods. The result can afterwards be feed back into the model. This chaining of architectural changes and impact analysis is probably always required if the properties of interest originate from complex interactions which cannot (or only with high effort) explicitly modelled. One class of such properties are the emergent properties or behaviours which are not known a priori and can therefore not be part of a rule.

The power of graph rewriting rules is in this context that rules tend to be locally and not taking the whole SoS into account. A rule could describe for example a local change at any point in time but also define which conditions must be fulfilled in a more global scope. To sum up, the presented approach can be used for modelling dynamicity and to generate design alternatives. The benefit to use it for both purposes is that the rules itself could partially reused in many case because of the dynamic nature of SoS.
5 CAE – SoS Behavioural Modelling for Run Time Analysis

In addition to the ER SoS structural modelling, behavioural modelling is required to simulate the SoS and to perform any kind of run time analysis. Being able to simulate the whole SoS makes it possible to observe (un)expected emergent behaviours, as well as to automatically verify that the SoS goals are achieved with a certain probability, using statistical Model Checking techniques.

The behavioural modelling has been performed on a subset of the CAE constituent systems. The intention is to provide an example that can support and drive the different run-time related techniques involved in DANSE.

The next sections will give a deeper insight of what has been modelled and how. The use of stochastic modelling artefacts, especially to support human behaviour modelling, will also be addressed. The topic of goals formalisation for our SoS will then be discussed, so that they can be eventually verified by Model Checking. The last two sections of this chapter will be about alternatives performance and emergent behaviour analysis.

5.1 Constituent Systems behaviour

In order to perform simulation and run time analysis, the behaviour of the SoS constituent systems must be modelled. The behavioural models can potentially be designed in different languages and tools, such as statecharts using Rhapsody, Simulink or Modelica models, etc. Since simulation of the whole SoS is targeted, all these constituent system models must be exportable to a format that enables hosted or co-simulation. The FMI standard (Functional Mock-up Interface) is the one chosen in DANSE.

In the perspective of running a SoS with thousands of constituents, it is necessary to abstract their behaviour to make the simulation work within the time and resources constraints. If the abstraction is too low, there is a risk that the whole simulation will not run, but if it is too high, the risk is to miss some emergent behaviours that could have appeared with more details. So ideally, the level of abstraction should be as low as possible, as far as the simulation is still practicable at SoS level. There are several ways to do behavioural abstraction, either manually by analysing behaviour and simplify it in a higher-level formalism (such as statecharts), or (semi-)automatically, for instance by using machine learning algorithms to build a surrogate model. Such techniques are discussed more in depth in a dedicated deliverable.

There is another interesting point to highlight: in a SoS context, the relevant behaviour of most constituent systems is likely to be more related to the way they are operated rather than their real intrinsic physics. In the frame of the CAE for instance, the behaviour of interest for a fire fighting car describes the way firemen use it to drive to the fires according to the orders they receive, rather than the physics of the car itself (even in an abstracted way).

Behavioural modelling in the CAE is focussed on following constituent systems: Fire HQ, Fire Station, Fire Fighting Car, Fireman and District. They compose the SoS of interest here (subset of the whole CAE) and are defined on a SV-1 BDD view inside Rhapsody, as shown in Figure 5-1.

Figure 5-1: SV-1 BDD of the CAE behavioural model

The city districts have been added as constituent systems because they play an important role in the SoS: their behaviour describes how the fires arise, expand and spread to neighbour districts. In the frame of the CAE, all behaviours are abstracted in statecharts using Rhapsody, except for the human behaviour of the Fireman, which was modelled externally using the Modelica language.
The example is roughly based on Paris (the 10 first districts, called “arrondissements”).

![City Districts](image)

Figure 5-2: City Districts

It is assumed that the role of the SoS architect consists in optimizing the process of the commanding Fire HQ, in order to minimize the risks in the city (burnt area, number of death…), given the position of the fire stations and fire fighting cars allocations. These positions/allocations could typically be the result of an architecture optimization.

Before giving some details about the behaviour of each constituent system, note that their instantiation (i.e. the structure architecture) appears on an internal block diagram: the SV-1 view of our SoS subset, as shown on Figure 5-4. The specific attribute values of each instance (called “resource role” in UPDM) are also captured on this diagram. Constituents whose communication ports are connected can interact by sending events to each other. As one can notice, the only connectors that are used between parts are drawn from a port to another; no association link is used. In other words, a constituent system cannot directly refer to the attributes/operations of another one; all the required information must have been sent previously to it and stored locally (pretty much as in real life). This is because each constituent system shall be independent from one another, since they are to be exported individually using FMI and integrated in a simulation platform.
The 10 districts are modelled with following public attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>area</td>
<td>in km²</td>
</tr>
<tr>
<td>centerX</td>
<td>x position of the district center relative to the city center in km</td>
</tr>
<tr>
<td>centerY</td>
<td>y position of the district center relative to the city center in km</td>
</tr>
<tr>
<td>fireArea</td>
<td>in km² [0 when no fire]</td>
</tr>
<tr>
<td>fireDangerousness</td>
<td>mean growth percentage per second of a fire (depends on the presence of parcs, parkings, wood houses, chemical industries...)</td>
</tr>
<tr>
<td>fireRisk</td>
<td>mean time between two fires in hours</td>
</tr>
<tr>
<td>ID</td>
<td></td>
</tr>
<tr>
<td>population</td>
<td></td>
</tr>
</tbody>
</table>

Table 5-1: City District Attributes
When a fire is declared (because of the intrinsic fire risk or by propagation from a fully burnt neighbour district), it is supposed to start at the centre of the district. There is a delay before the fire is notified. The “UnderFire” composite state of the District state chart contains 3 regions: the first one describes the expansion of the fire inside the district and the propagation to neighbour districts when it is fully burnt, the second one handles the fire fighting by monitoring events from the firemen forces, and the last one describes when the fire is notified. Some modelling aspects such as the intrinsic fire frequency or time to notify the fire are described stochastically, and will be explained in the next section.

The role of the Fire HQ is to:

- Receive the fire notification and get the details
- Assess the number of fire fighting cars to send to the impacted district
- Choose where to find those fire fighting cars depending on the available ones in the fire stations, the position of these stations, etc.
- Send the order of sending fire fighting cars to the relevant stations

In addition to this, the statechart also has a region for data updates, used to keep track of the existing fire stations and car availability in each one of them.

Note that the “OrderSendFireFightingCars” state contains a bit of code as entry action, in order to specify the strategy of car selection for fire fighting (which would have been difficult and tedious to model another way). Here the process consists in finding the closest fire station to the fire that is not empty and send as many hosted cars as required by the notified fire size. If there are not enough fire fighting cars in this one, the Fire HQ will then look for the second closest station, etc.
Figure 5-5: Simplified behaviour of Fire Headquarter (SV-10b statechart)

The 4 fire stations are modelled with following public attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x position of the fire station relative to the city center in km</td>
</tr>
<tr>
<td>y</td>
<td>y position of the fire station relative to the city center in km</td>
</tr>
</tbody>
</table>

Table 5-2: Fire Station attributes

A fire station basically gets the orders from the fire HQ and transmits them to its fire fighting cars. Some delay occurs to collect the information and forward the order to the fire fighting cars.
Finally, 7 fire fighting cars are modelled with the following public attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td></td>
</tr>
<tr>
<td>isAtFireStation</td>
<td></td>
</tr>
<tr>
<td>isAtFire</td>
<td></td>
</tr>
<tr>
<td>efficiencyN</td>
<td>Current efficiency of the fireman N from that car (N = 1, 2, 3)</td>
</tr>
<tr>
<td>communicationN</td>
<td>Current communication skill of the fireman N from that car (N = 1, 2, 3)</td>
</tr>
</tbody>
</table>

Table 5-3: Fire Fighting Car attributes
A fire fighting car can be either at station (isAtFireStation = true), driving to/from a fire (the time to do so is computed by the operation “computeDrivingTime”), or fighting fire (isAtFire = true), which leads to decreasing the fire area in the district. The fire fighting performance is determined by the current efficiency of the firemen from that car, modulated by their communication skills as a team. These pieces of information come via flowports (called “ResourcePort” in UPDM) from the firemen instances, whose human behaviour is modelled using Modelica and exported as FMU.
Figure 5-7: Simplified behaviour of Fire Fighting Car (SV-10b statechart)
In addition to those SV-10b behavioural models, the human modelling of the Fireman was done in the OpenModelica Connection Editor (OMEdit). Some human characteristics identified in chapter Fehler! Verweisquelle konnte nicht gefunden werden. have been taken into account in this model: experience, competency, language, fatigue under stressful situations... Some are model parameters which can be valued per fireman directly on the SoS SV-1 view (experience, competency, language). Other are computed at execution time, based on the “isWorking” input of the model (fatigue). The first set allows computing the basic efficiency and communication skill of each fireman, whereas the fatigue will increasingly degrade these output performances during the time the fireman is working. Moreover, efficiency and communication are modulated periodically throughout the day (they are better during the day and worse at night). Here are some details about the model parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>experience</td>
<td>Number of years of experience as fireman</td>
</tr>
<tr>
<td>competency</td>
<td>Score of competency between 0.0 (worst score) and 1.0 (best score), according to training level</td>
</tr>
<tr>
<td>language</td>
<td>Score of language between 0.0 (worst score) and 1.0 (best score), according to communication and terminology skills</td>
</tr>
</tbody>
</table>

![Modelica code of the Fireman inside OMEdit](image)

Basically, competency is the main factor determining the optimal efficiency of a fireman. The more experience a fireman has, the more constant it will be throughout the day (i.e. less efficiency loss at night). Language is the main factor determining the communication skill of a fireman, but the latter is affected by the fatigue.

Figure 5-8 shows the Modelica code of the Fireman inside OMEdit.
In the simulation result graph, time is expressed in seconds. As explained above, one can notice that efficiency evolves periodically throughout the day and is globally decreasing over working time. The latter is also true for the communication skill.
5.2 SoS Stochastic/Statistical Modelling

Stochastic modelling is a way to describe behaviours that are not deterministic by nature, or to abstract a behaviour that is simply too complex to be modelled explicitly (as white box). So it is typically very useful in a SoS context. Behavioural modelling in the CAE shows numerous attributes/parameters that would not be deterministic, such as the time between 2 fires or the duration of an action performed by a human.

A first proposal of how to put stochastic data in the model has been integrated into the CAE. It is based on a set of attribute stereotypes that can be applied to any block attribute. This idea is close to the suggestion of the non-normative distribution extensions made in appendix of the SysML 1.3 specification, but adds the possibility to regenerate a distribution-based random value whenever needed (and not only at initialization), as explained in the next paragraphs. This addition is important because even the same person does never perform the same task in the exact same amount of time, so that the duration of the task shall be recalculated every time.

![Diagram of stereotypes to represent stochastic models]

Before explaining more in-depth the stochastic proposal, here are the key drivers that lead to it:

- be simple enough for the end-user (the SoS architect)
- be easily readable and exploitable by the run-time technologies
- be consistent and make sense from a modelling point of view
- be able to use a single distribution to generate several random values

Stochastic behaviour is classically represented by the concept of random variable. The probability distribution of the variable is interpreted as the probability that the variable takes up a certain value when it is observed. To follow this classical approach, given a random variable R that takes values over the reals and a real variable V, the assignment “V = R.observe()” can be seen as an observation of the random variable, provided that “R.observe()” function is an automatically or manually defined to generate new random values.
A simple example to illustrate this could be the statement “real initialForce = force_observe()”, where all such assignment statements are interpreted as distinct observations of the variable force. For random timing delays, one would have to instantiate a random real variable “delay” and to use it as “tm(delay_observe())”.

Based on these reflexions, here is the list of the proposed stochastic stereotypes (inside the DANSE profile):

- **UniformRandomReal** → automatic addition of “RhpReal observe()” at code generation for generating a real based on uniform distribution
  - min:String
  - max:String

- **UniformRandomInteger** → automatic addition of “RhpInteger observe()” at code generation for generating an integer based on uniform distribution
  - min:String
  - max:String

- **NormalRandomReal** → automatic addition of “RhpReal observe()” at code generation for generating a real based on normal distribution
  - mean:String
  - standardDeviation:String

- **NormalRandomInteger** → automatic addition of “RhpInteger observe()” at code generation for generating an integer based on normal distribution
  - mean:String
  - standardDeviation:String

- **CustomRandomReal** → manual addition of “RhpReal observe()” to generate the random numbers (based on any kind of distribution)
  - customObserveFunction:String

- **CustomRandomInteger** → manual addition of “RhpInteger observe()” to generate the random numbers (based on any kind of distribution)
  - customObserveFunction:String
The reason for declaring all the attributes above as "String" is to be able to put expressions in there and not only numerical values (e.g. the mean value of the normal distribution of the city traffic is probably not fixed but dependent on the time of the day and the day of the week...). In addition to this, it would be worth considering the creation of an API allowing the overwriting of these expressions at any time.

The District statechart exemplifies the use of these stereotypes on 3 private attributes:

- <<NormalRandomReal>> delayBeforeFireNotification
- <<NormalRandomReal>> timeBetweenTwoFires
- <<UniformRandomReal>> initialTimeSinceLastFire

The first one is kind of a simple human modelling artefact expressing the time before someone notifies a new fire; the two last ones are used to stochastically model the frequency and phase of fires in the district.
5.3 Human Behaviour Modelling

System-of-systems (SoS) often include human-to-human and human-system interactions. Human behaviour may induce (ripple) effects that result in unintended consequences (emergent effects). A System of Systems design would be more effective if it would encompass an understanding of the human behavior of their intended users. When humans do not behave like assumed, it is difficult to create proper control functions with humans in the loop causing unexpected results.

DANSE DOW: One of the DANSE objectives is to evaluate (predict) emergent behavior related to evolving developments in the operational, human, and technical context.

5.3.1 Objectives:

Primary goal of this initiative is to create a method considering Human Behaviour and potential emergent effects in SoS environments.

Sub-goal is to exploit how Human Behaviour could be regarded in the design of SoSs.

Activities have been initiated to examine if an efficient method can be developed, which considers HB and potential emergent effects in SoS environments.

It is intended to present (demonstrate in a later phase) a method, which will identify emergent effects in SoS environments caused by HB.

Further analysis is planned to exploit HB in order to achieve most effective ways of HB interaction in SoS environments, and identify how HB could be regarded in the design of SoSs.

This initiative does not cover "on-line" (during operation) HB analysis. It considers "off-line" (during SoS concept and design) HB analysis.

5.3.2 Relevance of Human Behaviour (HB) in SoS:

The majority of System of Systems will have humans (operators), requiring to consider human behaviour in SoS design. Humans may be considered constituents being part of a System of Systems.

In the example we assume that the System-of-systems (SoS) includes human-to-x interactions (human-to-system interactions as well as human-to-human-interactions). Emergent effects in SoS environments may be caused by human behaviour, which may induce unintended consequences (emergent effects).

The human-to-x interaction comprises important, SoS-relevant aspects e.g. human beings initiate information exchange with and to systems. The information aspect has a high relevance to the SoS, because it serves as a basement for subsequent processes (e.g. planning, decision-making etc.). That's why the following chapters focus on information-sharing in Sos environments related to Human Behaviour. Quality-criteria for information are considered as well as HB aspects, which may influence the quality-criteria for information.

Human Behaviour relevance to Emergency Response

In Emergency Response domains information is closely related to Situational Awareness (SA). SA is depicted in systems and SA information is being exchanged in a SoS context. Human operators (Human Behaviour) contribute in many ways to the establishment of SA (e.g. human operators may provide incorrect or incomplete SA-information). The quality of SA depends not only on the technology available to distribute information but also on the quality of the information. Incomplete or false SA-information may influence the decision-making (and other aspects e.g. collaboration) in Emergency Response scenarios.

5.3.2.1 Examples of emergent effects related to HB:

HB may influence the quality of information. Since good quality of information is a prerequisite for many subsequent processes (e.g. decision-making, planning.), the following chapters focus on information-sharing related to Human Behaviour.

Example Scenario:

The CCC-officer needs to get local information from the centre of the catastrophe in order to gain a realistic picture of the current situation. Therefore a local policeman gives him an update every 30 minutes directly...
from the centre of the catastrophe. Based on this information the ccc-officer makes his decision and gives
commands to the different sub-systems. That means that the regularly updates from the police-office have
an influence on decisions which affect the whole SoS.

Human Behaviour can be related to operational and/or system performance in SoSs. E.g. a negative
operational effect in a SoS, caused by HB, could be a limited Situation Awareness. This would affect other
aspects e.g. in-effective decision-making, collaboration, etc.. among the constituents. Following examples of
HB and induced emergent effects (affecting the SoS) could be presented in the show-case:

**Operational impact:**

- HB effects on the quality of Information Sharing in the Emergency Response scenario. As a
consequence the emergent effect is changing the level of Situation Awareness. Further analysis
could lead to an assessment of operational performance, considering successive processes, which
use SA.

Further aspects of operational consequences needs to be considered in the analysis e.g. IS could
impact job performance, productivity level. IS could even be related to distress and hazard
situations, since incomplete and/or false information could cause/aggravate hazardous situations.

**System impact:**

- HB effects in providing inaccurate or false information inputs (caused by HB). The impact in the
(SoS) Emergency Response scenario causing erroneous processing of data, leading to errors or
misinterpretation of data.

**Criteria of Information:**

Information can be considered as a critical resource in SoS, which may significantly influence essential
processes in a SoS environment. Correct, relevant and timely information allows to make accurate
decisions, while erroneous, and irrelevant information hinders decision making, and may add to confusion,
which may substantially affects SoS performance.

**Information quality:**

Process performance in SoS environments is in many cases determined by the quality of information
available. HB may influence the quality of information in SoS environments. The diversity in type and quality
of information calls for methods and techniques that can support the analysis of information quality in SoS
environments, which may ultimately lead to improved SoS performance.

Information quality attributes are depicted in the table below:

<table>
<thead>
<tr>
<th>Table 5-5 Information Quality Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Info age</strong></td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>Uncertain</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Real-time</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Non real-time</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Cooperative</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Collaborative</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Date</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>Final</td>
<td>2013-04-25</td>
<td>56</td>
</tr>
</tbody>
</table>
Additional quality attributes on a different level may be considered e.g. Information accessibility, error-rate, Information objectivity, Information consistency, Information completeness.

### 5.4 SoS Goals Formalisation

Formalising the SoS goals makes it possible to verify them automatically (with a certain probability) using a statistical Model Checker such as PLASMA in combination with a simulation platform such as DESYRE. As explained in section 3.2, it is not necessary to formalise local goals in order to verify the global ones, but it can be helpful to make sure that the constituent system behavioural models are in-line with their respective goals.

The language of goal formalisation that is developed in the scope of DANSE is called GCSL (Goal and Contract Specification Language). To support this activity, a list of 16 goals related to the (behavioural) CAE has been written, accompanied by a first suggestion of formalisation language.

Three key elements are actually required for formalising behavioural goals (suggested ways to address them are indicated):

- Being able to refer to model elements → use of the same names as in the model (in red below)
- Being able to write expressions about them → use of OCL (in blue below)
- Being able to integrate these expressions inside a temporal logic → use of CSL patterns (in bold below)

OCL (Object Constraint Language) is a formal language by the OMG used to describe expressions on UML models. OCL can be used for a number of different purposes, but especially as a model-based query language and for writing expressions, which perfectly suits our needs here.

CSL (Contract Specification Language) has been developed in the previous EU project SPEEDS, and comes with convenient temporal patterns.

To see the full list of goals and suggested formalisation (with comments), please refer to the Excel sheet that comes with the CAE. Here are just a few examples:

<table>
<thead>
<tr>
<th>Goal</th>
<th>GCSL suggestion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proba(The fire is stoppable in the city) &gt; 99.9%</td>
<td>Whenever [SoS.itsDistricts.fireArea-&gt;sum() &gt; 0] occurs, [SoS.itsDistricts.fireArea-&gt;sum() = 0] occurs within [0, false]</td>
</tr>
<tr>
<td>Proba(A district shall not stay under fire more than 3 hours) &gt; 99%</td>
<td>SoS.itsDistricts-&gt;forAll(district</td>
</tr>
<tr>
<td>A fire station shall host at least 1 fire fighting car</td>
<td>SoS.itsFireStations-&gt;forAll(hostedFireFightingCars-&gt;size() &gt;= 1)</td>
</tr>
<tr>
<td>A district shall not have more than 1 fire station, except if all districts have at least 1</td>
<td>SoS.itsDistricts-&gt;exists(containsFireStations-&gt;size() &gt; 1) implies SoS.itsDistricts-&gt;forAll(containsFireStations-&gt;size() &gt;= 1)</td>
</tr>
<tr>
<td>Proba(A fire station shall not stay empty for more than 3 hours) &gt; 99%</td>
<td>SoS.itsFireStations-&gt;forAll(fireStation</td>
</tr>
<tr>
<td>Proba(Mean city area under fire shall be less than 0.01%) &gt; 99%</td>
<td>MEAN(SoS.itsDistricts.fireArea-&gt;sum(), duration/interval) &lt; 0.01/100 * SoS.itsDistricts.area-&gt;sum()</td>
</tr>
</tbody>
</table>

**Note:** MEAN(..., ...) shall be implemented in PLASMA
Note that some formalised goals embed a CSL pattern inside an OCL expression. This is another suggestion for GCGL in order to easily express that a temporal logic property must be verified several times on different constituents for instance.

### 5.5 Analysis of Alternatives Performance

Once the structural model of the SoS is established and the behavioural models of each constituent system are designed, everything is ready to be exported to the simulation platform that can virtually run the SoS. Since a lot of parameters are likely to be described stochastically (environment variables, human related values, random systems inside the SoS...), running the simulation several times should not always produce the exact same result. So even if the SoS is simulated a lot of times and all its global goals were verified so far, it is not certain that it will always be the case. Fortunately, if the goals were formalized, the statistical Model Checker is able to assess the probability that the goals are true, by monitoring the simulation traces (the more simulations are run the more precise the result will be).

The SoS architect is supposed to engineer and rework the SoS structure until he gets satisfying probabilities for all the SoS goals. Once it is the case, he can either stop or continue the process to try other SoS architectures, that may be also valid and even better than the first found regarding certain key criteria. These criteria can be potentially derived from the key SoS goals, reformulating them as optimization objectives. For instance, the goal “Proba(Mean city area under fire shall be less than 0.01%) > 99%” (which is still used as a threshold to define valid architectures), can be reformulated as the optimization objective “Minimize mean city area under fire” (which can be used to select the best architectures among the valid ones). A lot of other examples of reformulations are given in the Excel sheet about the goals of the CAE.

So far, the discussion was about the validity and performance analysis of several SoS architecture alternatives. But validity and performance analysis can also occur at runtime for one given SoS architecture if runtime dynamicity is used. Indeed, graph grammar rules (or other mechanisms expressed by the SoS architect) typically define a set of possible evolutions for the SoS. But at this point, in order to select which evolution to select in the set, an on-the-fly (simplified) validity and performance analysis may be performed at runtime. This should of course provide better evolutions than a purely random picking.

### 5.6 Emergent Behaviour Analysis

One primary goal of the DANSE methodology is to provide the ability to predict emergent behaviours before observing them in the real world. It is believed that the DANSE simulation tools will allow this to happen, as other simulations have seen in the past.

Emergent behaviours are defined as those behaviours that occur at the level of the whole that cannot be perceived at lower levels. They usually exist because there are characteristics of the whole that do not exist at lower levels, characteristics that only come into being when the parts are assembled into the whole. It is in the interactions of the constituent systems that such emergent behaviours come into being, and their existence often causes changes within the constituent systems as well, in response to what they perceive as environmental interactions.

#### 5.6.1 CAE Examples of Emergent Behaviour

Within the CAE, some evident examples of emergent behaviours are:

- **Cross-coordination of resource usage.** Fire brigades operating alone optimize their resources applied to emergency events based on their knowledge of their own resources. They do not consider the presence of ambulance services or police forces; only when the different domains exist together in an SoS does such cross-coordination occur (or even exist).

- **Adaptation of event perimeters.** Police forces set up perimeters around an emergency event, but they do not move or change those perimeters to adapt for the needs of fire brigades or ambulances unless the different forces exist together.

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Date</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>Final</td>
<td>2013-04-25</td>
<td>58</td>
</tr>
</tbody>
</table>
Governmental cost effectiveness. Standing alone, the various emergency services agencies optimize their cost of operation based only on their own needs and knowledge. When they co-exist, then the cost effectiveness can be optimized across all forces.

Coordinated communications development. As individual agencies, each force determines its own needs for communications. They select and acquire communications equipment for their own needs. When they interoperate, however, the inter-force interaction causes them to consider the needs of other forces in their selection of communications equipment.

Coordinated response to city growth. Individual forces respond to city growth by acquiring equipment and facilities based on the need of the individual force, placing the facilities and equipment where they will balance the need against the costs. When the forces interoperate as an SoS, however, the selection of new facilities/equipment may lead to co-location or at least consideration of the presence of other force facilities.

In all of these situations, and many more, the interaction with other forces as part of an SoS actually changes the behaviour of the constituent systems. The result at the SoS level is evidenced as behaviours that are useful (and only evident) at the level of the SoS: cross-coordination, adaptation, overall cost effectiveness, coordinated communications, and coordinated city growth all contribute to a better SoS. None of these occur at the level of the constituent systems. (How can there be cross-coordination if there is no other force with which to coordinate?) But the presence of these higher-level emergent behaviours also has an effect on the constituent system behaviours, causing them to self-modify due to the interactions with the other constituent systems so as to enhance the higher-level goals.

However, there are also other emergent behaviours that are surprises to the designers and/or participants of the SoS. Following the same definition, these behaviours also occur through the interactions of the constituent systems and are only evident at the level of the SoS. They may be either beneficial or detrimental as perceived by humans in the SoS. Some examples of possible surprise emergent behaviours in the CAE might be:

- Traffic jams in unplanned locations. In this case, the combined operations of all the emergency forces can be expected to create traffic jams in the vicinity of the event. However, jams may occur in more distant locations due to the vehicles transits to/from the event or due to extended effects of the local jams.
- Communications overloading. Despite coordinated plans, the interaction of many units on the same frequency can sometimes cause overloading. Such behaviour carries additional surprise behaviours, in the lack of coordination that was obstructed by the communications overload.
- “Hero” responses by individual units. Sudden situations in an event can cause unplanned favourable interactions among the responding units, such as where a police unit moves in to delay a fire line.

5.6.2 DANSE Discovery and Control of Emergent Behaviours

The DANSE technologies allow for both discovery and control of emergent behaviours within the CAE. Each of the following technologies contributes.

Joint Simulation. The use of joint simulation allows the SoS planners to observe the SoS behaviour within a safe environment. Simple simulations often suffer in this regard, in that the simulations do little more than confirm what the simulation designers built into the model. More complex simulations, however, allow the same type of interactions that occur within the actual SoS; the interaction of constituent system models results in behaviours that can approximate the real world behaviours. For this type of behaviour to occur, each model must provide sufficient realism that the models in fact interact in complex ways. However, increasing the realism of constituent system models may prevent the simulation from running in useful times. Therefore, DANSE considers the abstraction of the constituent system models both to enhance the simulation run times and to allow simulation within a common environment with the SoS model as described in Section 5.1.

Goal and Contract Specification Language (GCSL). Emergent behaviours may be either design issues or surprise issues. GCSL as discussed in 5.4 is useful for both. For design issues, GCSL can structure the necessary emergent behaviour as either goals or contracts. During simulation, the operation is checked to ensure that the necessary emergent behaviour actually occurs. For surprise issues, the GCSL also structures the expected and required behaviour. Surprise emergent behaviours evidence themselves as transgressions of the GCSL.
**Graph Grammar, Patterns, Architecture Optimization.** These techniques, discussed in Section 4, allow automated or manual variation of the SoS architecture. Emergent behaviours are usually highly dependent on the specific interactions of the architecture; changing architecture usually causes significant changes in the emergent behaviours. Within the DANSE methodology, the joint simulation is run repeatedly with variations in the architecture to discover which architectures evidence which emergent behaviours.

Running and observing the simulations based on the designed SoS structure and constituent systems behaviours appear to be the only way to detect emergent behaviour. As mentioned previously, the right level of modelling abstraction must be found in order to:

1. Be able to simulate the whole SoS within the time and resources constraints
2. Not exaggerate the over-approximation to avoid missing some emergent behaviours that may have appeared with a more detailed modelling
6 CAE- DANSE Technology Integration

The objectives behind DANSE technology integration is to generate an overarching methodology that integrates different technologies and solutions developed within the CAE together, and to define a structured way for SoSE in an iterative method, while filling the gaps in technical solutions interfaces. The process starts with defining a detailed model out of the CAE. The second step is DANSE technologies implementation under one model while applying the required model extension. Finally the technologies integration is extend to deal with emergent behaviour at runtime using the Predictive Runtime Optimization Approach.

6.1 Integration Model

The purpose of the integration model is to generate a detailed model that applies the DANSE technologies to the SoS development and evolution. The Command and Control Center as a part of the CAE is modelled in a detailed UPDM model for this purpose.

6.1.1 Detailed UPDM Model

In order to increase the work efficiency and to concentrate the work efforts, a part of the CAE was chosen to represent the use case for the integration model. The Command and Control Center (CCC) as a major part of the emergency SoS is chosen. The CCC is a typical directed SoS system where heterogeneous and autonomous constituent systems interact and collaborate with each other to deliver emergency response services. CCC joins the capabilities and services provided by different kind of organizational, technical, and services systems to mitigate the sequences of emergency accidents.

Figure 6-1 Command and Control Center Operational Node representation

Figure 6-1 depicts the high level node representation of CCC and its interaction with other operational nodes. Each of these nodes is an operational node where different systems collaborate with each other in order to implement several operational activities. The operational view OV-5 in the UPDM diagram model were used to map the operational activities to the operational nodes using swim lines. The OV-5 also shows the operational flow and their connections (Figure 6-2).

The purpose of operational activities to operational nodes mapping process is to simplify the functional allocation process and to indicate the geographical allocation of the systems that will implement these
functions. In the UPDM diagram several OV-5 views where used to describe the mapping and the operational flow.

![Figure 6-2 OV-5 CCC Operational activity flow example](image)

### 6.1.2 Intergradation Model Scenario

A use case scenario is develop to simplify operations identification and allocation for the CCC. The scenario describes a situation where the CCC deals with an explosion case. The events and activities are sequentially depicted within a time line frame.
The scenario starts with an explosion that happens in a part of a city. CCC receives a call notification for the explosion and starts the emergency response process. Due to the explosion volume a lot of people who start to call the CCC were injured and affected and asking for help. CCC has to handle huge number of calls and lunch the required response.

As a result of the huge number of calls the call handling unit is overloaded. The CCC tries to reconfigure its organizational structure in order to reduce the overload on the call handling unit. During this process a problem happens within the communication system which damages the communication connection between the CCC and the other supporting units (i.e. Police HQ, Fire HQ, and Hospital HQ) that affect the emergency response operations. The CCC tries to reconfigure its structure and use other communication systems that could mitigate the communication damage effect.
The new configuration could not solve the problem and the CCC tries to use external organization support in order to keep the CCC operations and deliver the required emergency response.

The operations in the scenario are controlled by rules that sometimes will be violated or conflict each other. The system should try to stay consistent with the predefined rules that control the current and the next step operations. In the case of rules violation or conflict, a proper solution must be provided using different technical approaches. Also as a SoS behaviour an emergent behaviour may occur which could be good or bad. The technical tools must deal with such a behaviour by exploiting the opportunity of the good emergent behaviour and mitigate the risk and damage of the bad behaviour effects.

### 6.2 Technology Integration Flow

Different technical methods were developed within the DANSE project in order to provide solutions for different SoS challenges. Technologies integration flow describes the sequence of steps that should be followed to integrate these technologies together.

Figure 6-6 illustrates the technologies integration flow. The purpose is to magnify the use of different DANSE technologies within a structured use case, and define the problems and gaps of integration.
The whole case will be modelled in UPDM using DoDAF framework. In this process the required views will be defined as well as the different modelling approaches to represent the required models for the technologies implementation. The UPDM model is the starting point that will describe in detailed views with a high level of abstraction the operational and functional flow of the SoS as well as the required constituent systems.

### 6.3 Integration Model Extensions

As mentioned in the previous section OV-5 where used to describe the operational flow as well as the operational activities to operational nodes mapping. After that the operational activities will be dissociated to group of functions that implement these activities. Each of the operational activities is mapped to its associated functions in SV-5 and a matrix of this mapping is generated in SV-5a. Figure 6-7 and Figure 6-8 are examples for operational activities and functional mapping, and operational activities and functions matrix respectively.
From this point and according to the operational activities to function mapping, and using the operational activities flow views (OV-5) the functional flow and interaction is generated. SV-4 view from UPDM diagram is used to illustrate the functions interaction, flow, and dependencies (Figure 6-9). These functions are the required functions for the SoS and must be fulfilled using the constituent systems. They represent the basis for choosing the constituent systems and their interaction, also they defines who will communicate with whom and the interface requirement between the constituent systems.

The next step is to collect the candidate constituent systems and map them to the functions. The system view SV-1 from the UPDM diagram was used to model the mapping between constituent systems and the functions that they implement. In order to generate a mapping table, a table was designed in UPDM for this purpose. Figure 6-9 and Figure 6-10 represent an example for the mapping views between constituent systems and functions and the mapping table respectively.

Now at this stage all the required basics to generate the system of system architecture are modelled in the UPDM diagram.
Figure 6-9 Integration Model functional flow diagram

Figure 6-10 Integration Model constituent systems to functions mapping diagram
The next step is to define the architecture patterns that will be used, for this purpose the systems view SV1 is used as illustrated in the figure below.

These patterns will be used by concise modelling approach in order to generate an optimized architecture description for the CAE SoS. From this point, the behaviour of the constituent system will be added to the
UPDM model in SV10a. The model now is ready for further analysis and simulation process using different DANSE tools.

6.4 Predictive Run Time Optimization

The previous section presents the technologies integration in solving SoSE problems in design phase without considering the dynamic analysis of SoS behaviour. as mention in section 5.6 for emergent behaviour analysis, a primary goal for DANSE methodology is to predict emergent behaviour and prevent its future consequences. For this purpose we introduced run time optimization approach. The run time optimization approach will be able to:

1. Manage the system of systems operations
2. Enhance decision making process
3. Mitigate the bad effect of emergent behaviour of the system
4. Prevent future bad emergent behaviour effect for the next operational steps
5. Leverage, modify , and update system working rules

The Predictive Run Time Optimization approach collects information about the SoS constituent systems states and parameters, the SoS environment parameters, the current state of the SoS, the next step for SoS, and the different options for the SoS architecture patterns that could be used. Depending on the current status and the coming steps the run time optimizer provide the best option to be used while predicting the results using a short simulation for the next steps. The short simulation process detects emergent behaviour that will affect the system operations and tries to mitigate or eliminate its bad effect by reconfiguring the SoS structure and its constituent systems parameters.

The Predictive Run Time Optimization approach is a complementary approach for the previously mentioned techniques. While the architecture patterns and the architecture description in the previous model are optimized under static environmental conditions, the run time optimization approach works under run time environment and also tries to predict the results for any pattern chosen using simulation techniques which avoids any bad emergent behaviour could be resulted from applying this pattern. It is also an approach that modifies and enhances the predefined architecture rules that controls the SoS and solve rules conflict. Figure 6-14 illustrates the connection of the run time optimization approach to the previous model.
Figure 6-14 Predictive Run Time Optimization Incubator
7 Conclusion

It can be concluded that the Concept Alignment Example provides reference examples and illustrations of 63% of the DANSE Requirements (RD11). It shall be noted that not all DANSE Requirements are subject to illustration through the CAE deliverable D3.3 only.

In the present final version of this document the key alignment examples and associated models in the context of an exemplified ER SoS have been described that drive the development of DANSE methods and technologies at M18. For the DANSE 2nd year Review scheduled December 2013, there is a lot of work in progress by DANSE technology providers in order to mature the demonstrations and illustrations of DANSE methods and technologies applied on the CAE as described in this document as well as on the industrial DANSE Test Cases.

The CAE provides reference examples for the following DANSE methods & technologies:

- Methodologies for handling dynamicity
- Methodologies for handling emergent properties
- Reusable architectural and interaction patterns
- Validation of SoS architectures
- SoS domain metamodel
- Goal and Contract Specification Language
- Simulation framework
- Abstraction techniques
- Statistical model checking
- Formal verification
- Synthesis for diagnosis and prognosis
- Automatic test case generation
- Optimization techniques
- Tools interoperability technique

All 14 DANSE solutions will be addressed during the concept evaluation of DANSE results for year two. Since the CAE presents various sets of exemplified models and scenarios is expected that final validation of DANSE results will be performed through the DANSE Test Cases. The next phase of the project will show whether the “alignment” examples are sufficient to develop the more detailed industrial examples and to apply DANSE technologies. At this point it is not clear how many DANSE topics will be able to be applied on industrial examples, due to project constraints in terms of available resources. Accordingly, it could be even relevant to extend the existing CAE work with some more elaborations or details. Outputs of this work would reside in a further (though not planned) release of this document.

Finally, it must be noted that the presented ER SoS examples and models in this document do not fully comply with or complement the DANSE methodology for SoS Engineering. As explained in chapter 2.1.2, the elaborated examples and models in the ER SoS context have primarily been developed to serve as an input and to drive the development of various DANSE solutions in order to create a better understanding of relation of the DANSE Requirements to the DANSE technologies. Due to existing technical limitations of commercial SoS modelling tools, we were not able at M18 to demonstrate DANSE methods and technologies fully compliant with current practices in SoS Engineering based on SoS Modelling languages and Architecture Frameworks such as UPDM, MoDAF, DoDAF or NAF. There is significant remaining work to be performed on the definition of the DANSE overall SoS modelling methodology that shall highlight how the DANSE methods and technologies can be used in a larger industrial SoS Engineering context.
## 8 References

<table>
<thead>
<tr>
<th>N°</th>
<th>Source</th>
<th>Reference</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD1</td>
<td>International Journal for Engineering and Technology</td>
<td>RF Coverage Estimation of cellular</td>
<td></td>
</tr>
<tr>
<td>RD3</td>
<td>GROOVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD4</td>
<td>D3.1 Project Requirements</td>
<td>DANSE Internal Website</td>
<td>V1.0</td>
</tr>
<tr>
<td>RD11</td>
<td>D3.6 Validation Report - Prototype Iteration I</td>
<td>DANSE Internal Website</td>
<td>Version 1.0</td>
</tr>
</tbody>
</table>