Designing for adaptability and evolution in system of systems engineering

Conceptual and architecture principles of SoS design and semantic interoperability of systems platform and SoS design Tool-Net

D_8.1.3: 3rd revision (D_8.1.2 + D_8.2.3)

<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>
</tbody>
</table>
AUTHORS TABLE

<table>
<thead>
<tr>
<th>Name</th>
<th>Company</th>
<th>E-Mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uri Shani</td>
<td>IBM</td>
<td><a href="mailto:shani@il.ibm.com">shani@il.ibm.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>2013-03-11</td>
<td>Initialize based on D8.2.2</td>
<td>All</td>
</tr>
<tr>
<td>1.0</td>
<td>2013-04-25</td>
<td>Finished.</td>
<td>All</td>
</tr>
</tbody>
</table>
CONTENTS

1 INTRODUCTION .......................................................................................................................... 7

1.1 OVERVIEW, PURPOSE AND SCOPE ......................................................................................... 7

1.2 BACKGROUND ........................................................................................................................ 8

1.2.1 SoS methodology concepts – from WP4 – A Reminder ............................................................ 8

2 USER USE-CASES .......................................................................................................................... 11

2.1 Use case 1 .................................................................................................................................. 11

3 SOS DESIGN ASPECTS ............................................................................................................... 14

4 TOOLS INTEROPERABILITY CONCEPTS ..................................................................................... 15

4.1 DISTRIBUTED COLLABORATION .............................................................................................. 17

4.2 THE CONCEPT OF TOOLS-NET versus THAT OF TOOLS-CHAIN ............................................. 19

4.3 TOOLS IN THE TOOLS-NET .................................................................................................... 19

5 USER ROLES IN THE INTEROPERABILITY ENVIRONMENT ......................................................... 21

5.1 TOOL VENDORS ....................................................................................................................... 21

5.2 IT NETWORK ADMINISTRATORS .............................................................................................. 21

5.3 POWER USERS ......................................................................................................................... 21

5.4 SYSTEM ENGINEERS ............................................................................................................... 22

6 DESIGN MODELLING SEMANTIC INTEGRATION CONCEPTS .................................................... 23

7 ARCHITECTURE PRINCIPLES OF THE TOOL-NET .................................................................... 28

7.1 ARCHITECTURE PATTERNS OF THE TOOLS-NET .................................................................. 28

8 TOOLS-NET SEMANTIC MEDIATION PLATFORM AND DOMAIN INTEROPERABILITY IMPLEMENTATION ......................................................................................................................... 29

8.1 THE SM CONTAINER ............................................................................................................... 29

8.2 DISTRIBUTION ......................................................................................................................... 30

8.2.1 Components of the SM Container .......................................................................................... 31

8.3 OPERATION OF THE ENVIRONMENT ...................................................................................... 35

9 CONCLUSIONS ............................................................................................................................ 40

10 THANKS .................................................................................................................................... 41

11 ABBREVIATIONS AND DEFINITIONS ......................................................................................... 42

Version | Status | Date         | Page
---------|--------|--------------|----
1.0      | Final  | 2013-05-06   | 3 of 48
REFERENCES
Content of Figures

Figure 1: The DANSE systems engineering lifecycle .................................................................8
Figure 2: Tools and models flow for initial design use-cases ......................................................11
Figure 3: Point to point vs. “hub and spoke” ..............................................................................15
Figure 4: Jazz components .........................................................................................................16
Figure 5: models, relations, ontologies and rules ........................................................................17
Figure 6: Sharing a single server for multiple tools and several partners. ..............................17
Figure 7: Distributed, multi-partner and multi-project collaboration environment ..................18
Figure 8: Example demo of network configuration of a distributed collaboration environment ...18
Figure 9: Mediation and links ....................................................................................................23
Figure 10: Semantic overlaps ......................................................................................................25
Figure 11: A hierarchy of semantic overlaps among modeling languages ...............................26
Figure 12: Semantic mediation network showing a “hub and spoke” layout of tools that exchange models with a server hosting a network of hierarchical mediations. ..................................................27
Figure 13: Architecture patterns of the tool-net .........................................................................28
Figure 14: SM Container architecture .......................................................................................29
Figure 15: Jazz client/server configurations .............................................................................30
Figure 16: Network considerations for the distribution SM Container .......................................31
Figure 17: SM Container components and typical configuration with two possible flows ........32
Figure 18: The mediator execution environment mediating a input model from an input port, to an output model in an output port ................................................................................33
Figure 19: SM container configuration UI such as for the above example. ...............................34
Figure 20: SM Container configuration UI in a Jazz/DM page ....................................................35
Figure 21: System Architect exports a model to a specific RESTful POST port in the platform configuration.36
Figure 22: The model being mediated and stored in a repository can be browsed via the web ....36
Figure 23: A multi-staged mediation process .............................................................................37
Figure 24: Progress of the mediation reflected on the status of repositories in the web console. 37
Figure 25: Browsing a repository RDF in which resources can be clicked and reviewed ..........37
Figure 26: Comparing versions of the model in a repository as it is being changed through mediations.38
Figure 27: Applying SPARQL on an RDF model .....................................................................38
Figure 28: Linked data information can be browsed for each individual mediation link among pairs of repositories. ....................................................................................................................39

Content of Tables

Table 1: Design Use case 1 for Initializing SoS Modeling from CS tools. ..................................13
Table 2: Tools proposition by IAI, not discussed yet. .................................................................20
1 Introduction

1.1 Overview, Purpose and Scope

This document is a follow up on the combined delivery of D8.2.2 [37] for the WP 8 (“SoS Tool Net and DANSE exploitation”) tasks T.8.2 (“Formalize the main concepts part B”) and T8.3 (“Create and validate architectural decisions” – M4-M30). In this document we document the working and operational tools ecosystem we are adopting from the SPRINT [7] project as an exploitation of that technology. We thus apply further understanding of the supporting tool-net platform required for SoS design within the DANSE ecosystem.

The document continues to support many of the concepts and principles described in the earlier document which are not repeated here.

The concepts of SoS design are a culmination of concepts following each of the major technological challenges of DANSE. That is the background for DANSE setting up the conceptual context of the Tool-Net. Each of these challenges is managed by one of DANSE technological work-packages (WP-s), which will also provide these concepts.

The collaborative context in which DANSE’s Tool-Net operates is also rich with standards that DANSE adopts. Moreover, the platform of choice that respects these standards and which will be used to implement the collaborative DANSE platform of tools is the open Jazz [1] platform by IBM. All of these are discussed in the background chapter of the earlier revisions [37] and has been scrapped off this document.

Therefore, our initial chapter discusses the SoS design aspects, which would usually start with legacy constituent systems, support evolution of that design into new architectures and new constituents. Here, we add an example of tools-net design work flow by users that was tackled during a face-to-face meeting of the project in Haifa, Israel on Feb 2013. That use case is an example to be followed and which will teach more capabilities required to do the SoS lifecycle.

We next deal with new material on the eco-system for tools interoperability to support the various work flows as identified by DANSE users and such that match the DANSE methodology. This chapter is technical in the sense that interoperability among tools and the SoS tool-chain must be well defined and anchored in technologies and standards that will make worthy the investment in time and effort in tools adaptation. We elaborate more with results obtained during the second half of the first DANSE year where that platform has been implemented and disseminated among partners being tools developers, WP leaders and users. That later dissemination is progressing slow and other tools integration does not occur in an effective pace.

The next two chapters deal with the DANSE architecture and the execution plans and the progress with these the development and deploying the DANSE service platform – a distributed and collaborative ecosystem based on the Jazz open system. In Chapter 5 we introduce users’ roles in dealing with the ecosystem which also includes the engineer who needs to be aware of the working in collaboration and in an
environment which extends beyond the tool he/she is using. These roles are than served to elaborate on the concepts and on the architecture of the DANSE tools net ecosystem in the following chapters, leading to the conclusions and summary chapters.

Abbreviations and references are updated and extended.

1.2 Background

For the background we refer the reader to the initial versions of this series of deliverable [37]. We keep in here only the SoS methodology discussion per WP4 as it reflects on the general concepts that the tool-net is designed to support. The rest of background discussions are scrapped from this revision. This next section has been kept as it was in the previous document.

1.2.1 SoS methodology concepts – from WP4 – A Reminder

The distinctive feature of the DANSE SoS engineering lifecycle is that it relies upon an evolutionary process. (See Figure 1) Classical models of systems engineering, such as the V Model rely upon a top-down view of design and a linear, once-through approach to systems engineering. The SoS of interest to DANSE are an aggregation of pre-existing 'legacy' systems, with added functionality, and new constituent systems developed specifically for the new SoS. This has consequences on the systems engineering lifecycle.

![Figure 1: The DANSE systems engineering lifecycle](image)

The classical approach to design aims at producing systems that are "correct the first time", that is, systems that satisfy all constraints and optimize given objectives without resorting to expensive redesigns after a system has been placed in operation. Because of the inherent complexity of an SoS in terms of sheer number of independent agents that are part of the SoS, and because an SoS is usually an opportunistic aggregation of legacy systems, obtaining the correct behavior when first placed in operation is highly improbable. For example, we experienced the many iterations and many months it took to bring up the Heathrow airport (UK) after renovation and the Bay Area Rapid Transit system (USA). We believe that an evolutionary approach to correcting the behavior of a SoS is necessary. The DANSE methodology will shape
the behavior of a SoS over time as more is learned about the constituent systems behavior and the emergent behavior due to their interaction, so that the system becomes “correct by evolution.”

**Figure 1** shows five system-of-systems engineering processes and a set of constituent systems engineering processes. The first two processes are complementary:

1) **Model SoS behavior** – Simulation models run continuously during the lifecycle of the SoS and performance indices obtained by simulation are compared as they become available with operational data. Since most SoS are too large and complex to run comprehensive model validation experiments, validation cannot be but an incremental process. Modeling, validation and design corrections based on the proven technology of contract-driven design developed in SPEEDS [5] will be performed continuously.

2) **Operate the SoS** – An SoS will operate continuously and experience incidents of emergent behavior on a regular basis. When the operating data diverges significantly from the predictions of the operational model, the operational model must be adapted to cope with the inaccuracy that may lead to negative emergent behaviors.

3) The lower three SoS engineering processes embody a learning cycle and support steady performance improvement for modeling and operations over time, as the SoS evolves:

   a) **Define potential needs** – This task is based on a two-step process:

   b) **Identifying a need** - Ongoing analysis of system performance can lead to the determination of the needs in two ways. Either: 1) modeling and analysis suggest that better performance is possible or 2) the system’s behavior differs significantly from the current prediction obtained by the DANSE operational simulation. Incidents of unexpected emergent behavior may be negative (indicating performance problems or malfunctioning), or positive (indicating opportunities).

   c) **Analyzing the cause** – Root-cause analysis is supplemented with a knowledge base of previously analysed incidents to support rapid identification of recurring patterns. The knowledge base becomes increasingly sophisticated as the system and its operators "learn". The validation of a causal mechanism is strongly supported by the DANSE simulation technology.

4) **Analyze possible architecture changes** – Analyzed needs drive changes in the system to exploit opportunities or correct problems. A change may suggest fundamental changes to the architecture of the SoS or to the architecture of constituent systems. A key architectural tool is the predictive modeling and simulation capability used to compare architectural alternatives. Simulation also has an important role in the verification and validation of proposed architectures since logical validation can offer significant savings in testing.

5) **Influence and implement changes** – Types of changes include: 1) Updating a simulation model to give more accurate estimates of system behavior, 2) Modification of a contract between the SoS and one or more constituent systems. 3) Re-factoring the architecture at the SoS level.

The Vs in the lower part of the lifecycle diagram represent systems engineering efforts at the level of constituent systems. Constituent systems are typically legacy systems that were developed using a variety of engineering processes. During SoS operation, when the capability engineering process identifies a needed capability enhancement it may influence the various constituent systems to implement a change. Engineering processes at the constituent systems level are not dictated from the SoS level, but must be coordinated with the overall SoS goals applying contract based technology.

Most SoS are initially created as opportunistic aggregations of legacy systems after studying and mapping the potential constituent systems, external actors and other significant features. Typically, most or all of the constituent systems have been operating for some time prior to being inserted as part of the SoS.

In the Initial Phase, a descriptive model of the system is created first, using the DANSE language and modeling approach. The DANSE approach to creating a descriptive model is unique in that it applies
"contracts" to represent key performance and interface requirements. Contracts then form the basis for the dynamic, predictive SoS models applied throughout the management, operation and evolution phase.

Given the special nature of SoS described so far, the research done in the DANSE methodology area will integrate all simulation, architecture alternative selection and analysis techniques developed in the project. This will be done using the DANSE language and modeling approach with a focus on the two distinct, overlapping, phases of its life cycle:

a) Initial phase - mostly dedicated to the design of the "new" SoS.

b) Phase S - management and continuous evolution, where the design is now a continuous activity, interleaved with the continuous operation and evolution.
2 User Use-cases

User use cases are work flows which should be applicable among the tools and according to the DANSE design methodology. We start with the one discussed in the Haifa face-to-face meeting on February 2013. A significant work on use cases will provide more directional guidance to the tools-net. One general workflow - as also used by the following use case - is depicted in this next Figure 2.

![Figure 2](image-url)

**Figure 2:** Tools and models flow for initial design use-cases.

### 2.1 Use case 1

<table>
<thead>
<tr>
<th>Name</th>
<th>Initializing SoS Modeling from CS tools.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Existing CS are constituents of a new SoS. The constituents are modelled in different tools. To work on SoS design, we need to see a holistic view of the SoS with all constituents. Follow up with the SoS development life-cycle includes analysis and simulation at the SoS level.</td>
</tr>
<tr>
<td>Initial Situation</td>
<td>Constituents modeling CS in SysML have design with internal models.</td>
</tr>
<tr>
<td>Example</td>
<td>Constituent A is modelled by an SA tool, consisting of subcomponents A1 and A2, while constituent D is designed in another SA tool, consisting of subcomponents D1 and D2. A Rhapsody UPDM tool is available to represent the two constituents into one SoS. As in the following figure</td>
</tr>
</tbody>
</table>
Conceptual and architecture principles of SoS design and semantic interoperability of systems platform and the SoS design Tool-Net – 3rd revision

**Common sequence of events (SOEs)**

1. For each constituent CS, share its model content with the tools-net, mediating it to a usable UPDM component description.
2. Combine all these mediated models in a single tool representing a multi-constituent UPDM model of the now SoS design.
3. At the UPDM level of the SoS design, add SoS level design modifications and continue to work at the SoS level.

**Alternative SOEs**

None

**Follow-up SOEs**

- **Purpose:** common case of continuing the SoS life-cycle.
  4. Add additional constituents from more tools into the UPDM holistic view.
  5. Add SoS level design changes pertaining also to the new constituents

**Alternative A to follow up SOEs**

- **Purpose:** Perform SoS-level simulation
  - **Status:** Some of the constituents have simulations. Need to generate SoS-level simulation of these constituents.
    4. Define SoS top-level behaviour over the holistic view of the SoS
    5. Integrate the individual simulations of the constituents into the SoS-level behaviour
    6. Perform an integrated simulation

**Alternative B to follow up SOEs**

- **Purpose:** Apply analysis using GCSL (see figure)
  4. Apply “A-P” statements and formulas at the SoS top-level
  5. Propagate “A-P” statements down into inter-constituent dependencies
  6. Propagate “A-P” statements down into internal constituents’ structure.
### Purpose: Apply analysis using different configuration.

7. TBD

<table>
<thead>
<tr>
<th>Alternative C to follow up SOEs</th>
<th><strong>Purpose:</strong> Apply analysis using different configuration.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7. TBD</td>
</tr>
</tbody>
</table>

**Table 1:** Design Use case 1 for Initializing SoS Modeling from CS tools.
3 SoS design aspects

This chapter can be read again from the previous revision [37] and not repeated here.
4 Tools interoperability concepts

Earlier [37] technical discussions brought about the recognition of emerging tools and standards coming from the successful web and collectively known as “Semantic Web” Technologies.

The interoperability concept we adopt relies on these technologies and on platforms which support these emerging standards. The concepts for tools interoperability in DANSE are:

1. Model data is represented in the “Resource Description Framework” RDF [3], and transferred in any recognized and convenient syntax including XML (RDF/XML [43]), N-Triples [41] and Turtle [42]. Additional syntaxes may also be used in the future.

2. Model meta data describing the language underlying the model data is coded in the Web Ontology Language (OWL [26]), which is also represented in RDF.

3. Communications protocol is HTTPS [24] for security, whereas access authentication protocol is OAuth [25]. The interaction semantics adopt the RESTful [9] protocol over this HTTPS.

4. The tools-net architecture is of “hub and spoke” as in the right side of Figure 3. That architecture provides tools with full isolation from other tools in the tools-net, present and future, increasing interoperability modularity to a maximum and allowing manageability of an environment serving an expected community of tools to number at the dozens of tools.

5. The “hub” in this architecture is based on the IBM Jazz [1] (see Figure 4) which also offers a collaborative, single sign-on distributed environment with a powerful triples-store (RDF database service). To support web access to platform resources, and management of models and ontologies, we adopt the “Design Management” [44] Jazz application to host the central hub.

Figure 3: Point to point vs. “hub and spoke”.

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Date</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Final</td>
<td>2013-05-06</td>
<td>15</td>
</tr>
</tbody>
</table>
6. Models of different tools and therefore of different languages (which are defined as ontologies) are transformed at different levels of commonalities in a method exploited from the SPRINT [7] project and which is termed “Semantic Mediation”. Initial “Semantic Mediation” publications are starting to emerge ([45], [46]).

7. The platform, via the semantic mediation capabilities, can be configured to flow models between different languages, allowing the sharing of model data among tools which work in their own proprietary environment, in the familiar user experience of the engineer. Flow of models can also be configured to work among distributed interoperating platforms over the internet, across internet zones within corporate intranets and across the vast spaces among them on the open internet of via virtual networks. That interconnectivity among the platforms is too a matter of configuration.

8. Once on the internet, all model data, ontologies and mediation status can be browsed using web browsers. While tools may be rich clients, everything else in the tools-net eco-system is web oriented.

9. All model data on the tool-net is internet resources, such which RDF (R – for “resource”) is built of, and which are fully addressable internet entities which can be linked in accordance with the “Linked Data” [27] W3C principles, and as is depicted in Figure 5 below. In this figure, tools and applications on the top may be working on a common project and expose models as “user documents” with a related “reference” model in which some aspects of the models are maintained. There are relations between the models which are maintained against the reference model. These relations are depicted in this diagram as rules. It is possible to apply these rules to extract such relations from the ontologies layer in which ontologies for each tool as well as for the reference model provide in combination with “semantic mediation” rules the ability to create links that connect the individual models into a single holistic view of the project.

10. Compliance with the “Open Services for Lifecycle Collaboration” OSLC standard [8]. This standard fits into the overall concepts described above in a way that will become cleared in the next sections and chapters. On its face, the DANSE interoperability complies with the adoption of the RDF
representation and RESTful protocol of OSLC. Yet, the semantic mediation principles adopted in DANSE extends the power of the very basic presently pursued OSLC extensions (AM, CM, RM, QM etc.). Moreover, semantic mediation requires describing RDF models with OWL ontologies, which go beyond the concept of “Resource Shapes” in the OSLC specifications.

![Figure 5: models, relations, ontologies and rules.](image)

### 4.1 Distributed Collaboration

The hub server in Figure 3 above can be elaborated to represent the points discussed in the previous section in two steps. Firstly, demonstrating the interoperability of multiple tools over a single central server as in Figure 6, and distributing this environment over the internet as in Figure 7.

![Figure 6: Sharing a single server for multiple tools and several partners.](image)
The distributed configuration dictates the use of multiple servers in different domains over intranet and internets, with many options of collaboration based on projects and partners to the projects who may range over different companies having to protect their intellectual properties and share only what is needed to be shared.

**Figure 7:** Distributed, multi-partner and multi-project collaboration environment.

To further demonstrate this distribution, based on a specific experiment, **Figure 8** shows two servers residing in the internet yellow-zones of the owning companies, being available internally in each company for tools interoperability over the internal intranet, and across the two companies over the open internet, passing the firewalls via proper network management and standard protocols.

**Figure 8:** Example demo of network configuration of a distributed collaboration environment.
4.2 The Concept of Tools-net versus that of Tools-Chain

The DANSE Tools-net architecture depicted in Figure 3 as a “hub and spoke” does not surrender any clue as to how tools indeed interoperate. One cannot identify from this diagram a “network” of tools, not “chains” of tool dependencies. The concept of network is applied in the services within the hub (or cluster of distributed hubs) which – as we will see – allow model data to flow between tools. The star architecture simply ensures high level of modularity among the tools which have no direct interface to any other tool. The model mediation within the hub component is a strong concept that DANSE tools enjoy.

The network is semantic since it allows only compatible semantic concepts to be shared among the tools and thus it dictates what tools can exchange information with what other tools, and in which level.

Tool-chain is in fact a development use-case, such as we discuss in section 2.1 (and Table 1), where tools deliver model information in a certain flow and in accordance with some methodology. That flow is a concept which can be applied to the tools over the semantic network, through some control mechanism whose automation is yet to be defined.

Therefore, both of these concepts “tools-net” and “tools-chain” are virtual concepts with not direct physical incarnation, which are materialized over the physical “hub and spoke” tools eco-system star architecture, with the help of the semantic mediation technology that is embedded into the hub component of that environment.

4.3 Tools in the tools-net

The set of tools included in the DANSE tools-net starts to emerge as follows:

1. Design tools:
   a. Rhapsody
      i. SysML profile – SysML models can now be shared with the interoperability platform using a plugin developed in the SPRINT project.
      ii. UPDM profile – UPDM models sharing is developed by DANSE partner SODIUS.
      iii. “Architecture patterns” profiles developed by DANSE partner Loughborough University.
   b. Wolfram’s SystemModeler – a Modelica based tool for design and analysis can be exploited from the SPRINT project, but is not directly available to partners.
   c. System Architect NAF modeling sharing developed by DANSE partner SODIUS.
   d. Graph Grammar tool based on the GROOVE system is integrated by the DASNE partner OFFIS
   e. Requirements, quality management, work items and work flow processing tools from the IBM Jazz application suite including RTC (Rational Team Concert), RQM (Rational Quality Manager), and DOORS next gen are already internet oriented, OSLC compliant, and can use linked data.
f. Global linked data management tools from the Jazz tool suit including the indexing and query tool LQE (Lifecycle Query Engine) and the RELM (Rational Engineering Lifecycle Manager) combined with the DANSE tools-net and its internet capabilities and linked data compliancy will provide holistic view, analysis (impact, etc.) and search services for users.

g. Contracts (for the GCSL – General Contracts Specification Language) editor system DESYRE is integrated by the DANSE partner ALES.

2. Analysis and Optimization

   a. Rhapsody
      i. “Concise Modeling” optimization plugin developed by DANSE partner IBM.
      ii. FMI generation

   b. Simulation execution and monitoring by DESYRE is integrated by the DANSE partner ALES.

   c. Static analysis tool PLASMA developed by DANSE partner INRIA is used through the DESYRE tool (by ALES).

The tools compliancy table modified by removing all tools which are not likely to be integrated at all, and marked with the present status is repeated below in Table 2 next.

### Table 2: Tools proposition by IAI, not discussed yet.

<table>
<thead>
<tr>
<th>User</th>
<th>Tool name</th>
<th>Vendor/Provider</th>
<th>Category</th>
<th>Time needed</th>
<th>Use case</th>
<th>Discipline</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAI</td>
<td>DOORS</td>
<td>IBM</td>
<td>Requirements</td>
<td>Start</td>
<td>Water</td>
<td>Cross discipline</td>
</tr>
<tr>
<td>IAI</td>
<td>System Architect</td>
<td>IBM</td>
<td>Design</td>
<td>Start</td>
<td>Water</td>
<td>System</td>
</tr>
<tr>
<td>IAI</td>
<td>Simulink</td>
<td>Mathworks</td>
<td>Design</td>
<td>Midway</td>
<td>Water</td>
<td>System</td>
</tr>
<tr>
<td>IAI</td>
<td>Rhapsody</td>
<td>IBM</td>
<td>Design</td>
<td>Start</td>
<td>Water</td>
<td>SW</td>
</tr>
<tr>
<td>IAI</td>
<td>Design Manager</td>
<td>IBM</td>
<td>Design</td>
<td>Start</td>
<td>Water</td>
<td>Cross discipline</td>
</tr>
<tr>
<td>IAI</td>
<td>Rational Team Concert</td>
<td>IBM</td>
<td>Task mgmt &amp; CM</td>
<td>Midway</td>
<td>Water</td>
<td>Cross discipline</td>
</tr>
<tr>
<td>IAI</td>
<td>RELM</td>
<td>IBM</td>
<td>Cross tool activities</td>
<td>Midway</td>
<td>Water</td>
<td>Cross discipline</td>
</tr>
<tr>
<td>IAI</td>
<td>MANTRA</td>
<td>IAI</td>
<td>CM</td>
<td>Midway</td>
<td>Water</td>
<td>Cross discipline</td>
</tr>
<tr>
<td>OFFIS</td>
<td>GROOVE</td>
<td>SF</td>
<td>dynamicity</td>
<td>Midway and on</td>
<td>All</td>
<td>Cross discipline</td>
</tr>
<tr>
<td>ALES</td>
<td>DESYRE</td>
<td>ALES</td>
<td>simulation</td>
<td>Midway and on</td>
<td>All</td>
<td>Cross discipline</td>
</tr>
<tr>
<td>INRIA</td>
<td>PLASMA</td>
<td>INRIA</td>
<td>Statistical model checking</td>
<td>All</td>
<td>All</td>
<td>Cross discipline</td>
</tr>
</tbody>
</table>

Version | Status | Date       | Page |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Final</td>
<td>2013-05-06</td>
<td>20</td>
</tr>
</tbody>
</table>
5 User roles in the interoperability environment

There are 4 levels of users in such environment, including the engineers working with tools. These roles are derived from the fundamentals of working in a distributed multi-party environment, and the use of semantic-web technologies at the base of tools interoperability over the tools-net and the semantic mediation approach to tools interoperability.

1. Tool vendors
2. IT network administrators
3. Power semantic-web users
4. System engineers tool users

Details for each of these roles will be further elaborated in the next sections, while some has been already sufficiently described above.

5.1 Tool vendors

These are developers who need to develop adapters for their tools to work with the collaboration environment, selecting any of several integration architectures described in the previous version of this document ([37], chapter 6 “ARCHITECTURE PRINCIPLES OF THE TOOL-NET”), and briefly repeated in this revision below.

5.2 IT Network Administrators

These are administrators responsible for the secured working environment in each of the participating companies in a project. These administrators are responsible for the provision of server machines, installing the products comprising the Jazz server software, configuring it for users internally and external. To accommodate interconnectivity among multiple partners, these administrators need to define networks, establish firewalls, configure communication permissions across these fire-walls, setting up private virtual networks were applicable and yellow-zone machines with proper firewalls as appropriate. The scene where this user operated is schematized in Figure 16 below in section 8.2, with some boxes representing the Jazz and a few tools components. These administrators are also responsible to set up users on the servers, creating project work areas, and so forth as required by the server software – which in our case is a Jazz administration tasks.

5.3 Power Users

These are engineers who complement the tools interoperability by configuring the “semantic-mediation container” (which will be described in the next sections). The interoperability on this platform is configured via ontologies, mediators and mediation rules, using the RDF and OWL languages. The platform is configured into a network of mediators where models flow, after being delivered into the
system from tools and retrieved from the network by the same and other tools, thus exchanging
modeling information.

5.4 System Engineers

These are users of tools. As such, the engineers would each use the tools they are used to work with,
yet each tool may have some extensions and adapters to enable its collaboration over the network. This
means that the user’s environment is extended beyond the isolated tool. Clearly, some tools may already
have some level of networking built into them. For instance, a tool may use some version control and
source management server on the network. Likewise, the DANSE tool-net interoperability platform is a
server to which the tools now connects and with which it need to share models. The specific methods of
communicating with this platform serve as an extension to the user’s working environment she needs to
learn and understand. Moreover, the web nature of the environment dictates the use of web tools such
as browsers to view and browse resources being shared on the platform, which may mean that RDF and
OWL concepts may creep into the engineer’s working environment. That of course depends on the level
of integration built into the tools, which at the minimum would rely on browsing through the internet, with
no adaptation of the tool for that, and at the other end of this scale, the tool may have elaborated built in
means to do the same through its own UI keeping a familiar user experience as with the other aspects of
its working environment.

Within this range of possibilities, the engineer needs at least be aware that she now works in a
collaborative environment in which data is shared over the network, using model mediation according to
the concepts of the DANSE tool-net eco-system.
6 Design modelling semantic integration concepts

Semantic integration in DANSE uses the “semantic mediation” mechanism adopted from the SPRINT [7] project, applied in DANSE to support the tools-net and the tools-chain services needed to implement the DANSE design methodology. We firstly present the essence of this mediation mechanism and the concepts at its base.

Semantic mediation is not just yet another model transformation mechanism, but an attempt to build some structure into the semantic dependencies of models. Therefore, each model is assumed to be an incarnation of some semantic description in ontology. As we said earlier, both models and ontologies are represented in RDF graphs, while the ontology uses the OWL specifications to describe the semantic of a model.

In essence, each modeling tool has its own modeling language, and its own internal representation of the models, not using RDF for that, or OWL to describe that language. We therefore make a significant requirement of tools to have an adapter which can export modeling content of the tool into RDF graph, and also the ability to import such model graphs into its internal representation, and merge that import appropriately into its internal structure in a reasonable way to reflect on any differences among the import and the existing status of that model.

Moreover, the external RDF model is described by ontology. We can assume that there may be multiple ontologies for a given tool. Tools today can be very powerful systems with many useful capabilities and features and can serve a multitude of purposes, each of which can be considered to have its own specific semantics. Moreover, for the purpose of interoperability, only some small portion of that semantic may be

Figure 9: Mediation and links
meaningful, so that this “pie” can be sliced in many ways; a tool may host a collection of languages with various relations to each other such as “extending”, or “included” or “distinct”.

In the context of semantic integration, we consider each such language a “tool” which works with ontology and which exports and (possibly) imports RDF models.

The value of semantic integration is in the ability to mediate a model from one such tool, to another model in a different tool. That process is depicted in the above **Figure 9**, showing the transformation of an instance of a “Car” concept from a Rhapsody language to a Modelica language.

The transformation works by first enriching the input model with concepts from the target ontology, according to rules such as class equivalence, resulting with a rich, yet ambiguous model, serving two languages that are related, but are not the same.

**Note:** Mediations may be implemented in different ways, so that rules as OWL ontologies may not apply to all implementations. In this document, the mediation implemented by the SODIUS partner, uses eCore [2] meta modeling internally, and transformations of such models via the MDW toolkit [48]. The mediation components of MDW are activated in the standard defined for the Semantic Mediation platform, so that its rules are written in a different way than the OWL ontology language.

The next step would be to filter out from this rich model only the concepts which are legal in the target ontology. The interesting thing in this process is that there is no master direction of flow and this process can be described in exactly the same way when models flow from left to right (Rhapsody to Modelica), or from right to left (Modelica to Rhapsody).

This is not necessarily the common solution and mediation can use more directional-sensitive rules as has been applied in the DANSE mediation network by DANSE partner SODIUS using their MDW workbench [48] – see framed note above.

The idea of common semantics is based on the assumption that transforming a model to another language can be meaningful only if that other language has some “semantic overlap” with the original language of that model, as depicted schematically in this next **Figure 10**. The schematic symbolic overlap in the left side can be labeled more meaningfully as shown on the right side of that figure – depicting specific aspects of the overlap such as pertaining to “structure” or to “behavior”.

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Date</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Final</td>
<td>2013-05-06</td>
<td>24 of 48</td>
</tr>
</tbody>
</table>
Note also that the common overlap for all 3 languages (A-B-C on the left side) is labeled as “AM” on the right side to signify the fact that some minimal overlap would always be among languages. “AM” is for the “Architecture Management” specifications of OSLC [22], which defines a single very basic entity – the “Resource” it is clear that such basic element exists in all models, specifically RDFs, where the R stands for Resource.

Is there value to such a low level commonality? Certainly there is, indicating the elements of some model, such which a tool using a different language may need to start with rather than creating a totally new model, as if being a totally unrelated model. Yet, if there is a relation among these models, it would be better to start with a common minimum, such as the items of which the original model consists of. With AM, these resources have some important properties such as the “name”, “description”, date of creation and of modification, perhaps even the identity of a user doing these updates, etc.

The level of overlap among modeling languages creates a hierarchy of semantic commonality and that is shown in the following Figure 11.
Conceptual and architecture principles of SoS design and semantic interoperability of systems platform and the SoS design Tool-Net – 3rd revision

Figure 11: A hierarchy of semantic overlaps among modeling languages

This diagram shows a possible hierarchy using known tools and engineering domains to represent those common semantics for tools that operate in those domains. “Base Semantic” is a term for a very basic concept language that can be common to all that is above it – much like the “AM” language we discussed above.

The same diagram is rendered according to the “hub and spoke” architecture diagram (Figure 3, Figure 6, Figure 7) is shown in the next Figure 12. In both these figures, two example instances for model flows are shown in green and red colors where models are mediated to other models of common semantic through several levels of abstractions. As the model is mediated along these paths, it is abstracted to some maximal common model (in the green case, this is the “Base Semantics” in this diagram, and in the red case this is a “SysML” model) and back up to a specialized domain. It is clear that as a model moves up the “Enrichment” direction, no new information can be applied, but as a model moves down the “Abstraction” direction, it clearly loses the specific information which cannot be expressed at these abstract levels.
Figure 12: Semantic mediation network showing a "hub and spoke" layout of tools that exchange models with a server hosting a network of hierarchical mediations.
7 Architecture principles of the Tool-Net

This part of the document is technically same as that described in [37], chapter 6 “ARCHITECTURE PRINCIPLES OF THE TOOL-NET”.

7.1 Architecture patterns of the Tools-net

Note: This chapter describes the “Product vendor user” role – see 5.3

The architecture patterns for tools integration are depicted in the next Figure 13. While this is a very comprehensive collection of integration options, the one option implemented so far is B and C, whereas option D is a reasonable option for standard tools implementing the OSLC server protocol. That option D requires that the server platform will have a “smart” OSLC Crawler which can access models in the OSLC server tool repository.

![Figure 13: Architecture patterns of the tool-net](image)

The full details of each of these integration patterns are left to be found in the previous document revision [37]. Back to our 4 types of users identified above (4 “System engineers tool users”), this technical part is relevant to tool vendors wishing to integrate into the DANSE tool-net.
8 Tools-net Semantic Mediation Platform and Domain Interoperability Implementation

Note: A new chapter describing prototype implementation includes 6.7 from previous version [37].

The DANSE prototype for serving semantic mediation to Tools-net integrated tools is a Jazz/DM plugin extension which hosts mediators and which manages the required artefacts to perform that mediation. These artefacts include Ports, Mediators, Ontologies and Rules. These terms and the management of these artefacts within a Jazz/DM extension as well as the tools that are so far integrated with it are described herein. We term this extension “Semantic Mediation Container” or in short “SM Container”. The present implementation relies on a simple “Interceptor” software design pattern in which mediators are intercepted by the container framework and activated in the proper order, whereas input and output of the mediators are managed by the container framework.

We than elaborate on tools interoperability with the SM Container which presently include a subset of the Rhapsody SysML modeling language from the IBM team, and from SODIUS the Rhapsody UPDM language and Systems Architect NAF language.

8.1 The SM Container

The SM container is a Jazz/DM plugin extension whose architecture is illustrated in Figure 14 next.

![Figure 14: SM Container architecture.](image)

The IBM product components are of the Jazz/DM package and some possibly additional product from the Jazz family – see Figure 15. The Jazz Team Server (JTS) provides the repository services over triples (RDF) store, whose services are provided via RESTful protocols. The DM Jazz application comes with the “Domains” feature which includes the DTK – Domain Tool Kit – to define ontologies and model instances of
these ontologies. The domains in DM are artifacts which are created by 3rd parties in this eco-system such as tool vendors acting to integrate their tools into this environment. RELM (Rational Enterprise Lifecycle Management) is a tool to index and perform impact analysis over resources managed by other tools, such as the DM domains and other entities in the DANSE SD Container extensions. VVC (Version, Variants & Configuration) provides an OSLC interface to resources managed in DM domains. That feature provides models in DM to be OSLC accessible and thus participate in this inter-operability standard.

The green components are developed in the project and consist of the SM Container. It consists of the plugin to DM in which mediators that are developed by 3rd parties contributing to the semantic mediation capability of the container, the management module and UI for that module, and a service module which serves RESTful requests from integrated tools.

Tools, mediators and domains are contribution of 3rd parties. These may come from different tool vendors and proposers in the DANSE consortium, as well as other parties that want to reuse the semantic mediation capability and technology.

8.2 Distribution

Note: This chapter describes the “IT Network Administrator” role – see 5.3

Distribution of the SM Container is an important capability in a multi user environment, and the modern multi-party collaboration which extends the boundaries of a single corporation. There are network management aspects to that which can make the exchange of information possible. The next section describes how mediation of models is done on a single server at some corporate site. That single server configuration can server also partners across the internet, or it may be working in cooperation with other SM containers, across the internet. In both cases, access between clients and servers as well as between servers works across internet zones known as:

- Intranet – internal companies internets which are protected behind “fire walls”
- Internet – the open internet where access restrictions do not exist and any hostile party can try to interfere and cause damages and infringe IP rights.
• Yellow-zones which are internet domains between these two and in which machines are protected behind firewalls, but are open to direct internet access by clients.

To facilitate the connectivity we require, a typical configuration is depicted in the next Figure 16, showing the SM Containers on both sides residing in a Yellow Zone, working with clients and local servers over the protected intranet of the respective companies, and within the protection of the fire-walls, working through “drills” of opened access paths through them that go between these two sites through the hostile internet using the secured internet SSL protocol.

That kind of configuration requires the involvement of the internet security officers of the IT networking administration of each corporation and once solved, work can proceed.

![Figure 16](image)

**Figure 16**: Network considerations for the distribution SM Container.

### 8.2.1 Components of the SM Container

**Note**: This chapter describes the “Power user” role – see 5.3

The components of the SM Container are illustrated within a specific typical configuration of mediation flows in that container in the following Figure 17:
Figure 17: SM Container components and typical configuration with two possible flows.

There are 4 types of components in the container, and the external components – the tools. As follows:

1. Ports – sinks and sources of a mediation or interaction with tools. A port is associated with an ontology which defines the language of models that are provided and/or accepted by it. There are 4 types of ports:
   
a. POST port (RESTful) to which a tool may post an RDF model. That is when a tool does **export** to the SM Container. This port can only serve as a source port for a mediator. It is always also the sink port for a tool.

   b. GET port (RESTful) from which a tool may obtain an RDF model. That is when a tool does **import** from the SM Container. This port can only serve as a sink port for a mediator. It is always also the source port for a tool.

   c. Repository port is a persistency service which can hold RDF models of a certain ontology (or language). That ontology may also be a domain in DM, in which case this association is managed by DM and its DTK. Repository port can serve both as a source and as a sink for mediators. A repository port can be implemented as the instance of a DM domain. When this is the case, then

   d. Pipe port is a non-persisting repository which serves to connect consecutive stages in a complex mediation flow in the SM Container. Can serve as both a sink and a source for mediators, and only to a single mediator in each side, each portraying inverse role (relative to the other) per this port on each side.

2. Mediators – actors in the mediation flow as described at length in section 6. A mediator is a Java class which implements an interface which corresponds to the schematic in Figure 18 below. When activated by the container framework, it will perform mediation of the input model (coming from its
source port), to the output model (going to its sink port), according to ontologies and rules associated with it in the container.

**Figure 18**: The mediator execution environment mediating a in input model from an input port, to an output model in an output port.

3. Ontologies – RDF ontologies which define modeling languages, and are associated with ports thus defining what kind of models can be handled through a port. When a port is a domain repository in DM, its ontology is also managed by DM which also keeps track of the validity of the domain model data and the domain ontology.

4. Rules-sets – RDF ontology, but may also represent other means of defining rules. An ontology based rule can use class equivalence, property relations, class relations and similar OWL restrictions as rules. Other technologies to apply rules on RDF models can also be defined. Presently, that requires RDF format. This may change in the future.

In this typical configuration of **Figure 17**, elements are tagged and named as follows:

- **PA** – Post port for tool A – using ontology OA
- **GA** – Get tool for tool A – using ontology OA
- **OA** – Ontology of tool A
- **MIA** – Mediator for tool A import – using rules-set RIA. In this example, the ports on both sides of this mediator use the same ontology. We have a special trivial mediator we term “the null mediator” to take care of such “mediations”.
- **MEA** – Mediator for tool A export – using rules-set REA. Here, the mediator is also working (like MIA) with the same language, yet, this mediator can also accept the URL of a resource considered the “root” of a substructure in the model being exported. That is the “Extractor” mediator.
- **RIA** – Rules-set for tool A import mediator MIA. In particular, this rules-set is empty since the null mediator does not interpret any rules nor apply any mediation algorithm within.
- **REA** – Rules-set for tool A export mediator MEA.
- RC – Common repository C such as the BSO (Basic Structured Ontology) that can serve a common subset of the A and C repository languages (according to OA and OB).
- OC – Ontology of common repository C – representing a common subset of the OA and OB ontologies which can be mediated to each other.
- MIC2A – Mediator for import from common C to A, from ontology OC to ontology OA.
- RIC2A – Rules-set for mediator MIC2A, to import from common C to A, from ontology OC to ontology OA.
- MEA2C – Mediator for export from tool A to common C, from ontology OA to ontology OA.
- REA2C – Rules-set for mediator MEA2C, to export from tool A to common C, from ontology OA to ontology OA.

The SM Container is managed via a UI module in which the configuration of these artifacts is done. A glimpse at this module as a web UI is shown in Figure 19, which shows 4 sections each of which olds a row per item, be it ontology stored on the server, a rules-set ontology stored there, a port or a mediator.

Ports that are RESTful post/get ports are installed as service entry point into which client tools can issue requests. Repository ports can be browsed via the web interface, and mediators can also be tested on a variety of input data. Mediators connecting two repositories maintain also the relations among the mediated resources. This relation can be browsed and queries and servers as an implementation of linked data [27] internally in the platform.

![Figure 19: SM container configuration UI such as for the above example.](image)
Another web interface is available that works within the Jazz/DM web interface as shown in the next Figure 20.

Figure 20: SM Container configuration UI in a Jazz/DM page

8.3 Operation of the environment

Note: This chapter describes the “Systems Engineer user” role – see 5.3

The next few slides describe the mediation process which the Engineer working with the system is aware of. This process includes the working at the tool GUI extension, and tracing the mediation process as it is executed. The example is small demonstration with three repositories: a common repository and two specific repositories of the tools with a Rhapsody and a System Architect tool, mediating models in UPDM family, with a mediator developed by partner SODIUS, using the MDW model-transformation toolkit.

1. Exporting a NAF model from System Architect (see Figure 21). Engineer is aware of the modeling environment on the platform and the ports through which a model can be exported and shared, as being set up by the power user of the platform.

2. The resulted model is mediated into a repository which is OSLC compliant and can be also browsed to validate its content by the engineer, doing that over the web with a browser as shown in Figure 22.

3. The mediation process is also shown schematically (see Figure 23) as a process in which models are transformed between repositories, each defined according to a different ontology, but which are
Conceptual and architecture principles of SoS design and semantic interoperability of systems platform and the SoS design Tool-Net – 3rd revision

linked via a mediator. The progress of the mediation is reflected on the web console as repositories are updated - **Figure 24**.

![Image](image1.png)

**Figure 21**: System Architect exports a model to a specific RESTful POST port in the platform configuration.

![Image](image2.png)

**Figure 22**: The model being mediated and stored in a repository can be browsed via the web.
Figure 23: A multi-staged mediation process.

Figure 24: Progress of the mediation reflected on the status of repositories in the web console.

4. Repository browsing as seen in Figure 22, has more browsing options such as the RDF content of the model (Figure 25), in which every resource can be followed through the related modeling resources via its properties, comparison of RDF revisions as the model is updated through mediations (Figure 26) and can be queried via the SPARQL query language of RDFs (Figure 27).

Figure 25: Browsing a repository RDF in which resources can be clicked and reviewed.
Conceptual and architecture principles of SoS design and semantic interoperability of systems platform and the SoS design Tool-Net – 3rd revision

Figure 26: Comparing versions of the model in a repository as it is being changed through mediations.

Figure 27: Applying SPARQL on an RDF model.

5. Resources in all repositories which are connected via mediation as well as the resource in the tools being exchanged with the platform are all linked to each other and that linkage can at present be viewed through the web browser. Future development in which these associations are represented
in an RDF data model will enable querying them as well, resulting with a holistic view over the entire extent of the models managed on the platform. The linked data maintained on the platform is depicted in **Figure 28**.

**Figure 28**: Linked data information can be browsed for each individual mediation link among pairs of repositories.
9 Conclusions

The work on the Tools net for DANSE is far from conclusion, but an initial possibilities for interoperability with selected tools start to emerge. At the time of publishing this revision, which is M18 of the project, the project is also reporting on the first prototype in a different document [49]. There is a set of tools by vendors some of which are extended with an interface to the collaboration eco-system, but important other tools are not yet connected. The adoption of the collaboration concepts of using ontologies needs to sink in with the vendors and real use cases need to be tried out so that the value of this ecosystem interoperability is appreciated.

At this point in time, the platform described in this document is installed and used by most partners, and about half of the users. More visibility of this technology is needed within partners and application of the concepts within their work procedures.

As we extend the number of tools and use cases in which the tools are used, the capabilities of the mediation will become more evident as well as its pain points and weaknesses. These will be important lessons to be learned as the technology adopted in DANSE for this purpose is further pursued in DANSE, its follow-ups and other relevant EU projects for which this technology seems to be of value.
10 Thanks

Thanks to WP leaders and participants who provided background material for the respective chapter of this document, and the DANSE members who will participate in the review sessions.
11 Abbreviations and Definitions

**ASDI**  
Aircraft Situation Display to Industry

**Application**  
A software program that provides added value on top of tools by applying functions that have not been addressed by individual tools and that are possible due to the integration of data from multiple tools.

Applications that add a new value to the data in the TOOLNET repository are referred to as AVAs – Added Value Applications.

**Data scoping**  
When shared, we distinguish several levels of scoping in data, such as private, internal, and public. Reasons for data scoping may be protection of rights as well as technical such as proprietary information. There may be more categories, yet presently we can discuss only these three levels:

- **Private**  
  Data that is located on and managed only by the tool. It may be available to applications by accessing the tool via some standard API (such as OSLC).

- **Internal**  
  Data that is shared and may be enriched to match a certain level of compatibility with the information bus, but is not shared with other partners.

- **Public**  
  Data that is shared with other partners.

**Data sharing**  
For specific tools, the data for a certain engineered system can be shared with other tools and applications. When data is shared, it is "exported" to the TOOLNET since we assume the only way to share the data is via the information bus implemented by the TOOLNET.

**DEE**  
The DANSE Engineering Environment consisting of the following:

- **Tool Net**  
The Tools Interoperability facilities and the Integration platform.

- **SSI**  
The Semantic Services Integration layer of the ToolNet platform

**DODAF**  
Department of Defence Architectural Framework

**DM**  
Design Management application of Jazz. Used to define modeling domains and provide visualization over the web of corresponding modeling data.
Conceptual and architecture principles of SoS design and
semantic interoperability of systems platform and the SoS
design Tool-Net – 3rd revision

**DTK**  
Design management ToolKit. Used for developing new ontology
meta-models (domains) in the DM

**Elements**  
Nodes constituting the model data of a project. The model also
consists of relations between these elements.

**Enrichment**  
Tool data when exposed and exported to the TOOLNET for
sharing must be enriched to integrate with data from other tools
serving the same developed system. Enrichment depends on
the applications intended to use that data; as new applications
are developed and enhanced, the requirements from the
enrichment function may change.

**GIS**  
Geographic Information System

**JIA**  
Jazz Integration Architecture lays out the architecture for
integrating services and application within the Jazz framework.

**JTS**  
Jazz Team Server is the core services provider of the Jazz
platform

**HTTP**  
Hypertext Transfer Protocol is the communication protocol over
the Internet which is used to connect Web clients (browsers and
applications) and servers.

**HRC**  
Heterogeneous Rich Components

**Links**  
Relations between element nodes in a model are known as
links. There are two kinds of links:

- **Intra-links**  
  Internal relations between elements of a model emanating from a single tool
  instance. These are natural links defined in that tool or such that are introduced or
  modified during the enrichment.

- **Inter-links**  
  Relations between elements originating in models from different tools or from
different projects. These relations can only result from enrichment, either
during data exportation (publishing) or during enrichments taking place in the
TOOLNET, using automatic or manual tools.

**MODAF**  
Ministry of Defense Architectural Framework

**NAF**  
NATO Architectural Framework

**OAUTH**  
An Authentication protocol that is used by Jazz to provide
secured interaction over the internet of users and the Jazz
platforms.
Open Services for Lifecycle Collaboration (also known as OSLC or Open Services) is a community and set of specifications for Linked Lifecycle Data. The community’s goal is to help product and software delivery teams by making it easier to use lifecycle tools in combination.

See: http://open-services.net/html/Home.html

OWA Open World Assumption
OSLC See: Open Services for Lifecycle Collaboration
OSLC-AM The Architecture domain of the OSLC specifications.
OSLC-CM The Change Management domain of the OSLC specifications.
PD Physical Device
OSLC-RM The Requirement Management domain of the OSLC specifications.
Project A component that is engineered collectively over a set of tools, and which is subject to processing by some of the applications. It must be clearly identified across TOOLNET and all the relevant tools.
Project Publishing Exporting project-related data stored in a certain tool into the TOOLNET. This mechanism also includes an enrichment function.
Resource Identified element or relation in any model data that is stored in the TOOLNET and which can identify back the original element in the originating tool. Some of the resources are generated by the TOOLNET – such as enriched data: elements and links. A resource has a single owner.
Resource Description Framework It’s a family of W3C specifications for conceptual description or modelling of information that is implemented in web resources. (See: http://www.w3.org/TR/rdf-primer/)
RDF See: Resource Description Framework
RTP Reference Technology Platform (of CESAR)
SDK Software Development Kit. Related to the development environment of services over Jazz/DM.

Semantic Mediation The transformation of model data between models according to the semantics of the modeling languages, and in an incomplete way according with the Open World Assumption (OWA).
SM Container A Jazz/DM plugin which executes mediation flow paths through mediators to carry out the semantic mediation task of model collaboration in the DANSE tools-net eco-system.
**SPARQL**

SPARQL (SPARQL Protocol And RDF Query Language) is a query language for RDF. [http://www.w3.org/TR/rdf-sparql-query/](http://www.w3.org/TR/rdf-sparql-query/)

**SysML**

Systems Markup Language

**Tool**

A software program that models some aspects of a product’s design. Tools have internal models of the design and can serve as part of a group of tools that together serve the full engineering process. However, used by itself, a tool is also an independent program with its own data repository and management and usability functions that allow users to work with it totally independent of other tools. A tool generally is said to hold some information about the engineered system.

**Tool data**

A model based on a well-defined meta-model that defines a certain aspect of an engineered system. For instance, the aspect can be the functional requirements of the product, and the model must be detailed enough so that each requirement can be assigned to a specific component of the system. Meta-models can also associate additional information such as the relations (structural, logical, or geometrical) between the components.

**Tools/Data isolation**

A mechanism that implements a set of rules for the access permissions by applications to certain portions of the tools' public data. Note that while access control may not be needed to functionally implement TOOLNET, it is a mandatory property of an TOOLNET that can be used commercially to collaborate between distinct private vendors.

**UML**

Unified Markup Language

**VVC**

Version, Variants & Configuration
### 12 References

<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[10] UPDM</td>
<td>&quot;Unified Profile for DoDAF and MoDAF“</td>
</tr>
<tr>
<td>Reference</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>[27] Linked Data</td>
<td>“Linked Data”, W3C Standard, <a href="http://www.w3.org/standards/semanticweb/data">http://www.w3.org/standards/semanticweb/data</a></td>
</tr>
<tr>
<td>[31] D3.1</td>
<td>D3.1 &quot;Project Requirements&quot;</td>
</tr>
<tr>
<td>[37] D8.2.2</td>
<td>&quot;D8.2.2: Conceptual and architecture principles of SoS design and semantic interoperability of systems platform and SoS design Tool-Net,&quot; DANSE delivery</td>
</tr>
</tbody>
</table>
Conceptual and architecture principles of SoS design and semantic interoperability of systems platform and the SoS design Tool-Net – 3rd revision

<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[38] Wisnowsky</td>
<td>Dennis Wisnosky, DOD BMA CTO &amp; Chief Architect on the Business Mission Area</td>
</tr>
<tr>
<td>[39] GROOVE</td>
<td>GRaphs for Object-Oriented VErification (GROOVE), <a href="http://groove.sourceforge.net/groove-index.html">http://groove.sourceforge.net/groove-index.html</a></td>
</tr>
<tr>
<td>[44] DM</td>
<td>Jazz Design Management <a href="https://jazz.net/products/design-management/">https://jazz.net/products/design-management/</a></td>
</tr>
<tr>
<td>[47] CLM</td>
<td>Jazz Collaborative Lifecycle Management (CLM) <a href="https://jazz.net/products/clm/">https://jazz.net/products/clm/</a></td>
</tr>
<tr>
<td>[48] MDW</td>
<td>MDWorkbench - The Eclipse-based powerful IDE by SODIUS ® <a href="http://sodius.com/mdworkbench">http://sodius.com/mdworkbench</a></td>
</tr>
<tr>
<td>[49] D8.3</td>
<td>DANSE Prototype I deliverable D8.3.</td>
</tr>
</tbody>
</table>