ECSEL Joint Undertaking

European Commission

ECSEL Research and Innovation actions (RIA)

AMASS

Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems

Baseline and Requirements for Seamless Interoperability
D5.1

Work Package:	WP5: Seamless Interoperability
Dissemination level:	PU (Public)
Status:	Final
Date:	30 September 2016
Responsible partner:	Jan Mauersberger (KMT)
Contact information:	jan.mauersberger@kpit.com
Document reference:	AMASS_D5.1_WP5_KMT_1.0

Contributors

Names	Organisation
Jan Mauersberger, Marc Born, Sascha Baumgart	KMT
Tomáš Kratochvíla	HON
Jose Luis de la Vara, Jose María Álvarez, Eugenia Parra, Gonzalo Génova, Juan Llorens, Miguel Téllez	UC3
Barbara Gallina	MDH
Pietro Braghieri	FBK
Bernard Botella, Morayo Adedjouma	CEA
Angel López	TEC
Jose Miguel Fuentes, Luis María Alonso, Julio Encinas	TRC

Reviewers

Names	Organisation
[Peer Reviewer] Irfan Sljivo	MDH
[Peer Reviewer] Fredrik Warg	SP
Barbara Gallina	MDH
Huascar Espinoza	TEC
Stefano Puri	INT

TABLE OF CONTENTS

List of Figures

List of Tables

Executive Summary

This document (D5.1; Baseline and Requirements for Seamless Interoperability) presents the outcome of the AMASS task T5.1 (Consolidation of Current Approach for Seamless Interoperability), whose main objective was to analyse state-of-art and the state-of-the-practice approaches for seamless interoperability of tools, as well as to analyse gaps that may exist and are candidates for work in the scope of WP5 of AMASS. This document contributes requirements towards the general AMASS architecture and platform.

The investigation regarding seamless interoperability in this deliverable has been focused on existing approaches already available in many tools, such as proprietary file-based data exchange, data exchange in standardized formats (e.g. XML), and direct tool coupling via interface technologies. Furthermore, a number of new and upcoming paradigms have been analysed, e.g. OSLC, Model Bus, and automation and integration hubs.

In addition to this traditional tool coupling aspects (i.e. data exchange between tools), it has also been investigated how “seamless tools” can be used from a user perspective, e.g. in team environments. That means that, for example, the tools should support the intended use case in a way which requires as little user interaction as possible, e.g. without asking the user if the information can be guessed from the context. Many nice examples for this aspect of seamless can be observed in modern smartphones, such as calling a contact by just moving the phone to the ear without pressing the call button. It also means that the user should not be requested to perform tasks, especially those that might be beyond his expertise. Such tasks include difficult installation procedures or difficult merge decisions of complex models in team contexts, among others. One answer to these kind of problems could be the usage of web- and cloud-based approaches with no effort for installation and with an immediate sharing of data among all connected users.

The investigations in T5.1 for the seamless interoperability aspects of tools considered existing results from ongoing and past research projects, as well as available technologies in the market such as the OPENCOSS platform, OSLC, and cloud/web services in general; but also existing safety tools, e.g. medini analyze. Practical experiments have been executed to obtain more sound results when necessary. For each investigated tool, approach, or technology, gaps have been derived which led to a number of requirements upon seamless interoperability. The derived requirements, which constitute the way forward for WP5, include:

- Support for seamless traceability of information
- Seamless and intermediate work in a team
- Avoidance of any data copies
- Support of the whole process activities by the integrated tools
- Reducing the complexity of using multiple tools for different purposes
- Providing tool usage interfaces dedicated to the user’s role
- Generation of meaningful reports covering multiple information of many activities

The derived requirements will be fed into WP2, consolidated, and later on used for the conceptual and implementation work in the project, as well as for validation. For example, the results and insights presented in D5.1 will be used in WP5 a basis for the preparation of D5.2 (Design of the AMASS tools and methods for seamless interoperability (a)) and D5.4 (Prototype for seamless interoperability (a)), and of the deliverables that correspond to their evolution (D5.3 for D5.2, and D5.5 and D5.6 for D5.4).

1. Introduction

Embedded systems have significantly increased in number, technical complexity, and sophistication toward open, interconnected, networked systems (such as "the connected car" and the cloud). This has brought a “cyber-physical” dimension with it, exacerbating the problem of ensuring safety, security, availability, robustness and reliability in the presence of human, environmental and technological risks. Furthermore, the products into which these Cyber-Physical Systems (CPS) are integrated (e.g. aircrafts) need to respect applicable standards for assurance and in some areas they even need certification. The dimension of the certification issue becomes clear if we look at the passenger plane B 787 as a recent example – it has been reported that the certification process lasted 8 years and has consumed 200.000 staff hours at the FAA just for technical work. The staff hours of the manufacturer even exceeded this figure, as more than 1500 regulations had to be fulfilled, with evidence reflected onto 4000+ documents. Although aircrafts are an extremely safety-critical product with many of such regulations, the situation in other areas (railway, automotive, medical devices etc.) is similar.

To face all these challenges, the AMASS approach focuses on the development and consolidation of an open and holistic assurance and certification framework for cyber physical systems, which constitutes the evolution of the OPENCOSS and SafeCer approaches towards an architecture-driven, multi-concern assurance, and seamlessly interoperable tool platform.

The AMASS tangible expected results are:

a) **The AMASS Reference Tool Architecture**, which will extend the OPENCOSS and SafeCer conceptual, modelling and methodological frameworks for architecture-driven and multi-concern assurance, as well as for further cross-domain and intra-domain reuse capabilities and seamless interoperability mechanisms (based on OSLC specifications).
b) **The AMASS Open Tool Platform**, which will correspond to a collaborative tool environment that supports CPS assurance and certification. This platform represents a concrete implementation of the AMASS Reference Tool Architecture, with a capability for evolution and adaptation, which will be released as an open technological solution by the AMASS project. AMASS openness is based on both standard OSLC APIs with external tools (e.g. engineering tools including V&V tools) and on open-source release of the AMASS building blocks.
c) **The Open AMASS Community**, which will manage the project outcomes, for maintenance, evolution and industrialization. The Open Community will be supported by governance board, rules, policies, and quality models. This includes support for AMASS base tools (tool infrastructure for database and access management, among others) and extension tools (enriching AMASS functionality). As Eclipse Foundation is part of the AMASS consortium, the Polarsys/Eclipse community [1] is a strong candidate to host AMASS.

To achieve the AMASS results, and as depicted in Figure 1, the multiple challenges and corresponding project scientific and technical objectives are addressed by different WPs.

WP5 (Seamless Interoperability) roughly aims at tool interoperability. More specifically, with respect to the AMASS goals, the WP deals with the problem and solution related to AMASS goal G4 and the corresponding project objective O3:

- G4: to demonstrate a potential sustainable impact in CPS industry by increasing the harmonization and interoperability of assurance and certification/qualification tool technologies by 60%
- O3: to develop a fully-fledged open tool platform that will allow developers and other assurance stakeholders to guarantee seamless interoperability of the platform with other tools used in the development of CPSs

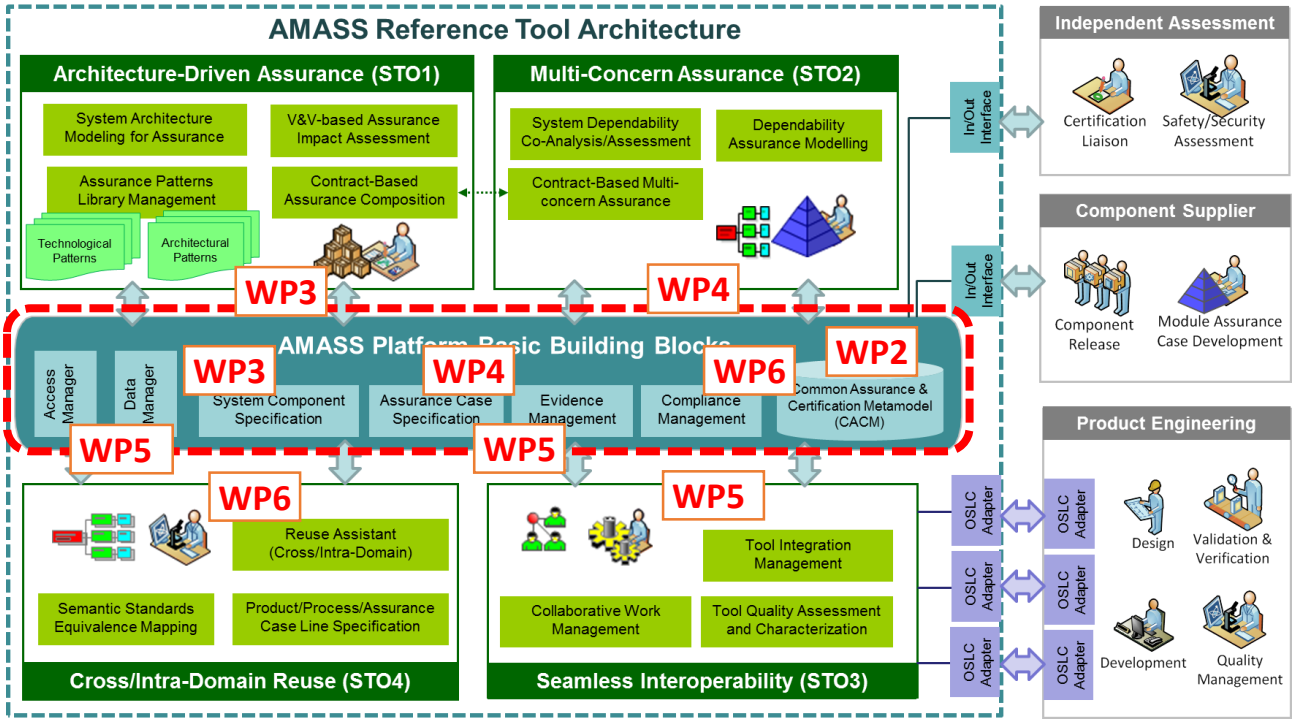

Figure 1. AMASS building blocks

WP5 shall investigate and provide an open and generically applicable approach to ensure the interoperability between the tools used in the modelling, analysis, and development of CPS, among other possible engineering activities. The WP will address interoperability from an assurance and certification-specific perspective, and the resulting approach will further aim to support collaborative work among the stakeholders of the assurance and certification of CPS. This will facilitate the determination of the consequences of the use of a given engineering tool (e.g., based on its available qualification information and documentation), and to ensure that the integrated information makes CPS certification possible. In addition, WP5 is responsible for consolidating previous work on evidence management in order to design and implement the basic building block called 'Evidence Management' (Figure 1). WP5 will also take care of the 'Access Manager' and 'Data Manager' basic building blocks.

This document is deliverable D5.1, the first WP5 deliverable. The deliverable contributes to the WP5 overall objective regarding the definition of the baseline for an intelligent, automated, and highly customizable tool infrastructure for seamless interoperability and for its management. To this end, the deliverable analyses the state of the art concerning seamless interoperability for the adoption of the best features from existing approaches and for enabling and guarantying compatibility. Results from ongoing and past projects and initiatives have been studied, as well as available technology in the market. The initial candidates for the investigation include the OPENCOSS platform, Cloud/Web Services, and OSLC, among others. Based on the study of the approaches, gaps have been identified. These gaps have allowed us to identify requirements to achieve the seamless interoperability vision of AMASS and to propose a consolidation and way forward for WP5 work.

The rest of the deliverable is organised as follows. Section 2 presents a problem description and background information for WP5. Section 3 analyses state-of-the-art technologies, approaches, and architectures for seamless interoperability, in order to identify gaps. Section 4 reports on the experience from current practice, in contrast to state-of-art technologies, to gain further insights into possible gaps. Finally, Section 5 presents our main conclusions.

2. Problem Statement and Concerns

This section introduces the main motivation of the work on seamless interoperability in AMASS and different aspects of seamless interoperability.

Further insights into the motivation and needs for seamless interoperability are expected to be gained from the formalization of AMASS case studies. Their description will be presented in D1.1 (Case studies description and business impact). Other needs could be derived from D2.1 (Business cases and high-level requirements). Possible updates on the motivation and needs to tackle for seamless interoperability will be determined and presented in future WP5 deliverables.

2.1 Motivation for Seamless Interoperability

WP5 aims at guaranteeing the interoperability of the AMASS tool framework with tools used in the development of safety-critical systems, whereby evidence can be manually generated or automatically generated by the tools themselves (code generators, testing tools, safety analysis tools, etc.). The challenge is to be able to gather evidence from different types of tools by means of standardized and well-defined adapters or exchange tools. There are some axes in this direction that can considerably improve the opportunities of AMASS adoption.

Tool Integration Management
AMASS will deal with the problem that assurance information is present at each development phase (concept, design, implementation, testing, validation) and multiple different tools involved at each phase. The AMASS tool framework needs to interwork with each of these tools. One promising approach is to use OSLC, by e.g. extending it to assurance aspects (safety, security, etc.). As part of this work, the AMASS consortium plans to reuse existing results from the Crystal (http://www.crystal-artemis.eu/) and MBAT projects (http://www.mbat-artemis.eu/) for OSLC-based tool interoperability, since many of their partners are also in AMASS. The data models for tool integration will be also part of the AMASS CACM metamodel.

Collaborative Work Management
We mean supply chain and collaborative issues when developing, assuring and certifying CPSs. Examples of these issues include DIA definition (ISO 26262 OEM-Supplier interaction definition), the use of a platform to exchange safety related information, information exchange potentially via (private) cloud-based collaboration services, related compositional, versioning, and update problems, related security and scalability problems, and provision of server-side services (like intelligent search and cross project consistency checks).

Tool Quality Assessment and Characterization
The development and verification of safety-critical systems increasingly relies on the use of tools which automate, replace, or supplement complex verification and development tasks. The safety of such systems can be jeopardised if the tools fail. To mitigate this risk, safety standards (e.g. DO-178C/DO330, IEC 61508, and ISO 26262) define tool qualification processes, including tool characterization. Compliance with these processes can be required for (re-)certification purposes. Within SafeCer, a tool qualification process line was investigated in order to reduce time and cost via reuse. Within AMASS, this exploratory work will be deepened and broadened to consider also AMASS-related tool chains.

2.2 Aspects of Seamless Interoperability

Seamless simply means to have “no seams”. Originated in sewing pieces of clothes together, it means to join them together without or *as if without* stiches. Figuratively we could say that it denotes an integration or interoperation of two parts or *things* so that the seams are not visible or at least smoothly continuous. There are several more aspects here, since for example an integration already starts when bringing things

together, not just later when using them together. Systems and tools that interoperate first have to come together – the less effort is required and the better their integration capabilities are, *the less seams* will be there later.

The following sub-sections present the main aspects of seamless interoperability that AMASS currently envisions to address.

2.2.1 Seamless Tool Integration

Tool vendors typically develop tools for a specific domain, solving a dedicated problem or problem area. Their main interest is to extend and improve the functionality to cover most aspects of the problem domain. However, tools are never used isolated so customers require integrating these tools with each other. Based on this customer demand and on the fact that an integration with another tools is (or was) often not the major interest of a tool vendor, typical point-to-point integrations are developed. The tool vendor and the final user are forced to support or use different communication protocols, different data exchange formats, different handling and more.

Section 4 reports on experiences with typical point-to-point integration issues. As a possible alternative, frameworks such as OSLC have started to address these issues via standardised, harmonised integration means. These frameworks are presented in Section 3.

2.2.2 Seamless Team Collaboration

When it comes to team collaboration, i.e. multiple people working on the same data, a big aspect behind "seamless" tooling is the management and handling of changes, created by multiple sources potentially distributed at the same time or time shifted. These problems are independent of the domain or abstraction level, thus valid whether working on a shared text document or on hardware schematics.

Different concerns regarding the tool architecture have to be taken in account when team collaboration is considered. It may either support parallel changes on local copies of artefacts, a kind of "check-in" and "check-out" philosophy. Changes are made on local copies of the data which requires either fine-grained locking or sophisticated compare and merge capabilities later. An often-followed approach is to support parallel work using a central server-based project repository, typically supporting arbitrary parallel changes using transactions. This way conflicts are avoided or immediately detected and propagated. Tools must be always on and connected to the server. New approaches as the Google Docs apps bring both worlds together. Documents are stored centrally while still being edited by multiple persons either at the same time or being offline.

Section 3 presents our experiences with collaborative approaches.

2.2.3 Seamless User Interface

When talking about seamless user interfaces, we do not address the demand for UI guidelines or rules for UI design that make a tool look coherent or as *made from the same mould*. That is typically a quality assurance or design issue.

What we address here are gaps or *seams* in the diversified tool landscape. Users typically are not using just one tool to perform their work. They instead have to be familiar with different applications when linking data together. Information constantly has to be collected from many different (engineering) tools, e.g. a safety analysis tool, a requirements management tool, and a system modelling tool, each one addressing a specific domain or problem and having a dedicated user interface that fits the need of that tool. Tool users in such environments need to be familiar with different applications when linking data, and such familiarity might ne not easy to gain or might require specific competences or training.

Here typically the tool user feels or sees the seams in various ways: the communication between tools for example might be very slow (data transfers like importing requirements) such that the integration becomes

obvious. Data collection most often is a synonym for manual information import/export and sharing of data and derived tasks (e.g. search and consistency assurance). Exporting data from one tool just to import it into another tools is most often felt as an annoying "tool gap". In addition, they are represented differently in the importing tool so recognition and further work with the data is made more difficult.

The analysis work on seamless UI that we have performed is reported in Section 3.

3. State of the Art

This section provides an overview of the state of the art concerning different aspects of seamless interoperability as presented in the previous section. We have investigated a broad range of technologies that enhance seamless interoperability. They are related to data storage and data access, creation of user interfaces in web environments, the synchronization of distributed model data, and team collaboration. Furthermore, context-aware UIs reduce complexity in user interfaces as well as the amount of user interactions. Lastly, evidence management and traceability require consistent data across tool borders, therefore we analyse implications of seamless tool integration on model data structures. An overview of tool qualification is also included.

3.1 File-based solutions

3.1.1 Overview

Eclipse stores the models in form of local files by default. This way has a good edition performance but it can difficult the collaborative edition of models as needed in AMASS.

For sure, it could be possible to create this kind of functionality from scratch but a huge effort would be needed. At least some compare, merge and synchronize features would be needed to allow the users to work with the same models without problems of losing data.

The Eclipse EMFStore framework [2] can be used to facilitate this task. This Eclipse technology allows the offline collaboration when the models are stored as local files. It is very similar to source code versioning systems such as SVN. EMFStore provides features for collaboration such as synchronization of clients with server, conflict detection, interactive merging and authentication and authorization. In addition, it provides a versioning system with branching, tagging and version comparison capabilities.

EMFStore has a default user interfaces for almost every available functionality that could be extended, customized and replaced as required according the AMASS needs. This framework has an API that allows us to perform most of the functionality that can be access with the provided UI programmatically.

The EMFStore Server can be run in standalone/non-Eclipse environment.

3.1.2 Experiments

The performed experiments consisted in the integration of EMFStore with the current OPENCOSS tool version.

The first step was the installation of EMFStore in the same development environment used by OPENCOSS (Eclipse Kepler and Java 1.8). The first challenge was to make EMFStore work properly in this Eclipse version.

Following the EMFStore tutorial (published in the EMFStore Site), we succeeded to make it work and we tested all the functionalities related to editing models and sharing models with other EMFStore clients.

Later, we used EMFStore tutorials applied to OPENCOSS models without problems.

Finally, we tried to make EMFStore and OPENCOSS work at the same time but separately. The problem at this point was that when launching EMFStore as “org.eclipse.platform.ide” (the same way as used in OPENCOSS) instead of “org.eclipse.emf.ecp.application.e3.application” (as the EMFStore documentation states), the model editors have not been generated by EMFStore (error showed in Figure 2).

```
Eclipse Application [Eclipse Application] C:\Program Files\Java\jdk1.8.0_31\bin\javaw.exe (14/09/2016 11:45:54)

!ENTRY org.eclipse.emf.ecp.ui 4 0 2016-09-14 11:47:49.826
!MESSAGE null argument:
!STACK 0
org.eclipse.core.runtime.AssertionFailedException: null argument:
        at org.eclipse.core.runtime.Assert.isNotNull(Assert.java:85)
        at org.eclipse.core.runtime.Assert.isNotNull(Assert.java:73)
        at org.eclipse.e4.ui.internal.workbench.PartServiceImpl.showPart(PartServiceImpl.java:1026)
        at org.eclipse.e4.ui.internal.workbench.ApplicationPartServiceImpl.showPart(ApplicationPartServiceImpl.java:111)
        at org.eclipse.emf.ecp.application.e4.editor.E4ModelElementOpener.openModelElement(E4ModelElementOpener.java:59)
        at org.eclipse.emf.ecp.internal.ui.util.ECPHandlerHelper.openModelElement(ECPHandlerHelper.java:445)
        at org.eclipse.emf.ecp.internal.ui.util.ECPHandlerHelper.addModelElement(ECPHandlerHelper.java:203)
        at org.eclipse.emf.ecp.ui.commands.NewModelElementWizardHandler.execute(NewModelElementWizardHandler.java:40)
        at org.eclipse.ui.internal.handlers.HandlerProxy.execute(HandlerProxy.java:290)
```

Figure 2. Error displayed when using EMFStore together with OPENCOSS

3.1.3 Identified Gaps

The main gap of the EMFStore technology is its lack of integration with the Eclipse Project Explorer view (see Figure 3 and Figure 4). This view is the predefined way provided by Eclipse to manage local model files of a given editor. Since the other AMASS tools will use this view, we will get problems to integrate with EMFStore. EMFStore has its own view called Model Explorer (showed in Figure 3) to create, modify, update and delete models. This EMFStore view is also used to share and synchronize local models with the EMFStore server repository, in order to make them available to other EMFStore clients. All these functionalities are accessible through context menus managed by the Model Explorer.

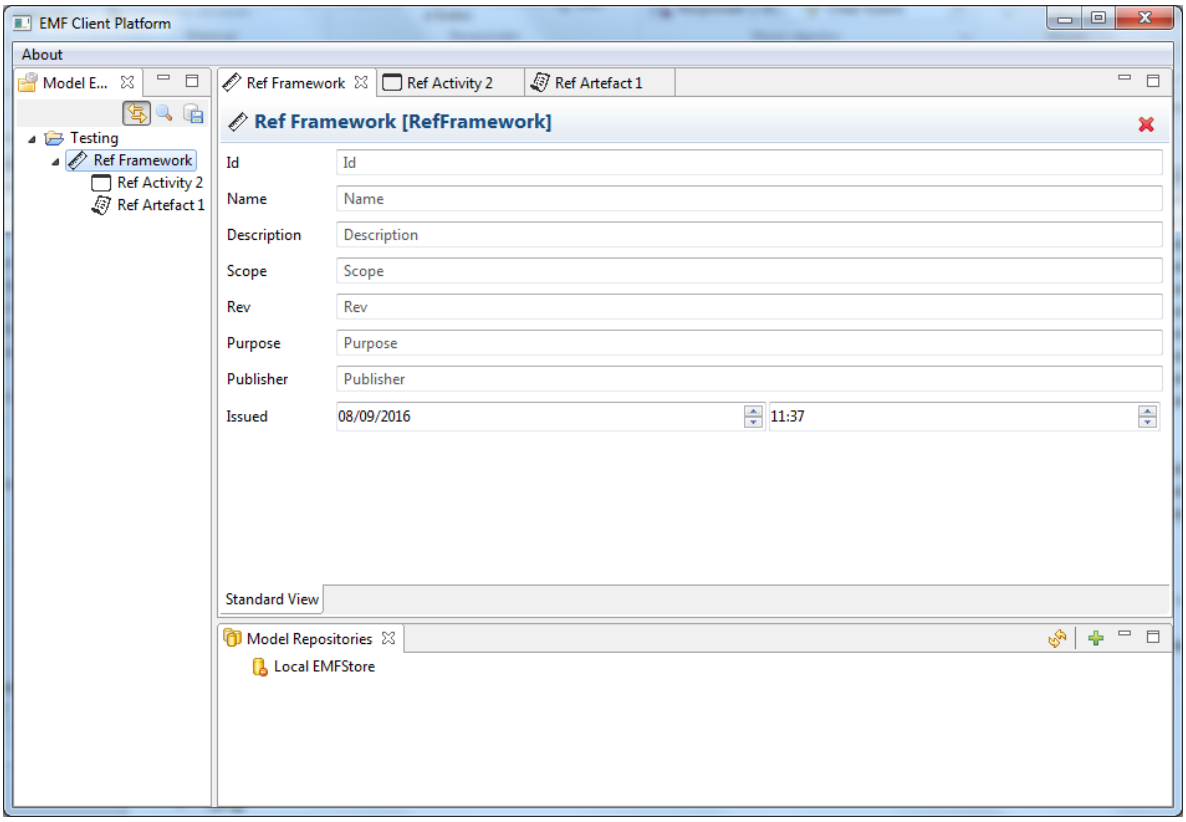

Figure 3. EMFStore environment editing and OPENCOSS model

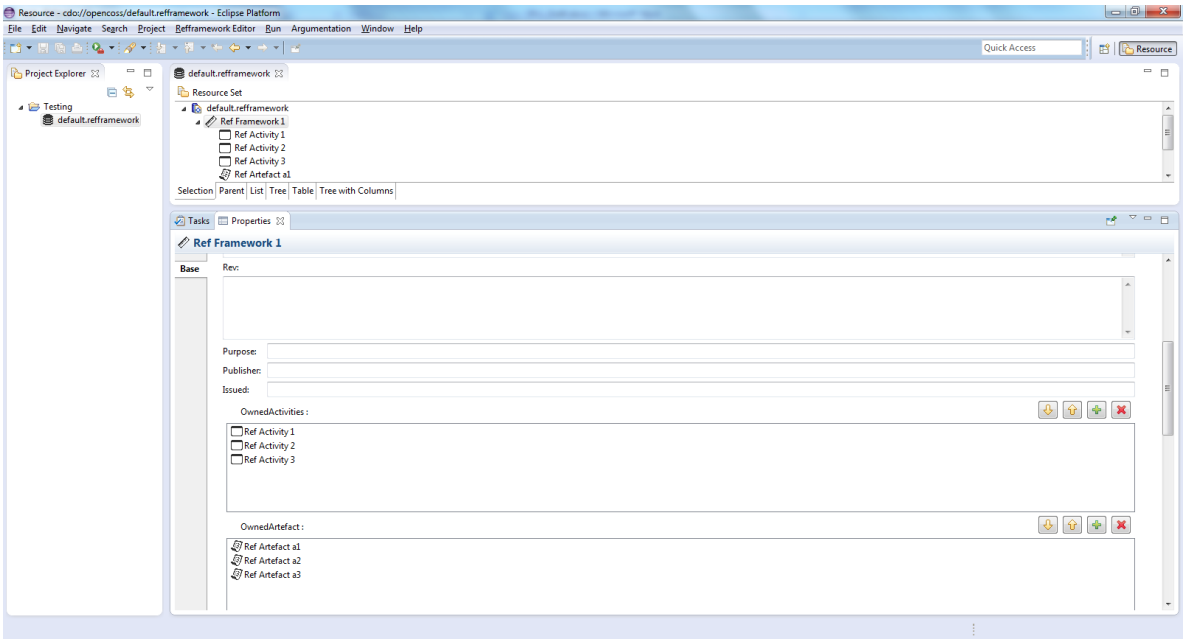

Figure 4. OPENCOSS environment editing OPENCOSS model

There is no support in EMFStore forums to solve this issue[1]. The solution to this problem would be to copy models from one view into another, but we see this solution impractical.

Another gap is that EMFStore generates dynamically at runtime the editors for the models using EMF Forms. This is a problem because we could need customizations for the generated editors, which must be done at design time. With EMFStore we will be unable to use EEF technology, which improves the EMF Forms interfaces making them easier to use and useful for big models as we are going to manage in AMASS.

3.2 Database-based solutions

3.2.1 Overview

Connected Data Objects (CDO) [3] is storage technology compatible with the Eclipse Modeling Framework (EMF). It should serve all AMASS functional modules with common, monolithic data storage facilities to create a solid base for further development, for example creating editors and viewers not based in eclipse technologies. This technology provides some interesting features that could facilitate the development of the AMASS modules:

- **Persistence** of the models in all kinds of database backends like major relational databases or NoSQL databases.
- **Multi user access** to the models. Model repositories can be configured to require secure authentication of users and various authorization policies can be established programmatically.
- **Transparent temporality** is available through audit views, a way to get any model in the state it has been at a point in the past.
- **Scalability**, the ability to store and access models of arbitrary size. CDO supports **lazy loading** which may increase the responsiveness.
- **Thread safety** ensures that multiple threads of your application can access and modify the models without worrying about the synchronization details.
- **Collaboration** on models with CDO is a snap because an application can be notified about remote changes to the models made by other user.

[1] https://www.eclipse.org/forums/index.php/t/728174/

- **Clean API** to work with sessions, views, transactions and objects.
- **Offline work** with models is supported by two different mechanisms:
  - Create a **clone** of a complete remote model repository, including all history of all branches.
  - Check out a single version of the object graph from a particular branch point of the repository into a local CDO **workspace.**
- **Security Manager** allows configure the CDO repository's users, groups, roles and the access permissions to models.
- It is possible to run CDO Server in **standalone/non-Eclipse environment**. For example, in a Tomcat Server, that could be used also to host Web based Model Editors or OSLC provider (a set of web services which follows the OSLC specification to expose our models data to other tools).

3.2.2 Experiments

During the OPENCOSS project, we adopted this technology [4] and it worked decently well except for the problems identified in this Gaps section. As some partners already have experience with this technology, this will reduce development effort in AMASS and it will be useful to help partners with less experience to adopt it.

Besides, we used Eclipse Dawn framework [5] that helped a lot in creating models graphical and form based editors to work with CDO.

3.2.3 Identified Gaps

1. Access Performance problems, which must be studied more carefully.
2. Model evolution is a problem. It would be good to have a similar solution like EMFStore, which automatically update the models if the meta-models change. Some information could be found in the Eclipse forums to solve this, but it seems to be no trivial.
3. Data integrity problems introduced after deletion operations (references to deleted objects are not deleted automatically). We implemented the delete operation to avoid data integrity problems.

3.3 Cloud Storage-based solutions

3.3.1 Overview

In cloud computing, applications can be distributed over multiple servers and are available to different types of clients (desktop PCs, mobile phones). Different parts of a web application communicate using web protocols, resulting in a network of connected web services. Much of the data in such applications is stored in relational databases. That approach is not suitable for document-based data, which is usually stored in file systems. Documents have traditionally been published online using FTP servers, but he FTP protocol does not integrate well with web services who like to authenticate with OAuth and communicate with protocols based on XML and REST.

Cloud storage provides online storage with web-technology friendly interfaces. It is accompanied by web services, which offer authentication and browsing of files. There is usually a web interface for web browsers as well as an API for integration. Most cloud storage providers store rich metadata along with the files including the date of creation, file type, access restrictions and a history of activities for the file.

Using cloud storage over other types has three major benefits. Firstly, cloud storage is easily accessible using browsers, from web applications and even as remote drives in desktop computers if the user has the appropriate client software involved. Secondly, cloud storage platforms do not have to be maintained by the users. There is no need to install security updates or make backups. Thirdly, the scalability is very good for commercial use. Data is mirrored among different servers of the cloud storage provider resulting in high uptime and good server performance. Many clients can download data from the same account and are

likely to end up on different mirrors. More advantages and disadvantages of cloud storage are discussed in [6].

3.3.2 Experiments

There are several popular cloud storage providers for consumers and business customers. While the basic functionality remains the same, providers offer good integration with their other products and services. That would make Microsoft OneDrive the go-to choice for Microsoft Office 365 subscribers. Amazon Cloud Drive is best suited for consumers with an Amazon Prime subscription or enterprises who rely on Amazon Web Services for cloud computing.

For our experiments, we picked Dropbox and Google Drive for being the most popular and well-known providers. Dropbox has been highly successful in popularizing cloud storage and many users are already familiar with it. Google Drive is accessible with a Google account, which many users already have because of Android or another Google service like Gmail. That being said, there is no particular feature of either service that influenced this choice, it was based on popularity and familiarity of potential users with cloud storage services.

Our first goal at this stage was to authenticate to the storage providers. Both use the normal OAuth 2.0 protocol, which has been a hurdle in the past. A Java implementation called Lyo exists, but it takes effort to authenticate with tools like IBM Doors using that implementation. To make integration of cloud storage easier, both Dropbox and Google Drive offer their own APIs in Java and provide working examples. The last remaining step is to register the name of the demo application to the storage provider and get a JSON file with a secret and a key.

While executing the demo, the authentication process is started needs to open an HTML page to log into Google and Dropbox. We showed this page in the web browser for Google Drive and inside an Eclipse RCP demo application for Dropbox. On that page, we can enter the account data and log into the service provider. It is not clear as of yet which type of client might use cloud storage. The OAuth authentication process using HTML pages is suitable for clients with graphical user interfaces but not so much for server-to-server integration or command line tools. Silent OAuth integration is possible by sending HTTP requests with API calls and parsing the resulting HTML page with the access token. It is just not very desirable because Dropbox may change the parameter names of the HTML login form, etc.

When the authentication is completed, the client is granted an access token. While we have this token and it did not expire, we can access the resources at the storage provider.

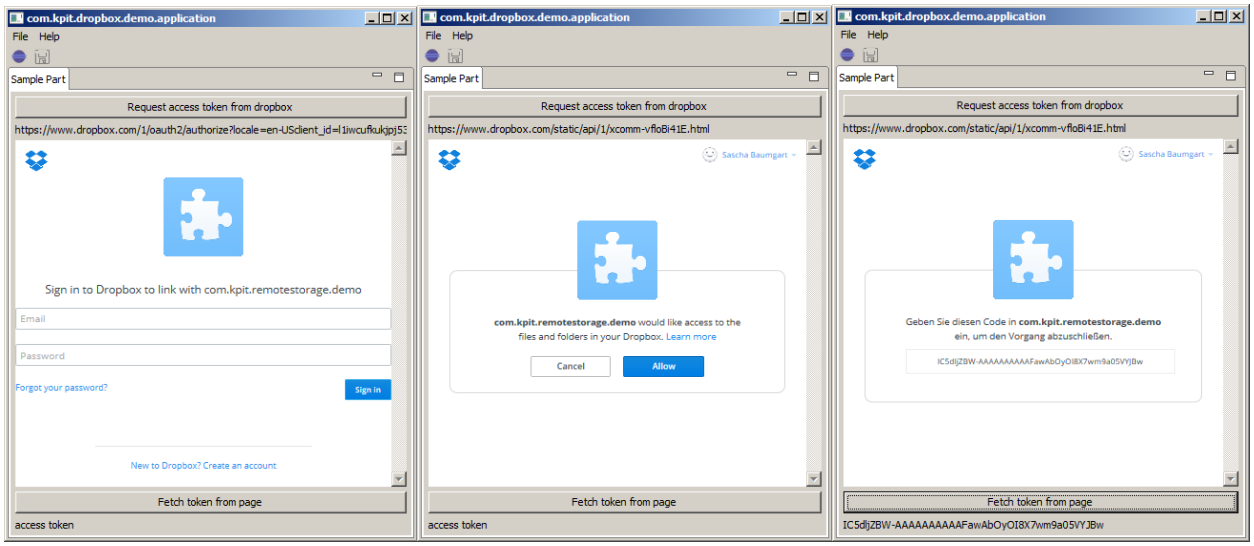

Figure 5. OAuth 2.0 access to Dropbox from within a desktop application

After that, we tested round-tripping of files by (create, upload, download, modify, re-upload) as well as user access restrictions and versioning. Both Dropbox and Google Drive come with a basic form of version control for single files. We stored files as revisions of already existing files with an unlimited number of revisions. Dropbox and Google Drive delete revisions older than 30 days on their free accounts. There is no compare, branch or merge functionality like in Subversion or Git.

We found that performance suffers when data is spread over many very small files. In that scenario the overhead of uploading a file becomes very noticeable. Sequential upload of 1-byte-files took 0.65 seconds per file on average.

3.3.3 Identified Gaps

Cloud storage, relational databases, version control systems and other online storage technologies each have been designed around certain use cases. That is why they each excel in certain areas whereas they are less suitable for some tasks. Cloud storage should only be used when a web application wants to save data into files. It is not a replacement for version control systems in software development and has worse performance than relational databases. Instead, it offers some integration and scalability benefits.

Table 1. Comparison of Online Storage Technologies

	Cloud Storage	Relational Database	Version Control System
Store files	Yes	As BLOB	yes
Store relational data	as CSV or XML	yes	as CSV or XML
Store objects	as XML or custom file format	requires ORM	as XML or custom file format
Multiple Revisions	yes (branch/merge)	no (secondary set of tables or CDO, no branch/merge)	yes
Index data	Some indexing by storage provider	yes (for keys or custom columns)	no
Query data	only in known file types	yes (not inside BLOBS)	no
Web application integration	Yes	yes	no
Data size	virtually unlimited	should fit into server memory	good for huge amounts of source code, takes lots of HD space for binary revisions

There are some implications of having another company host confidential project data. Confidential information is stored online by big companies with lots of employees. The attack area for hackers is increased and the service provider may be forced by law to give data access to security agencies. This is can be remedied by client side encryption of data which adds another layer of complexity and reduces performance. If the online storage is hosted in a different country, there may be international regulations (e.g. trade regulations) which the customer must comply to.

3.4 OSLC-based solutions

This section focuses on OSLC-based solution for seamless interoperability. OSLC is not a storage solution, instead it defines interfaces to reference, read and modify artefacts in ALM tools. First, an overview is given. Then gaps are defined based on experiments executed.

3.4.1 Overview

In the context of software and system artefacts reuse, the Open Services for Lifecycle Collaboration (OSLC) initiative [7] is a joint effort between academia and industry to boost data sharing and interoperability among applications by applying the Linked Data principles [8] [9]:

1) Use URIs as names for things.
2) Use HTTP URIs so that people can look up those names.
3) When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL) and
4) Include links to other URIs, so that they can discover more things".

Led by the OASIS OSLC working group, OSLC is based on a set of specifications that take advantage of web-based standards such as the Resource Description Framework (RDF) [10] and the Hypertext Transfer Protocol (HTTP) to share data under a common data model (RDF) and protocol (HTTP). Every OSLC specification defines a shape for a particular type of resource. For instance, requirements, changes, test cases, models (the OSLC-MBSE specification Model-Based Systems Engineering by the Object Management Group) or estimation and measurement metrics, to name a few, have already a defined shape (also called OSLC Resource Shape).

Thus, tools for supporting Application Life-cycle Management (ALM) or Product Life-cycle Management (PLM) have now an agreement on what data must be shared, and how. In the knowledge management area, the Assets Management and the Tracked Resource Set are the most convenient specifications for the purpose of managing artefacts. However, there are many artefacts generated during the development lifecycle which may not fit to existing shapes or standard vocabularies. Simulation models, business rules or physical circuits are examples of potential artefacts whose OSLC resource shape is not defined yet.

Therefore, one of the current trends in software and systems development lies in boosting interoperability and collaboration through the sharing of existing artefacts under common data models, formats and protocols. In this context, OSLC is becoming a collaborative software ecosystem for software product lines through the definition of data shapes that serve as a contract to get access to information resources through HTTP-based services.

In particular, the Representational State Transfer (REST) software architecture style is used to manage information resources that are publicly represented and exchanged in RDF. Obviously, OSLC represents a big step towards the integration and interoperability between the agents involved in the development lifecycle.

However, RDF has been also demonstrated [11]to contain some restrictions to represent certain knowledge features such as N-ary relationships [12], practical issues dealing with reification and blank nodes [13]. On the other hand, some common services such as indexing, retrieval or quality assessment of any kind of information resource are restricted to the internal storage and the query capabilities offered by each particular tool (usually a SPARQL interface).

3.4.1.1 Background on semantic technologies

Taking into account that software and system artefacts reuse is continuously being explored and new technologies and techniques arise to tackle the problems of storage, representation and retrieval, it seems that semantic approaches can ease these tasks. In this light, the Semantic Web, coined by Tim Berners-Lee in 2001, has experienced during last years a growing commitment from both academia and industrial areas with the objective of elevating the abstraction level of web information resources.

The Resource Description Framework (RDF), based on a graph model, and the Web Ontology Language (OWL), designed to formalize, model and share domain knowledge, are the two main ingredients to reuse information and data in a knowledge-based realm. Thus, data, information and knowledge can be easily represented, shared, exchanged and linked to other knowledge bases through the use of Uniform Resource Identifiers (URIs), more specifically HTTP-URIs. As a practical view of the Semantic Web, the Linked Data

initiative [8] [9] emerges to create a large and distributed database on the Web by reusing existing and standard protocols. In order to reach this major objective, the publication of information and data under a common data model (RDF) with a specific formal query language (SPARQL) provides the required building blocks to turn the Web of Documents into a real database or Web of Data. In this context, a large body of work can be found in different domains such as Geography, Bibliography, e-Government, e-Tourism or e-Health, all of them having common needs, such as: interoperability among tools, different schemes or data models, or cross-cutting services (index and search).
On the other hand, in recent times we have seen the deployment of service oriented computing as a new environment to enable the reuse of software in organizations. In general, a service oriented architecture comprises an infrastructure (e.g. Enterprise Service Bus) in which services (e.g. software as web services) are deployed under a certain set of policies. A composite application is then implemented by means of a coordinated collection of invocations (e.g. Business Process Execution Language). In this context, Enterprise Integration Patterns (EAI) have played a key role to ease the collaboration among services. Furthermore, existing W3C recommendations such as the Web Services Description Language (WSDL) or the Simple Object Access Protocol (SOAP) have improved interoperability through a clear definition of the input/output interface of a service and communication protocol.

In order to improve the capabilities of this type of web services, semantics [14] was applied to ease some tasks such as discovery, selection, composition, orchestration, grounding and automatic invocation of web services. The Web Services Modelling Ontology (WSMO) represented the main effort to define and to implement semantic web services using formal ontologies. OWL-S (Semantic Mark-up for Web Services), SA-WSDL (Semantic Annotations for WSDL) or WSDL-S (Web Service Semantics) were other approaches to annotate web services, by merging ontologies and standardizing data models in the web services realm.

However, these semantics-based efforts did not reach the expected outcome of automatically enabling enterprise services collaboration. Formal ontologies were used to model data and logical restrictions that were validated by formal reasoning methods implemented in semantic web reasoners. Although this approach was theoretically very promising, since it included consistency checking or type inference, the reality proved that the supreme effort to create formal ontologies in different domains, to make them interoperable at a semantic level, and to provide functionalities such as data validation, was not efficient. More specifically, it was demonstrated [15] that, in most of cases, data validation, data lifting and data lowering processes were enough to provide an interoperable environment.

That is why the approach based on the W3C recommendations, WSDL+SOAP, fulfilled most of these requirements with a huge industrial and technological support. However, the lack of agreement on the schemas to be shared (any service provider offered their own schema) and the use of a restricted data model such as XML was still present with the result of preventing a paradigm shift.

In the specific case of software engineering and reuse, the application of semantics-based technologies has also been focused in the creation of OWL ontologies [16] to support requirements elicitation, and to model development processes [17] or Model Driven Architecture [18], or to identify commonality and variability among safety processes [19], to name just a few. These works leverage ontologies to formally design a meta-model and to meet the requirements of knowledge-based development processes.

Taking advantage of the Linked Data principles and Web standards and protocols, the OSLC effort emerges to create a family of web-based specifications for products, services and tools that support all the phases of the software lifecycle. To do so, OSLC defines several specifications based on the following principles:

1) Build on the WWW;
2) Keep things simple;
3) Accommodate difference schemes and protocols;
4) Accommodate different representations under a common data model (RDF) with different serialization formats (RDF/XML, JSON, etc.);
5) Align with the W3C Linked Data initiative.

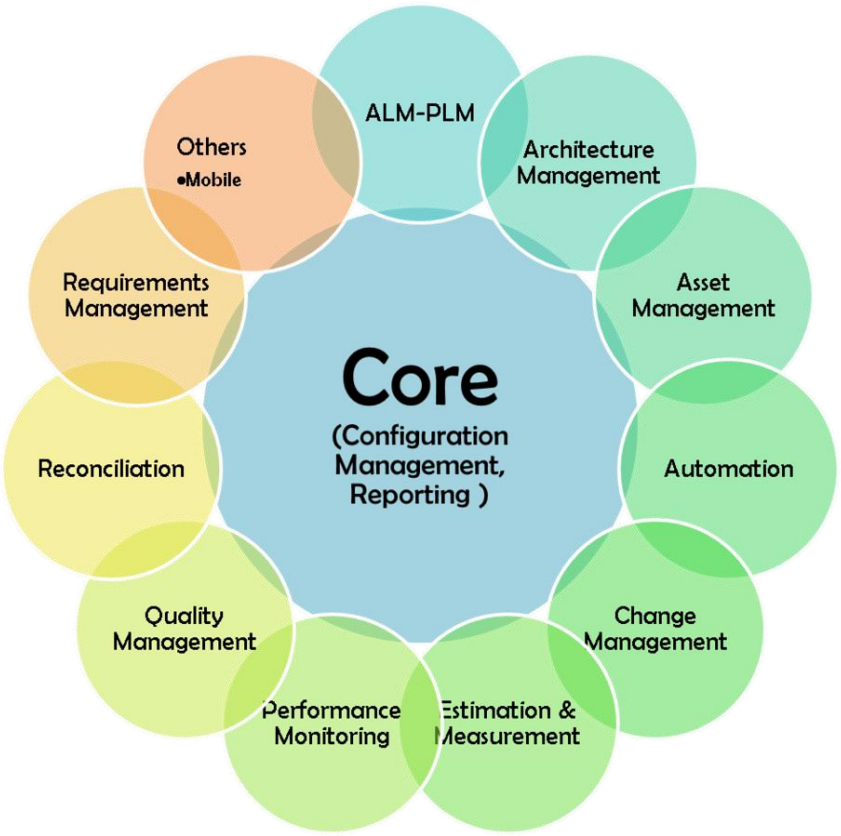

Figure 6. Engineering domains

Similar to OSLC, Agosense Symphony offers an integration platform for application and product lifecycle management, covering all stages and processes in a development lifecycle. It represents a service-based solution with a huge implantation in the industry due to the possibility of connecting existing tools. WSO2 is another middleware platform for service-oriented computing based on standards for business process modelling and management. However, it does not offer standard input/output interfaces based on lightweight data models and software architectures such as RDF and REST. Other industry platforms such as PTC Integrity [20], Siemens Team Center [21], IBM Jazz Platform [22] or HP PLM [23] are now offering OSLC interfaces for different types of artefacts.

In conclusion, it is clear that software and system artefacts reuse is an active research area that evolves according to the current trends in development lifecycles. It may have the potential of leveraging new technologies such as the web environment, service-oriented computing, semantics and Linked Data. That is why current software reuse efforts are focused on providing reuse via software as a service while interoperability is being reached through the agreement on flexible data schemes. Both data schemes and data are being shared using a Linked Data approach (REST services + RDF) with the aim of exchanging any piece of information in a standard environment.

Table 2 summarizes main OSLC Specifications, their status for each specification release version and their current usage by AMASS Partners. C means OSLC Client, S means OSLC Server.

Table 2. OSLC Specification Status and AMASS Partner Usage

Specification	Status				AMASS Partners Usage	
Version	1.0	2.0	2.1	3.0	Fill other Partners…	HON
Core		Final		Draft		C&S
Architecture Management		Final		Draft		
Asset Management		Final				
Automation		Final	Draft			C&S
Change Management		Final		Draft		
Configuration Management	Draft					C
Estimation and Measurement		Draft				
Performance Monitoring		Final				C&S
Quality Management		Final				C&S
Reconciliation		Final				
Requirements Management		Final				C
…						

3.4.2 Experiments

MDH in cooperation with Scania AB, in the context of Gen&ReuseSafetyCases-SSF, is conducting experiments aimed at exploring OSLC as well as semantic web-related knowledge representation means. In particular, via a couple of master theses and (related) scientific publications (see [24] and [25]), an investigation of OSLC-domains and their possible extensions has been pioneered. The extension focuses on the provision of ISO 26262-specific vocabulary in order to enable the exchange of information needed for the creation of arguments fragments aimed at arguing about assurance, and ultimately creating semi-automatically assurance cases. More specifically, two domains have been under exploration: The Quality Management (QM) domain, in order to incorporate resources related to ISO 26262, Part 6, clauses 9-11 and the Architecture Management (AM) domain in order to incorporate resources related to ISO 26262, Part 6, clauses 8.4.2-8.4.4. The choice of these two domains is motivated by the intention of considering a horizontal slice of the V-model during the software development and to explore OSLC-enabled traceability related to immediate and direct evidence [26]. Currently, the work is expected to be further developed by creating OSLC adaptors for the implemented software architecture and for the testing tool. The ultimate aim is twofold: 1) to achieve a proof of concepts showing traceability between the populated instances related to the left and right-hand side of the software V-model; 2) exploit this traceability and argue about satisfaction of ISO 26262 requirements and compliance.

Honeywell plans to experiment with service discovery and notifications for OSLC Automation and OSLC Performance Monitoring Specifications. The goal is not only to offer seamless integration of the tools but also user experience and increase availability and responsibility of all the services. Moreover, comparison with competing integration technologies will be performed to select the best technology for our needs.

In the context of the Crystal project, FBK has developed an OSLC adapter prototype for its tool nuXmv. The current OSLC Domain specifications does not fit the application domain of nuXmv that is validation and verification. Therefore, based on the Lyo framework [27], FBK has chosen to implement the adapter

compliant with the OSLC Automation 2.0 specifications. In addition, the adapter provides a preliminary web interface based on the OSLC Delegated User Interface specification.

3.4.3 Identified Gaps

1. Too much information is transferred – consumes bandwidth.
2. Not designed for heavy event-triggered traffic.
3. Servers are assumed to be always online.
4. There is no service discovery.
5. It does not support control integration.
6. When integration uses OAuth, is it very difficult to detect errors.
7. No free Configuration Management Specification implementation.
8. Linking data requires the users to become familiar with using different applications.
9. No standard way to synchronize documents.
10. Each link is defined unidirectional only.
11. Automation of linking of items that meet certain criteria is not possible.
12. Since Eclipse Reference Implementation for OSLC (Lyo) is not too active at this moment, there is a risk of having limited support for any issue with the implementation.
13. Data link aging.

3.5 Web based Model Editors

3.5.1 Overview

Web editors are viable alternatives to rich clients. They usually manipulate data stored on a central server. Many users and multiple different web applications can work on the same set of data, potentially offering seamless integration real-time team collaboration using web technologies and internet browsers.

Many models can be edited using form editors. Forms are naturally supported by HTML which allow text fields and buttons to submit the data. Advanced features like live validation can be implemented asynchronously. Form data is easily saved in a model and implementers have a wide variety of web control frameworks they can chose from, including pure HTML forms and custom JavaScript.

Text from a text area can also be parsed into a data model. That approach is supported by the Xtext [28] library. A grammar for the domain specific language (DSL) needs to be defined which internally becomes the input for the parser generator ANTLR. In addition to the parser, an EMF metamodel is generated along with a text editor. Users can enter text in the specified DSL into the text editor which has syntax highlighting. Valid text input is directly mapped to the EMF model. The Xtext framework can generate either text editors for eclipse RCP applications or HTML text editors for web applications. DSLs are not a very common way of creating model data, but there are situations when it may be preferable, e.g., when a custom view of the data is generated by defining a query.

Models are more commonly created in a graphical way with graph diagrams (meaning nodes and vertices, not to be confused with charts and plots) where boxes and connection lines represent model elements. This has been successfully done in OPENCOSS [4] and many other projects. Although a tree-based editor could be used to create the very same models, a diagram is the most natural and user-friendly type of editor for models where a graphical notation is defined by a standard. That includes standards as diverse as SysML, GSN and ISO 26262 compliant fault trees. The OPENCOSS prototype put most editors into one rich client, but there is a general trend towards specialized and interconnected tools. It may be beneficial to implement parts of the AMASS platform as standalone editors and have them work on parts of the models defined by CACM. Diagram editors implemented as standalone web applications could be re-used in other contexts and achieve great interoperability through established web standards. The success of such an editor will greatly depend on not just the amount of implemented features but foremost the ease of use.

Let's contrast that with a full-blown eclipse RCP application with many different GMF-based diagram editors. Although eclipse RCP and GMF offer a tremendous amount of features with little initial effort, the usability of the application can be lacking due to the unintuitive and complex workflows. It may be worthwhile to put some of that complexity into lightweight stand-alone web applications working on just a subset of the complete CACM model.

3.5.2 Experiments

As we put a high priority on usability, it is desirable to build the diagram editor on top of a robust library with basic functionality which users expect, e.g. load & safe, copy & paste, undo & redo, grids, zooming, automatic layout, model validation and many more. There are several JavaScript frameworks which come close, but we picked mxGraph [29] by market leader JGraph over its competitors with Go.js by Northwoods Software being a close second place. The two main reasons are the comprehensive documentation and code examples by mxGraph and the menu style which is close to regular office software.

mxGraph is a mature diagram library and used in popular web applications such as draw.io, a free diagram editor app built on top of mxGraph. We modified the example application and incorporated different types of diagram elements. After defining the stencils, we were able to create basic GSN diagrams.

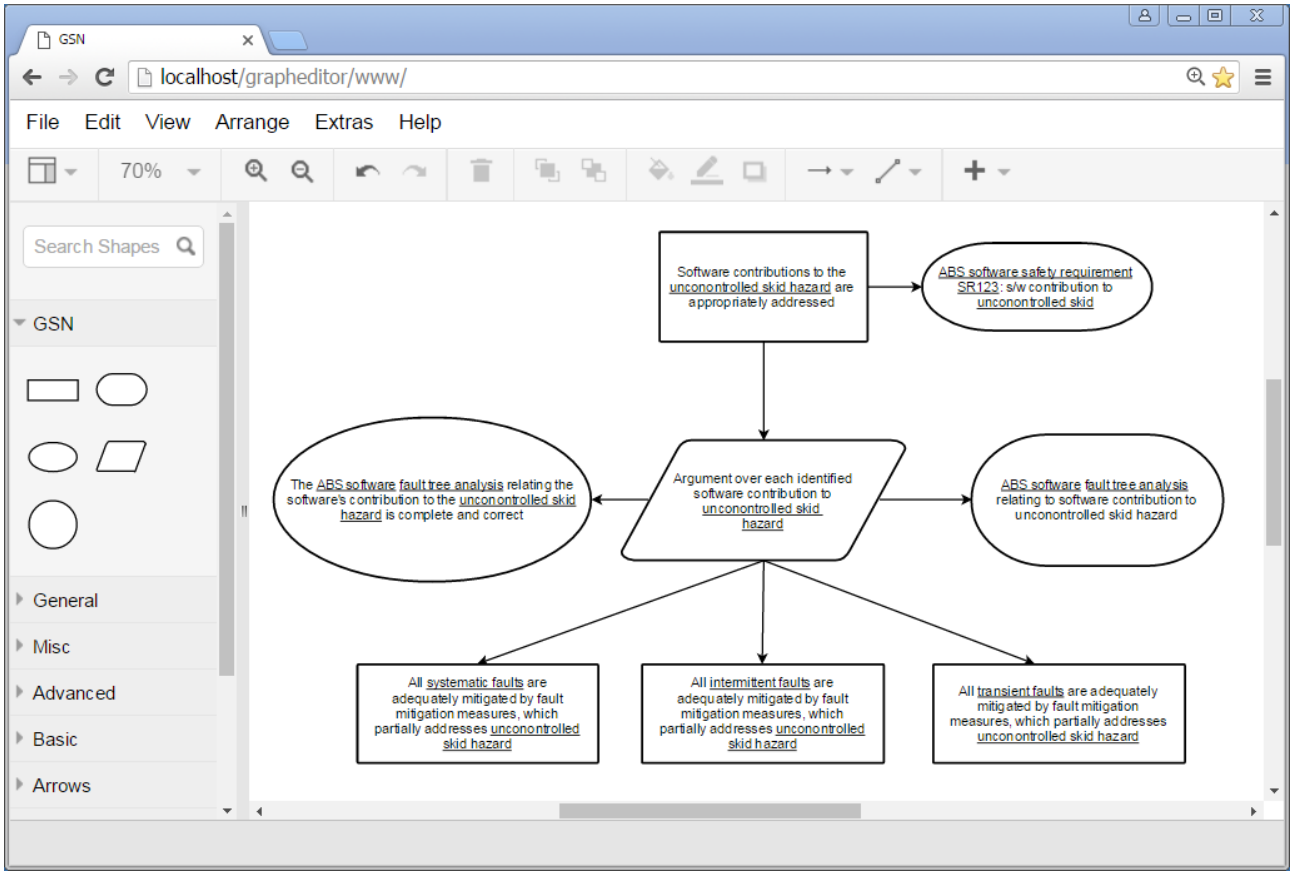

Figure 7. mxGraph diagram with custom GSN palette

3.5.3 Identified Gaps

Our experiments yielded very positive results concerning the interface and the user experience while working on diagrams. It has to be noted, though, that drawing diagrams is not sufficient for most model editing purposes. In all likelihood, the editor will need a section where properties of highlighted elements can be changed. These changes need to be stored in a secondary data model alongside the pure diagram

nodes and vertices. Data from both parts of the model should be coded into CACM-compatible XML during save operations. The mxGraph library supports loading and saving from other formats by means of custom codecs (coder-decoder) which do the model transformations. Additionally, the editor needs some sense of the larger certification project when one model spans several diagrams.

3.6 Real-time Collaborative Editing-based solutions

3.6.1 Overview

Most software development teams employ some form of software versioning and revision control. They enable team collaboration on a set of text files, which sit in one central source code repository. Developers checkout the code of their project from the repository, modify the code and check-in their changes. The repository keeps track of all changes so that the development team can easily revert any changes. Sometimes developers want to modify the same source file. Small teams in one location can take turns editing. Distributed teams either lock some files or use the branch-merge functionality of the revision control system. The first approach is suitable for BLOBs, the latter for text files. Structured text files such as XML can be difficult to merge on a line-by-line basis because the result needs to be well-formed and conformant to a schema definition.

Engineering has borrowed the tools from software development as part of their change management plan. The latter is prescribed by many engineering standards such as ISO 26262. The approach is lacking when it comes to concurrent editing of large binary files. Some tools like IBM Rational DOORS Next Generation have adapted concurrent editing of modules because it is not feasible to write-lock the whole module. Instead, DOORS locks only a single artefact during the editing operation.

In the Web 2.0 world, Google has supported collaborative editing of the same file in Google Docs since 2006. Another real-time collaboration tool named Google Wave was cancelled because of the lack of user acceptance. It merged functionality from mail, wiki, messaging in a real-time environment.

In 2013, Google published the Google Drive Real-Time API, the framework underlying Google Docs real-time collaboration capabilities. Google worked with three software projects to support the real-time API, namely Neutron Drive (source code editor), Gantter (Gantt diagram-based project scheduling) and draw.io (diagram editor). Collaborative diagram editing is particularly interesting as a way of creating models in AMASS. The free diagram editor draw.io is built on top of the JavaScript diagram library mxGraph, which we described in Section 3.5. All data is stored in Google Drive, see Section 3.9. The Google Drive Real-Time API is not designed to work with other cloud storage providers. The web application developer needs to provide a document data model, which inherits predefined base types, and then the API takes care of data synchronization across Google Drive and all opened editors.

The synchronization protocol of the real-time API guarantees that, when all users stop editing the document, the data will eventually be consistent across all browsers [30]. Eventual consistency only ensures that Google Drive and all Clients will end up having the same data. It makes no guarantees as to when all users will see the final data and in which order the editing operations are applied. The API uses a last-write-wins consistency model. This is acceptable because all users can see the data after a brief period.

After an editing operation, the API sends a description of the model change to the server, which updates all clients. This description is called a mutation. The server and each client store a complete list of all mutations since the last save operation. Now when the client receives a mutation from the server, there may have been local changes on the client, which the server does not know about yet. That means that the mutation is out of date. The client solves the conflict by applying a list of transformations to the mutation from the server. Each transformation corresponds to one of the local mutations on the client. The server uses the same transformation mechanism to update a mutation before it propagates it to other clients.

Most of the time, web application developers do not have to consider the synchronization mechanism of the real-time API but there are some pitfalls. Fields of an object should not be modified step by step if that would leave the object with inconsistent data on the server in between steps. Instead, a compound operation should group those modifications. Additionally, a client should not sort a CollaborativeList. If the client needs to present a sorted list to the user, there should be a sorted view on the list only.

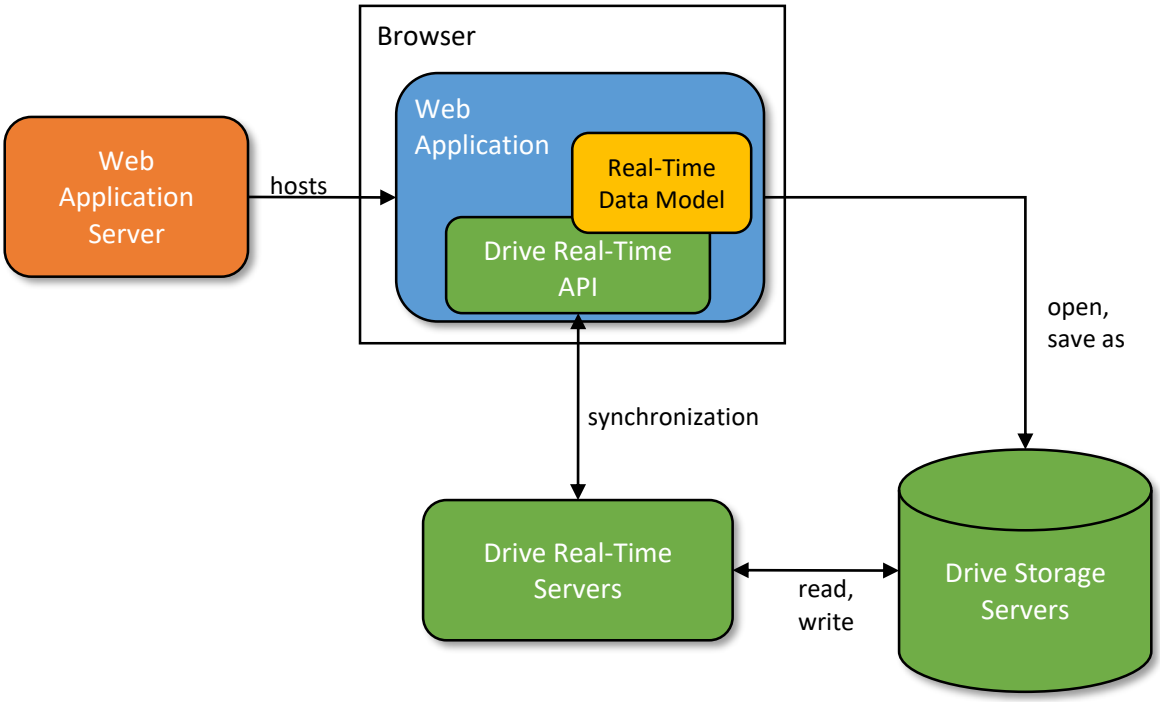

Figure 8. Real-time collaborative editing architecture

3.6.2 Experiments

Google provides a set of demo applications for its real-time API, ranging from the very simple quick start demo to the larger single page playground application based on the Polymer framework. We took the source code and hosted each application on a local web server. Using multiple browsers and Google accounts, we were able to run each demo. Next, we extended our mxGraph demo with real-time capabilities. It takes some effort to integrate the real-time API into the user interface in a seamless and transparent manner. The API provides the required functionality, but still there has to be web controls in the user interface to display the current list of collaborators, highlight the last changes from each collaborator in an unobtrusive way and provide chat functionality. Moreover, it proved to be difficult to debug single page JavaScript web applications where parts of the application are written in a declarative style like with Polymer.

In our experiments, the real-time API is well suited to collaborative document editing for broad documents with little depth. Examples include long text files, requirements lists and even flat graph diagrams (not nested, i.e. no diagram inside the node of another diagram). In these examples, multiple users can edit different parts of the data and see changes from other users in real-time. Furthermore, they are easier to implement given the pitfalls of the API.

The screenshot shows one of the demo applications, a puzzle game build on top of the real-time API. Multiple collaborators can rotate faces of the cube and each editing operation is propagated to all other collaborator. The URL contains the file ID of the Google Drive document that synchronizes all changes.

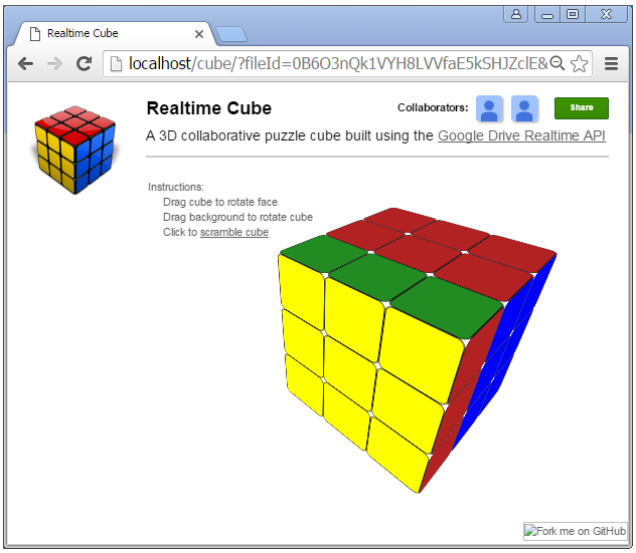

Figure 9. Real-time collaboration in a Rubik cube puzzle app

3.6.3 Identified Gaps

The real-time API is designed to handle single documents. It has no notion of projects consisting of multiple documents, project settings and application settings. The web application has to handle the issue in one of two ways. Firstly, it could create a drive folder with a project file and add more files later. It could even provide different editors for different types of data. That type of implementation would emulate a rich client very similar to the Eclipse Rich Client Platform. Secondly, it could just edit one single document type and be a specialized tool.

The OPENCOSS research project used a relational database as its main integration mechanism. OPENCOSS achieves this using CDO, which handled the object-relational mapping. CDO proved to be a difficult obstacle during the prototyping phase and the object-relational impedance mismatch was very noticeable. For example, the query performance is lacking when objects from a big meta-model with high depths of sub-classes are spread out over many tables. There can be situations where reading the model requires joining all tables. As CDO handles the ORM part, joins are not optimized in a way to minimize the size of intermediate results. Overall, using a relational database for integration had mixed success regarding the development process, leaving some partners looking for alternatives.

The logical step may be object-oriented databases, NoSQL-style databases or some kind of online file storage. Cloud storage offers tooling benefits with regards to real-time collaboration, which is tied to cloud storage APIs. As of yet, it is unclear how cloud storage can be used as an integration mechanism to support the open-minded nature of the AMASS reference tool architecture (ARTA). Alternatively, it is entirely possible to implement only some basic cloud storage capabilities and use it only as required infrastructure for real-time collaboration APIs.

3.7 Automation and Integration Hubs

3.7.1 Overview

Software development projects use a variety of specialized tools for project management, requirement analysis, implementation, defect tracking, version control and test management. Each tool generates project data, which cannot be accessed by other tools, effectively creating a data silo. To share project data among tools, OSLC has been proposed but tool vendors have been reluctant to support the standard for two reasons. Firstly, it creates a classical chicken-egg-problem. Many tool vendors do not support the

standard because customers do not ask for it. And customers do not ask for it because not many vendors support it, therefore it does not contribute much to their project data integration strategy. Secondly, OSLC is not sufficient for data integration. It provides the basic framework to exchange data but not much else. Many tool vendors offer an OSLC compliant REST API but do not bother to integrate other tools.

Tasktop Technologies tries to solve the problem by implementing customized connectors on top of OSLC for tool vendors. To integrate two tools, Tasktop Sync Studio is used to create a mapping between a producer and consumer of OSLC artefacts, e.g. requirements or defects. This approach is superior to point-to-point integration because no one vendor was able to keep up with the large amount of application life cycle management tools.

Integration can be achieved with different strategies, either linking or synchronizing or a mix of the two. Linking has the advantage that there is only a single source of truth. Synchronizing is more user friendly when developers do not want to keep all the different tools open. In any case, there should be a mechanism to avoid data inconsistencies in the form of old links or conflicting versions of the same data.

IFTTT (If This Then That) and Zapier are automation tools and not specific to software development projects. Both let the user create custom scripts which poll a REST API for a trigger event and then initiate some kind of action, using some different REST API. Both support a large number of web services for triggers and actions, e.g. mail providers, cloud storage and many social media websites. String data for the action is directly taken from data fields of the trigger. While the underlying automation mechanism is very similar, Zapier supports some more business and software development oriented web services.

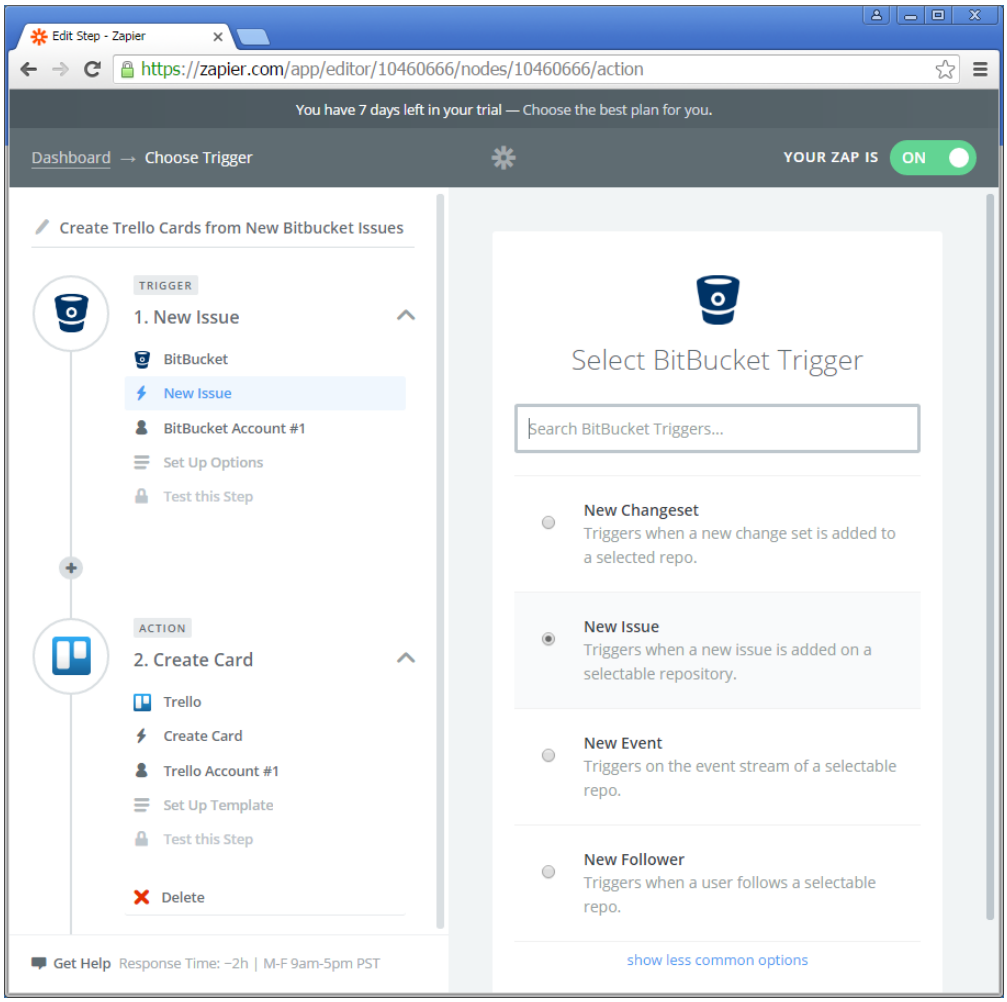

Figure 10. Connecting BitBucket (Version Control) with Trello (Kanban Board)

IFTTT and Zapier allow users to do cross-app integration for hundreds of apps. The only requirement is that IFTTT and Zapier can handle the interface of the web service. While Zapier provides APIs for web app developers, the IFTTT developer platform is still in closed beta.

Tasktop Sync and IFTTT/Zapier are able to integrate tools and synchronize data. They use a similar approach where the service in the middle knows how to handle the end points, thereby avoiding point-to-point integration. Instead, tools need to implement standardized end points for the platform.

3.7.2 Experiments

As Tasktop Sync is not free software, we were limited to evaluating whitepapers, tool demonstrations webinars from Tasktop and their certified connectors. It is possible to achieve good tool integration across various ALM tools by configuring a mapping in Tasktop Sync Studio. Demonstrations showed that the configuration is not overly complex and leads to productivity gains by connecting data from different tools.

IFTTT is a completely free service and Zapier allows testing during an evaluation period. At the end of that period, some automation scripts, which use more complex triggers and actions, are turned off. Looking at the available automation scripts has proved to be very informative. Sometimes not all desirable types of triggers or data fields inside a trigger are available for a web service. Zapier allows custom scripting beyond the normal form based editor for scripts for more complex scenarios.

3.7.3 Identified Gaps

The automation and synchronization hubs define an interface for end points as well as a data exchange format. For Tasktop Sync, usually members of Tasktop Technologies implement the connector to the Tasktop Sync Bus. The connector publishes a normalized façade of the underlying data model so other tools can the data from the Tasktop Sync Bus. The data format is taken from the OSLC specifications, which are tailored to software development and only partially overlap with certification. Extending the OSLC specifications is possible but beyond the scope of AMASS.

IFTTT and Zapier let users do cross-app automation. Data synchronization is possible by duplication, which is triggered when source data is modified. The same mechanism can be used for data updates. At the time of writing, there was a general lack of software development apps that support IFTTT and Zapier. The focus is more on consumers as there are many social media websites supported.

3.8 Model Bus

3.8.1 Overview

ModelBus® is a framework for managing complex development processes and integrating heterogeneous tools. It allows to integrate tools serving different purposes from different vendors. This integration creates a virtual bus-like tool environment, where data can be seamlessly exchanged between tools, avoiding the manual import and export of tool-specific data, which is usually accompanied by manually executed data alignment steps. The ModelBus® framework also allows to link data, thereby establishing traceability of the work products created during the development process. In the latest release, ModelBus extends the linked data support for OSLC standards.

It uses the following transportation protocols: HTTP, HTTPS, XMPP, CXF, JMS, SOAP, OSLC

The orchestration is based on the following standards:

- Business Process Model and Notation (BPMN)
- Business Process Execution Language (BPEL)
- ODE

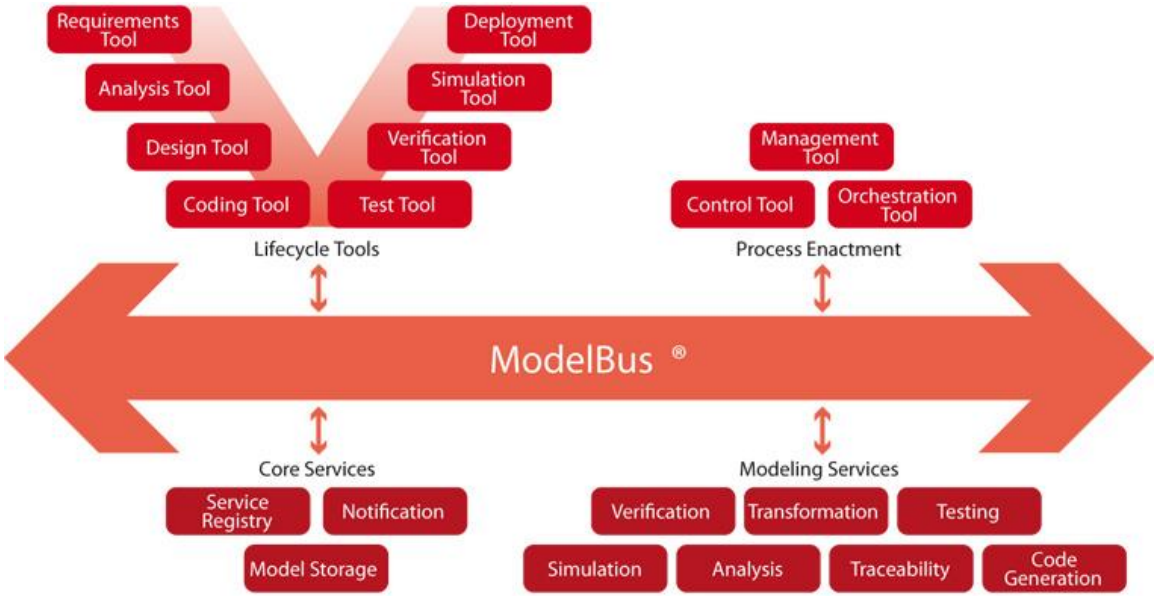

Figure 11. Model-driven data management and service execution

This orchestration engine allows workflows to be fully automated, via interconnected web-services driven from the standardized process specifications. It should be noted that this approach allows for flexible deployment strategies, allowing the ModelBus orchestration to integrate and cooperate with external orchestration technologies.

It is based on the following core technologies:

- DOSGi
- SVN
- Git
- MOF/EMF

3.8.2 Experiments

Evaluation of ModelBus and OSLC suitability of the AMASS needs will be performed by Honeywell. The output will be a comparison with the based on the following proposed criteria:

- Capabilities
- Time to integrate tools
- Coverage
- Scalability and Extensibility
- Risks
- Community/Company support

In the latest release, ModelBus extends the linked data support for OSLC [7] standards (see next section). It was this attribute, which led the Advance Technology team in USA to select ModelBus as proxy integration technology for their velocity product development activities.

3.8.3 Identified Gaps

It is not an open standard as OSLC. ModelBus does not aim to be general integration framework; it is the virtual bus-like service-oriented architecture. There are gaps in Linked Data support.

3.9 Context Aware UIs

3.9.1 Overview

The notion of context is not new and when we want to create applications, devices, and systems that are easy to use, it is essential to understand the context of use. In Human-Computer Interaction, people traditionally aim to understand the user and the context of use and create designs that support the major anticipated use cases and situations of use. At runtime – when the user interacts with the application — the system can decide what the current context of use is and provide a user interface specifically optimized for this context. Context awareness is the ability of a system or system component to gather information about its environment at any given time and adapt behaviours accordingly. Contextual or context-aware computing uses software and hardware to automatically collect and analyse data to guide responses. Context includes any information that is relevant to a given entity, such as a person, a device or an application. As such, contextual information falls into a wide range of categories including time, location, device, identity, user, role, privilege level, activity, task, process and nearby devices/users.

Numerous articles related to context-aware systems have been published [31]. The survey on Engineering context-aware systems and applications [32] proposes an analysis of types of context-aware systems and of the engineering techniques and methodologies used for their development. The review [33] explores the techniques used to model and reason on contexts. The notion of context is closely linked to the notion of situation identification. [34] makes an overview of the techniques used to build high level situational contexts from collected data. Allowing context-aware application to communicate in the web requires the definition of interfaces, protocols, and models using techniques for context-aware web services [35].

The term is mainly applied to the world of mobile and IoT (Internet of Things) applications and linked with the capabilities of sensors. For example, a lot of applications use geolocation data to create context-aware user services; but while location is one of the most excellent example of context awareness, one could just as easily describe many other kinds of information as leading to context awareness. The AMASS tool framework aims at proposing core building blocks addressing multiple development phases and concerns (system component specification, evidence management, compliance management, etc.). The tool platform will also interwork with external tools, e.g. using OSLC. Instead of login to individual blocks and connected tools of the AMASS platform, context awareness will allow a unique authentication to user (single sign-on) to access to all different tools he needs to perform its work.

3.9.2 Experiments

Context awareness can be applied to wearable devices, the internet, cloud services and much more. This is the case with the Google Awareness APIs which using different types of context —e.g., including location, weather, user activity, time— enable app to understand a device's current context and use this information to provide optimized and customized user experiences [36].

When applied in the domain of processes modelling, the context may include the notions of authentication, authorization and accounting. As for an example, the Google Sign-in client library based on an OAuth 2.0 authentication system can be easily used to allow users to connect to your app or web server using their Google account credentials, the same credentials they already use with Gmail and other Google services. For simple access, Google generates an API key that uniquely identifies your application in its transactions with the Google Auth server. For authorized access, you must also tell Google your website's protocol and domain. In return, Google generates a client ID. Your application submits this to the Google Auth server to get an OAuth 2.0 access token.

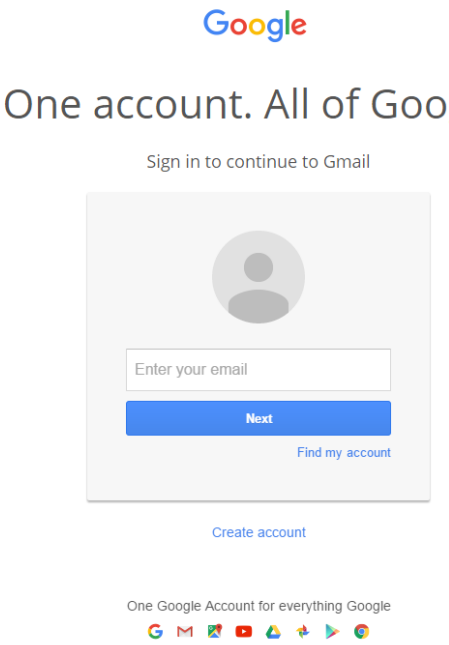

Figure 12. Unique authentication for all Google services

3.9.3 Identified Gaps

There are some shortcoming coming from a context-awareness approach:

- Complexity: Taking into account contexts in a system, adds a higher level of interaction at the cost of an extra-complexity. This complexity may compromise the development and the maintenance of the system. The level of context-awareness introduced must be carefully decided: With more context-awareness, the job of designing specific-context user interface typically becomes more complex as the number of situations and contexts which the system will be used in usually increases. This complexity increase implies an adapted engineering process; the modelling of context data and the capability to exploit them requires specific techniques; the sharing of context in web-distributed applications raises some additional issues.
- Interaction: The aim of context-aware system is to reduce the user intervention, easing its use and decreasing user distraction. According to this purpose, a context-aware system is intended to monitor the context and then act without any human mediation. There is a risk that a system takes away the user control due to a misinterpretation of the context, in situations where the user has a better understanding of what is happening. In these situations, the user can reject the system and stop using it.
- Security: Security issues are a big challenge in running a context-aware system. Context information must always be accurate and authenticated. It is hard to absolutely secure accumulated context data and people rightly feel that context systems violate their privacy because it is like having one's everyday activities exposed for the world to see. Therefore, it is very important to address the security and privacy challenges of running a context-aware system. The user should have a good degree of control over their accounts and repository data. That way, the service provider will not be liable for damages resulting from data leaked from a user's device.

3.10 Other Means related to Seamless Interoperability

The following collection of technologies, tools and approaches are also considered as state-of-the-art related to seamless interoperability, but they have not been investigated in detail. Therefore, the reviews do not follow the structure of the previous sections (overview, experiments, and gaps).

3.10.1 Evidence Metamodels

This section presents evidence metamodels that could be used as basis for the design of the Evidence Management Basic Building Block of the ARTA. We use the following definition of (safety) evidence: artefacts that contribute to developing confidence in the safe operation of a system and to showing the fulfilment of the requirements of one or more safety standards [37].

In addition to the metamodels below, others with a different scope include artefact-related information that could be used for evidence management, e.g. SPEM 2.0 [38] for process modelling. The concept of evidence is also used in some system and components metamodels presented in D3.1.

3.10.1.1 OPENCOSS Evidence Metamodel

The OPENCOSS evidence metamodel [39] is built from the notion of artefact: an individual unit of data managed in an assurance project. In OPENCOSS:

- Safety evidence can be defined as the artefacts that contribute to developing confidence in the safe operation of a system and that are used to show the fulfilment of the criteria of a safety standard (e.g., safety analysis results, testing results, and source code).
- A chain of safety evidence can be defined as a set of pieces of safety evidence that are related (e.g., a requirement, the test cases that validate it and the report where the test results are presented).
- Evidence management is the safety assurance and certification area concerned with the collection and handling of the body of safety evidence of an assurance project, including chains of evidence.

The OPENCOSS evidence metamodel defines the metadata about artefacts which should be captured. The classes and associations in the metamodel can be used to support reasoning about the use of artefacts as evidence of standards compliance or in support of an assurance argument. The evidence metamodel links directly to the OPENCOSS process and argumentation metamodels.

Figure 13, Figure 14 and Figure 15 show the main elements of the OPENCOSS evidence metamodel. The main concepts are:

- Artefact Definition: abstract unit of data to manage in an assurance project; e.g. hazard log.
- Artefact: concrete, individual, and identifiable unit of data; it represents an instance and version of an artefact definition; e.g. the hazard log after design and after V&V.
- Artefact Property: attribute of an artefact, and value.
- Resource: the place, either electronic or not, where an artefact is stored.
- Artefact Relationship: existence of a relationship between two artefacts (i.e., traceability).
- Event: happening in the lifecycle of an artefact.
- Evaluation: specification of the result of making some judgement regarding an artefact.

Regarding the main possible **limitations and extension needs** for the use of the OPENCOSS evidence metamodel in AMASS:

- The metamodel should be extended for better supporting seamless interoperability needs, such as aspects related to tool integration.
- The management of parts of an artefact could be better managed. Nonetheless, the main reason for this limitation is the underlying technological solution adopted in OPENCOSS for implementing the metamodel (EMF).
- As implemented in OPENCOSS, limited support is provided to certain basic evidence management activities such as the creation of traceability matrices and the suggestion of artefact relationships.
- The concepts of the metamodel could be further aligned with existing OMG specifications, e.g. SPEM

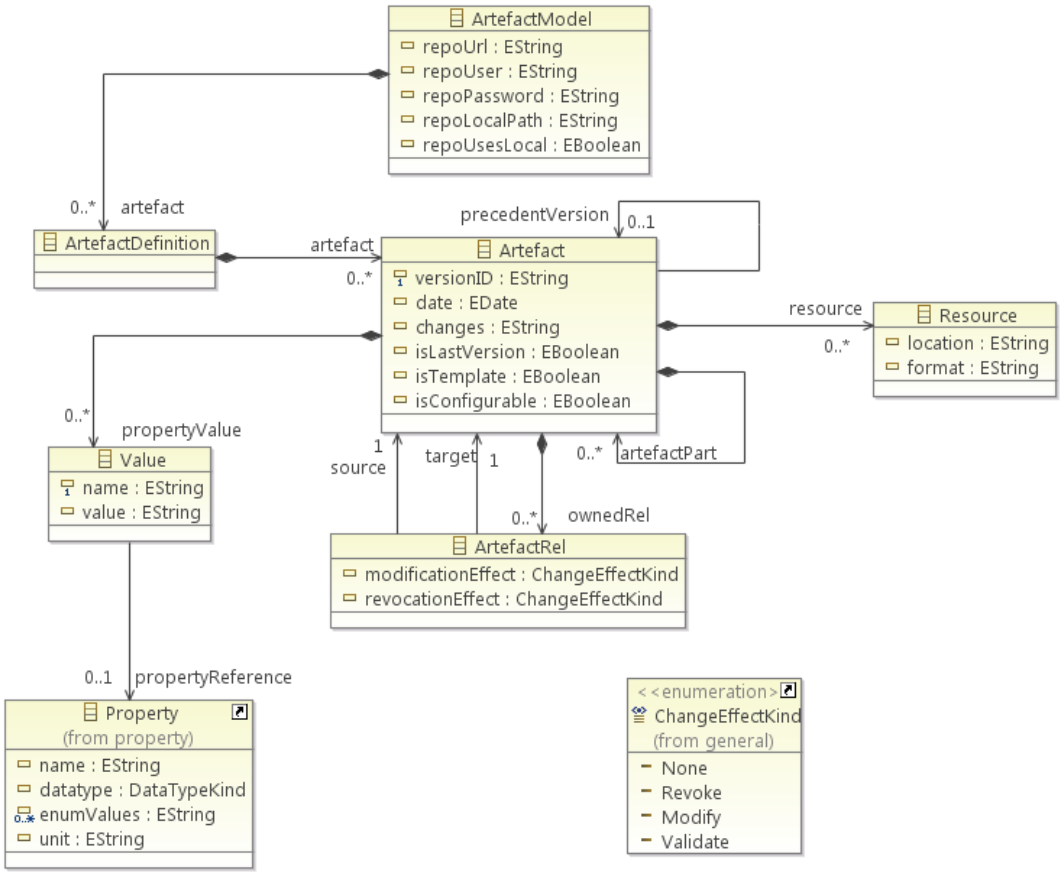

Figure 13. OPENCOSS evidence metamodel: main structure [39]

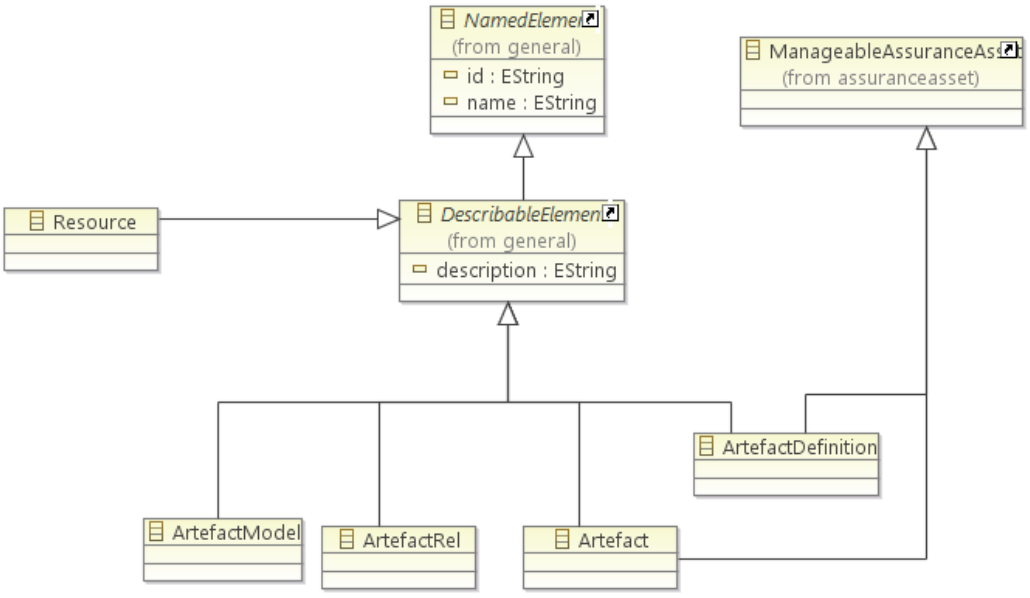

Figure 14. OPENCOSS evidence metamodel: specialization hierarchy [39]

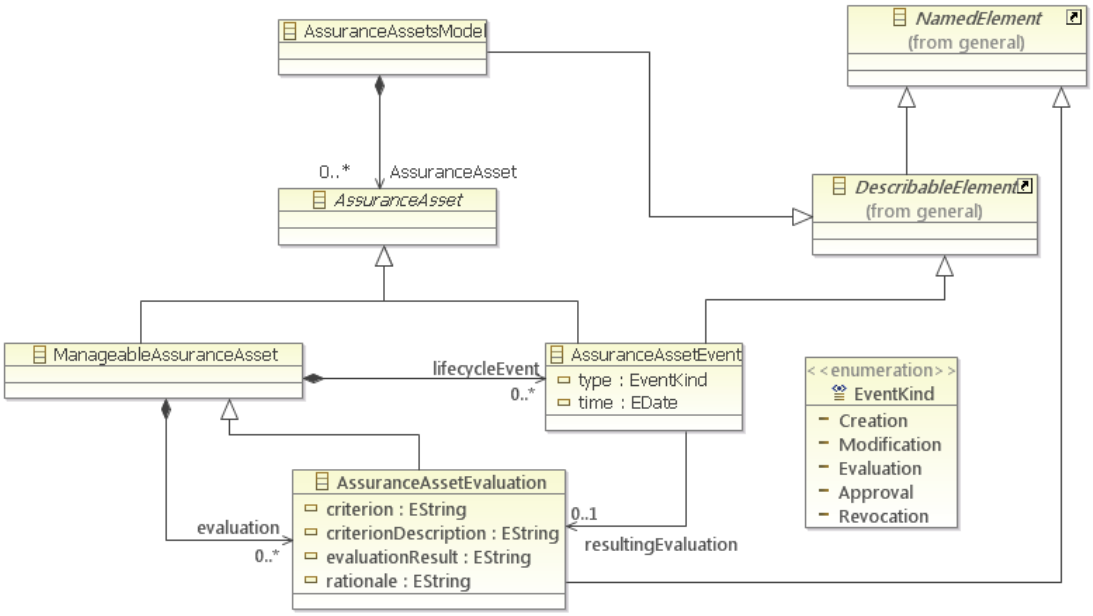

Figure 15. OPENCOSS evidence metamodel: assurance asset information [39]

3.10.1.2 SACM 1.1 Evidence Metamodel

SACM (Structured Assurance Case Metamodel; [40]) is an OMG standard that provides a common framework for assurance case development and information exchange. The work related to SACM started in 2008, initial versions were published in 2010, and the approved 1.1 version has been publicly released in July 2015. The standard is divided into two main parts: the argumentation metamodel and the evidence metamodel. These metamodels were initially created independently and later combined to form SACM because they are complementary. In SACM, an assurance case (Figure 16) consists of a set of argumentations that correspond to its argument and a set of evidence containers that correspond to its evidence.

The evidence metamodel defines a catalogue of elements for constructing and interchanging precise statements involved in evidence-related efforts. This metamodel aims to: (a) identify the main factors that determine the evidence collection process; (b) identify the main factors that determine the evaluation of evidence; (c) identify and define the elements of evidence; and, (d) define a common interchange format to facilitate the exchange of information between different software assurance tools and services. The metamodel consists of 18 different class diagrams and over 100 classes. Two excerpts are shown in Figure 17 and Figure 18. The metamodel has four main logical parts:

- Evidence items (e.g., the software verification results) part, which defines the physical evidence (e.g., the document that reports software verification results).
- Formal elements part, which defines the logical assertions, provided in the form of individual propositions (e.g., the definition of 'software verification results' as a formal object for making assertions about it in an assurance case).
- Evidence assertions part, which defines various statements that can be made about the evidence items (e.g., for indicating that the custodian of the software verification results is a given person).
- Administration part, which can be used to organise individual evidence items and evaluations into a package that becomes a unit of exchange (e.g., for specifying that the software verification results are part of a larger evidence container called 'software artefacts').

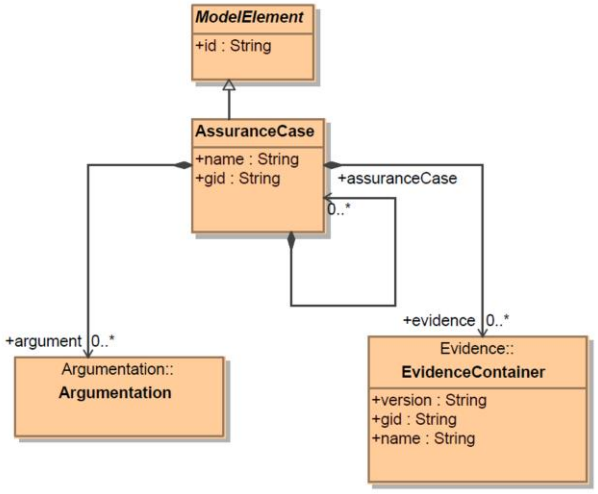

Figure 16. SACM 1.1 evidence metamodel: assurance case structure [40]

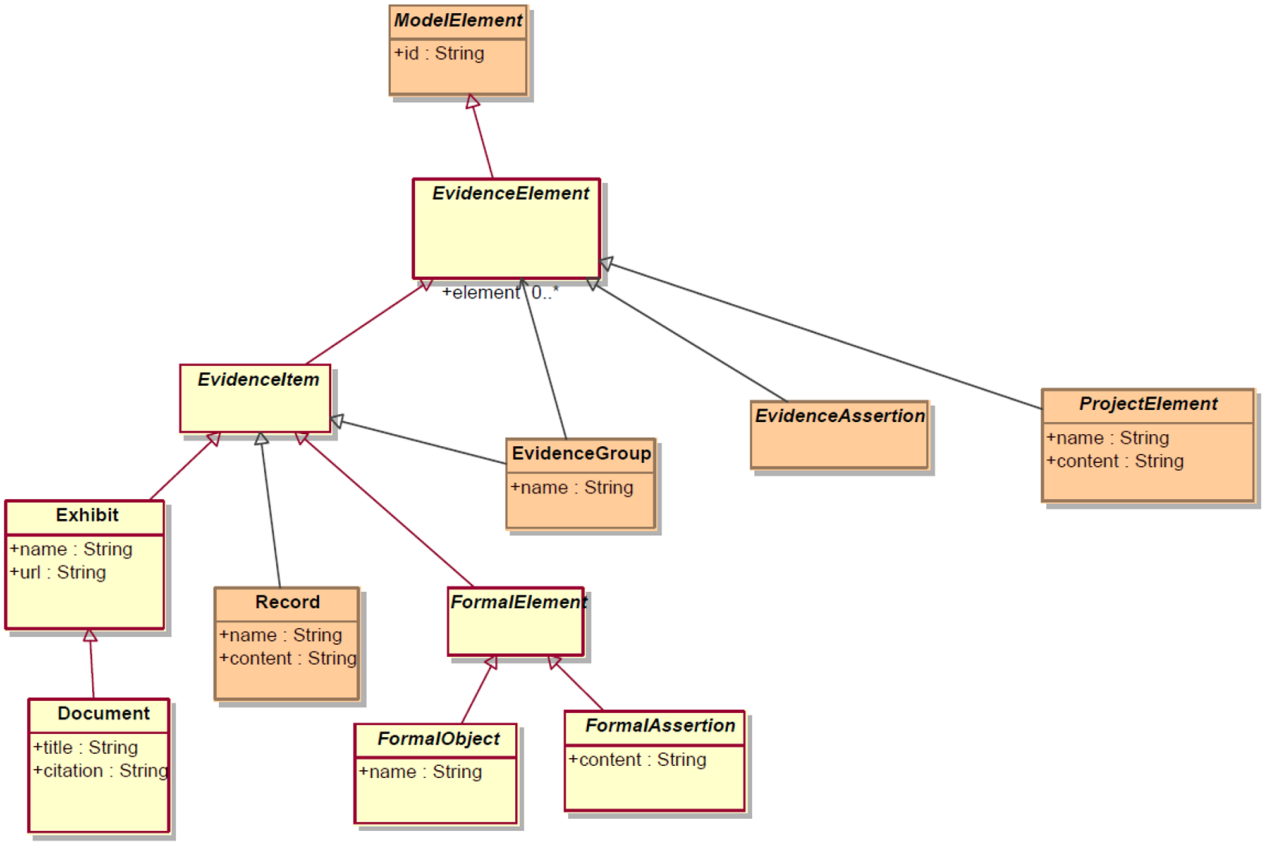

Figure 17. SACM 1.1 evidence metamodel: main evidence elements [40]

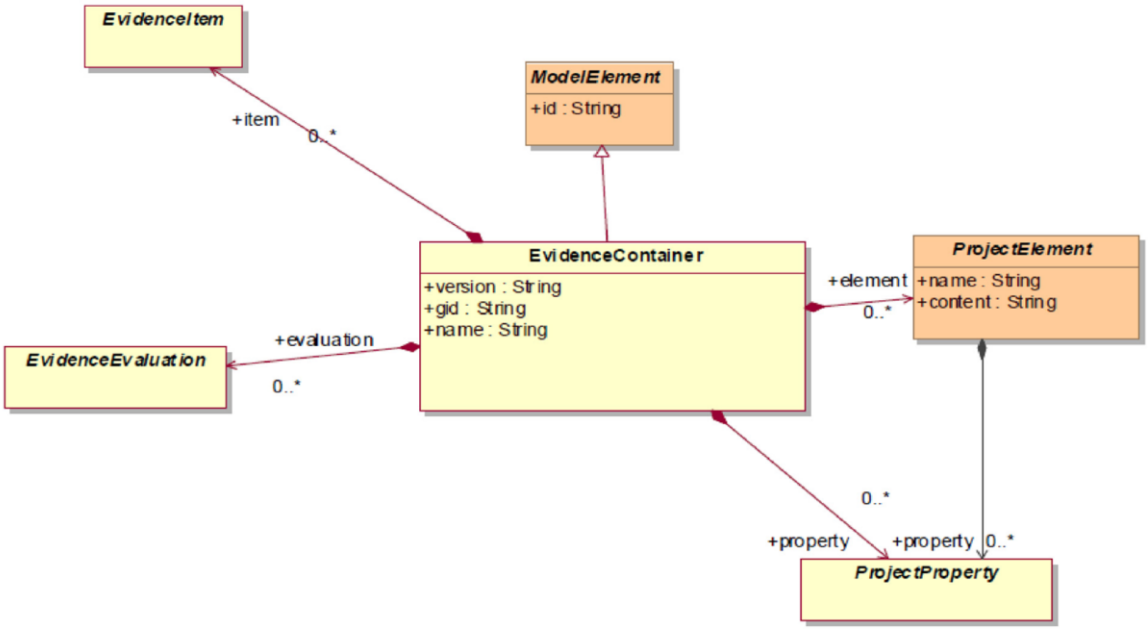

Figure 18. SACM 1.1 evidence metamodel: main project elements [40]

The use of the SACM evidence metamodel was considered initially in OPENCOSS but later discarded due to several weaknesses. Regarding the main possible **limitations and extension needs** for the use of the SACM 1.1 evidence metamodel in AMASS, the metamodel could be improved in the following aspects:

- Clarification of the notion of evidence.
- Clarification of the notion of evidence assertion.
- Reduction in the number of classes and associations, as several overlap.
- Restructuring of classes and associations for better evidence information representation.
- Redundancy reduction.
- Reduction in the scope of some classes, as some are too broad as defined for their intended usage.
- Further justification of the need for the classes and associations, to more clearly understand why and when they are needed.
- Consistent concept definition.
- Extended evidence lifecycle, to support further evidence management activities.

3.10.1.3 SACM 2.0 Evidence Metamodel

A new version of SACM has been prepared recently and it is currently in alpha version (June 2016). The new version addresses most of the limitations of SACM 1.1 and further aims to support a higher number of practices, e.g. the specification of modular assurance cases. For specifically, the main SACM 2.0 goals were:

- Improve support for ISO/IEC 15026-2.
- Improve support for GSN.
- Harmonization of parts.
- Add initial support for Patterns/Templates.

The evidence metamodel (now called artefact metamodel; see Figure 19, Figure 20, and Figure 21) aims to aid in communicating the way in which evidence artefacts are collected and characterising the artefacts. The metamodel identifies the main elements that that determine the evidence collection process: artefacts, participants, resources, activities, and techniques. Artefacts may be exchanged as packages or combined into composites. In conjunction with the argumentation metamodel, certain claims may be expressed to be supported by evidence that is within an artefact model, to permit the authors of the assurance claims to offer evidentiary support for their positions.

Regarding the main possible **limitations and extension needs** for the use of the SACM 2.0 evidence metamodel in AMASS:

- The metamodel focuses on evidence-related information for assurance cases, thus does not provide full support to evidence management activities not closely related to assurance case specification (e.g. change impact analysis).
- As a follow-up indication, the metamodel should be tailored and extended for certain evidence management-related purposes.
- The structure of the metamodel has been aligned with the argumentation metamodel, which has resulted in the provision of general mechanisms for evidence information specification that might need to be restricted or guided purposes.
- A more explicit link with other OMG specifications, e.g. SPEM, could be provided.

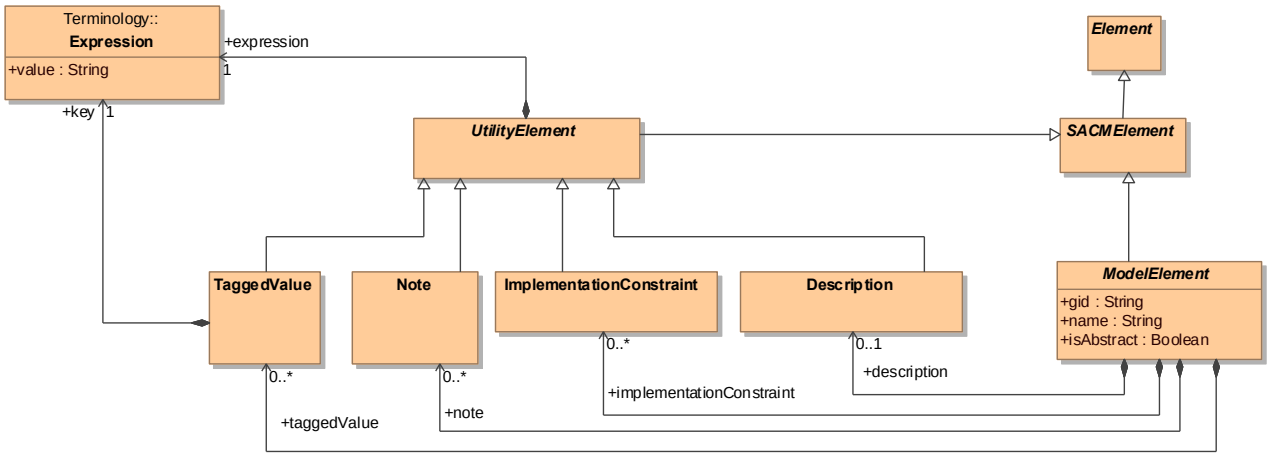

Figure 19. SACM 2.0 artefact metamodel: general characteristics [40]

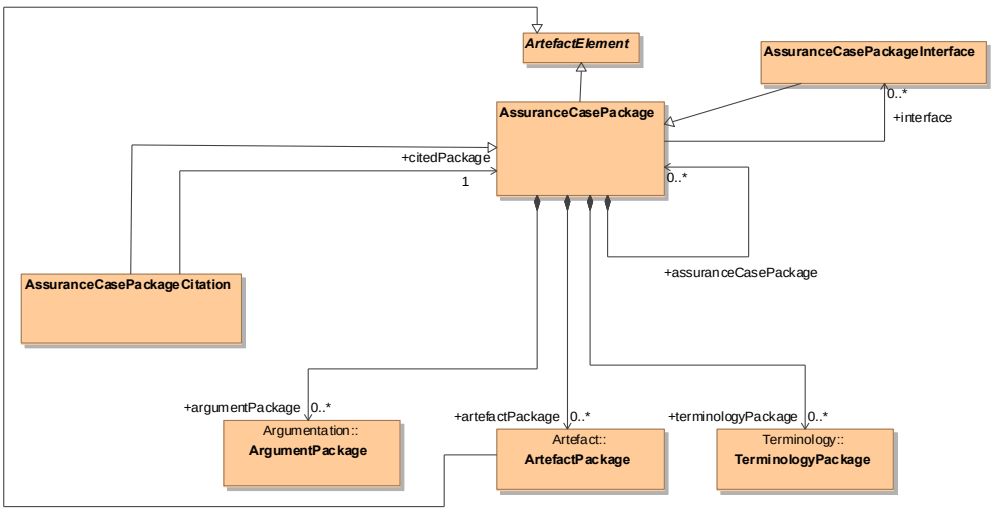

Figure 20. SACM 2.0 artefact metamodel: assurance case structure [40]

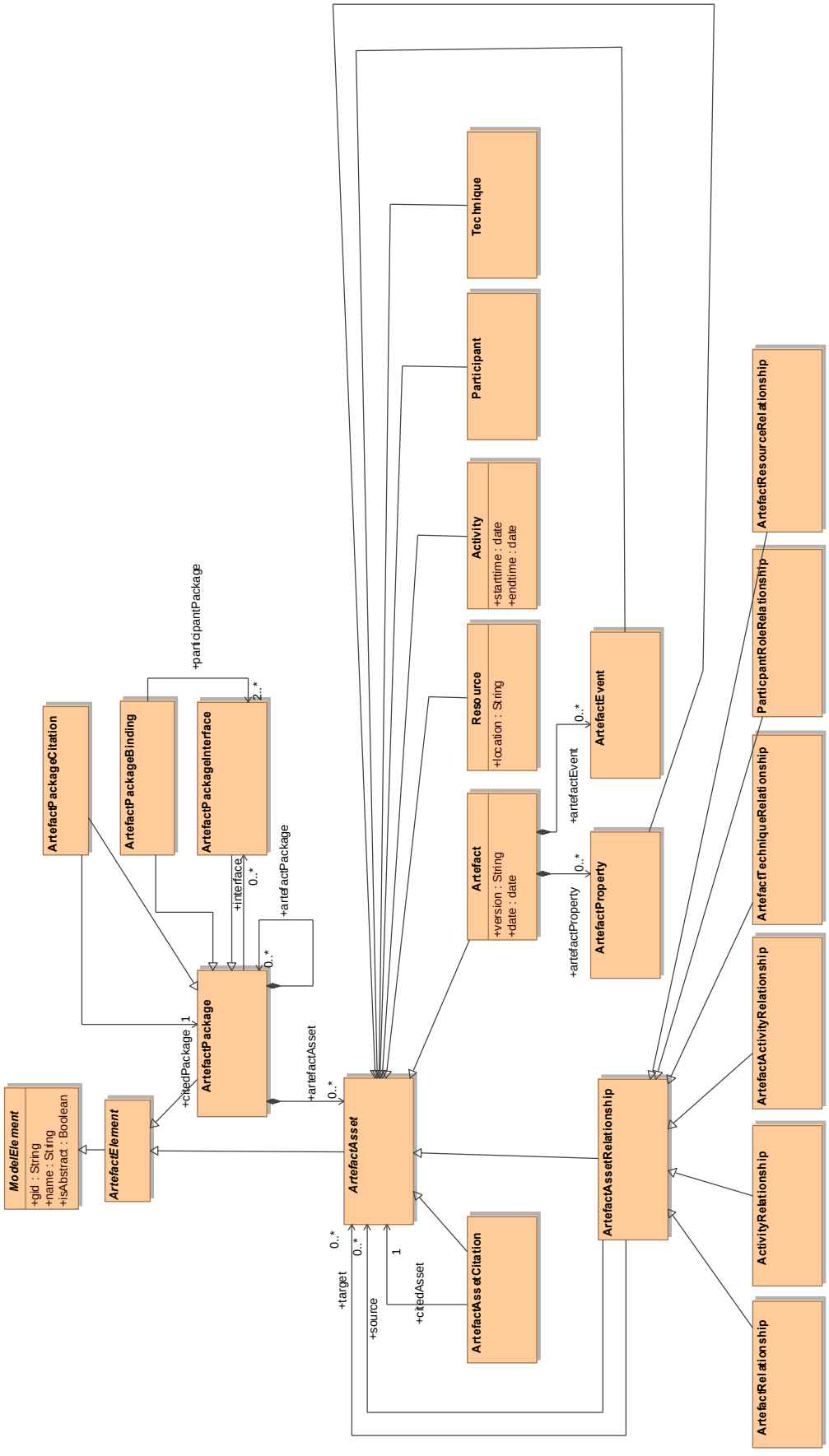

Figure 21. SACM 2.0 artefact metamodel: main artefact elements [40]

3.10.1.4 DAF Evidence-Related Classes

DAF [41] is an OMG specification that aims to provide a new system assurance methodology for dependability argumentation for consumer devices. This is achieved by integrating conventional system assurance approaches such as risk analysis and assessments with a new way of approaching unique characteristics of consumer devices. DAF provides a dependability assurance methodology for safety-sensitive consumer devices, a template for dependability argumentation, and a dependability assurance process. DAF 1.0 has been released in February 2016.

The specification includes the classes:

- Evidence (Figure 22), defined as the basis of the argument for the dependability claim.
- Artifact (Figure 23), which corresponds to a work product that is produced and/or referred by activities. An artifact is an activity-specific occurrence of input/output materials, thus it needs to be related to a corresponding activity. An artifact can be referred to by multiple activities, and can be evidence of dependability processes.

A positive aspect of DAF is its explicit link with other OMG specifications, e.g. SACM and SPEM.

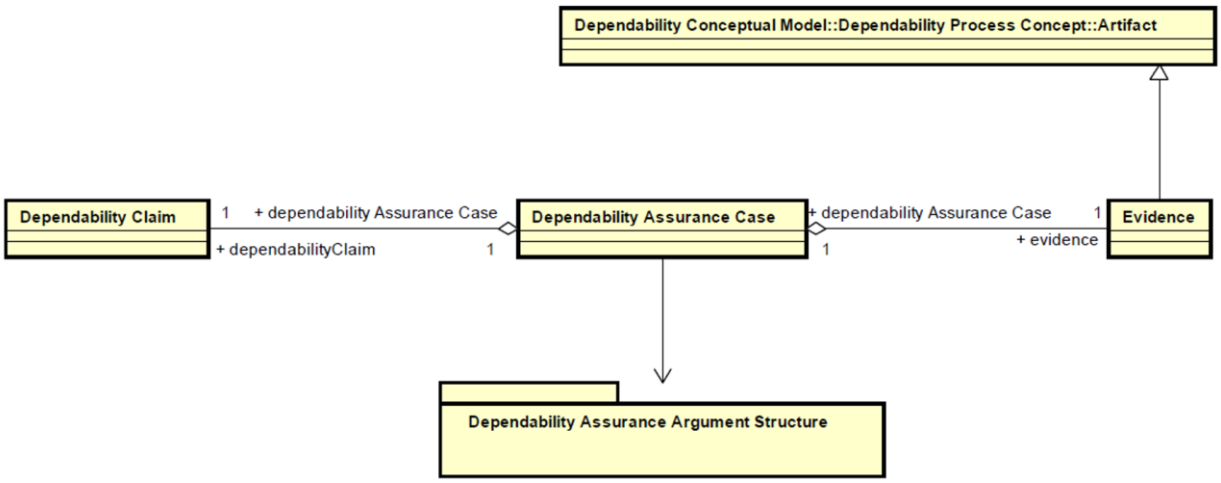

Figure 22. DAF evidence information [41]

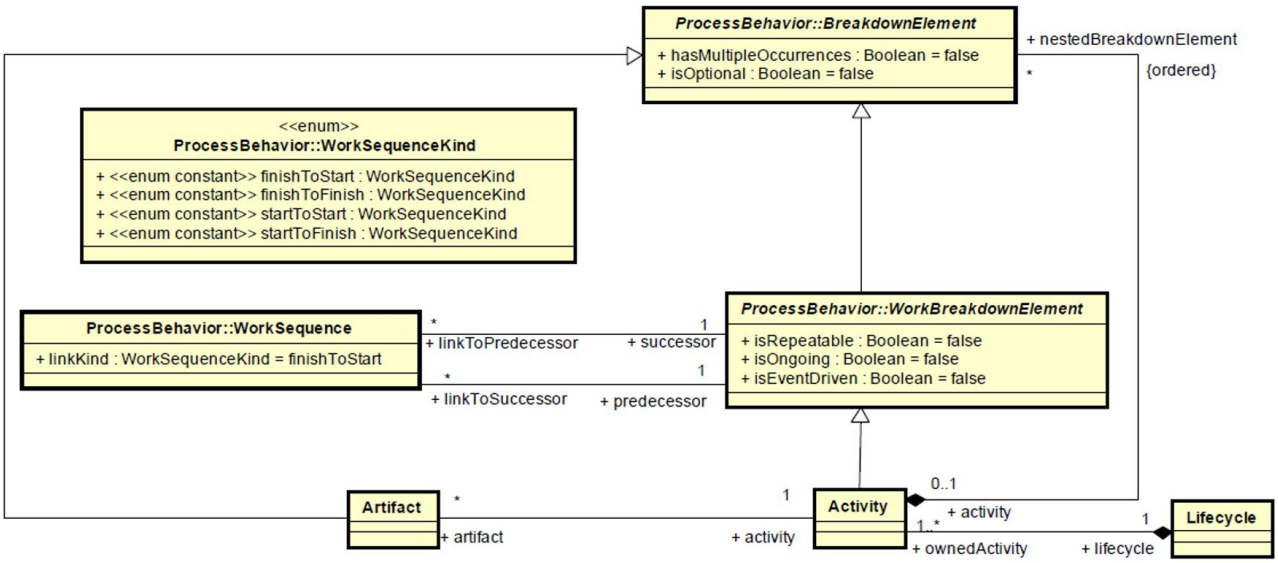

Figure 23. DAF artefact information [41]

Regarding the main possible **limitations and extension needs** for the use of the DAF evidence concepts in AMASS:

- DAF focuses on dependability processes and dependability cases, so its support to many evidence management activities is very limited, including evidence collection and traceability. These two activities are essential for seamless interoperability.
- There are very few usage examples, thus it is not clear how the specification can be applied.
- In line with the previous point, little validation has been performed on the specification. The determination of the benefits and possible weaknesses in using it needs further study.
- DAF overlaps with SACM and it is not clear how the overlap should be managed. For example, a user can easily wonder when DAF should be used instead of SACM for assurance/dependability case specification, and the management of the associated evidence.

3.10.2 Traceability and Automated Traceability

Traceability can be defined as the degree to which a relationship can be established between two or more products of a system's lifecycle (aka artefacts), especially products having a predecessor-successor or master-subordinate relationship to one another [42]. A trace can be defined as a specified triplet of elements comprising a source artefact, a target artefact and a trace link associating the two artefacts [43], where the source artefact is the origin of the trace, the target artefact is the destination of the trace, and the trace link is the specified association between the pair of artefacts.

Traceability is an important aspect for seamless interoperability in AMASS. First, traceability between different artefacts is prescribed in practically all assurance standards, information about such artefacts can be collected via tool interoperability mechanisms, and thus traceability information should, be managed in the AMASS platform. Second, it must be ensured that all the information managed as a result of the seamless interoperability is consistent and coherent with regard to the applicable assurance standards. Adequate traceability management in relation to the information imported and managed in the AMASS platform is a prerequisite for CPS certification.

A traceability process typically consists of several general activities [43]:

- Planning and management, concerned with the determination of stakeholder and system requirements for traceability, the design of a suitable traceability solution, and the provision of the control necessary to keep these requirements and solutions relevant and effective during the life of a project.
- Creation, which is the general activity of associating two (or more) artefacts by specifying trace links between them.
- Use, concerned with exploiting traces in various software and systems engineering activities and tasks, such as verification and validation, impact analysis, and change management.
- Maintenance, which consists of those activities associated with updating pre-existing traces as changes are made to the traced artefacts and the traceability evolves, creating new traces where needed to keep the traceability relevant and up to date.

Traceability can be handled in different ways [43]:

- Manually, when traceability is established by the activities of a human tracer. This includes traceability creation and maintenance using the drag and drop methods commonly found requirements management tools.
- Semi-automatically, when traceability is established via a combination of automated techniques, methods, and tools, and human activities. For example, automated techniques may suggest candidate trace links or suspect trace links and then the human tracer may be prompted to verify them.
- Automatically, when traceability is established completely via automated techniques, methods, and tools. Currently, it is the decision as to among which artefacts to create and maintain trace links

that is automated. Fully automatic traceability is probably not realistic for critical systems due to the stringent requirements on the tools used in a system's lifecycle and on the validity of their output.

By automated traceability, we refer to both semi-automatically and automatically traceability. AMASS could provide support to automated traceability so that traceability management becomes seamless, or at least more seamless, by relieving a user from manual tasks. This tasks can further be error-prone and time-consuming (e.g. trace verification), especially when taking into account the large number of artefacts to usually manage and trace for a critical system [44]. In an ideal scenario, a user could import information to the AMASS platform for assurance purposes from different engineering tools, and the AMASS platform could e.g. suggest and validate traces between artefacts.

We distinguish three main types of approaches for automated traceability:

- **Model-based approaches** (see e.g. [45]), whose main mean for traceability automation is the comparison of traceability (or artefact) information with a reference traceability model. This way, the approaches can e.g. suggest traces, identify missing traces, and validate traces.
- **Information retrieval-based approaches** (see e.g. [46]), which determine the extent to which two artefacts are related based on their textual data (sentences, terms, term structures, etc.). The assumption is that the higher the textual similarity between two artefacts, the higher the likelihood that a relationship exists between them.
- **Ontology-based approaches** (see e.g. [47]), which aim to overcome the limitations of information-retrieval-based approaches by exploiting the knowledge and semantic information about a given domain. For example, information retrieval-based approaches do not perform well when not the same terms but synonyms are used in two different artefacts.

3.10.3 Tool qualification

In the context of safety-critical systems engineering, software is increasingly developed and verified (semi)-automatically. Tools for code generation as well as for verification are introduced to (semi-) automate, replace, or supplement complex tasks. Since safety might be compromised if such tools fail, safety standards prescribe tool qualification processes [48].

Tool qualification can roughly be defined as the provision of formal assurance that a tool's output can be trusted, e.g. the object code that a compiler generates. For AMASS, tool qualification in the scope of WP5 mainly concerns the possible need of tool information, as part of the seamless interoperability, for assurance and certification of CPSs. For example, if Papyrus is part of a tool chain resulting from the enactment of the seamless interoperability approach, what information about Papyrus should be managed in the corresponding assurance project? On the other hand, the use of a qualified tool, e.g. the GNATcheck tool for static analysis of Ada programs, should lead to the management of its qualification dossier as part of the assurance project. Tool qualification processes deal with two categories of tools: development tools and verification tools.

Tool qualification processes typically consist of three phases [49]: classification, qualification, and usage. During the classification phase, the tools are classified according to the level of confidence that is required to ensure their behaviour is in-line with the safety requirements. Levels are named differently from one standard to another (see standard-specific information below). If a tool is considered to be harmless, it can be used without requiring any qualification. During the qualification phase, the tools that were considered potentially harmful have to be qualified, i.e. manufacturers have to show absence of hazardous events (failures that might lead to accidents). Finally, during the usage phase, tools can be used within the specified restrictions.

It could be an issue for seamless interoperability to keep track of tool qualification status/level and of which tools have had an impact on each artefact. This would be a problem for certification since someone might need to show that the tool qualification level is sufficient for the artefacts. With many tools interacting and manipulating the same data, it can be difficult to keep track of this manually.

It should be noted that tool qualification processes do not tackle tool-chains. In the context of AMASS, the qualification of the AMASS-platfom (seamless toolchain) is not the primary objective. Within AMASS, the requirements coming from the standards and pertaining to the tool qualification will be taken into consideration during the development in order to point out what should be done.

Overall, the above needs still require further investigation, especially in relation to industrial requirements from the case studies. Specific aspects for seamless interoperability related to tool qualification will be addressed in T5.2 (Conceptual Approach for Seamless Interoperability) and T5.3 (Implementation for Seamless Interoperability). In the remaining of this section, we present the main aspects to consider regarding tool qualification in three specific domains: automotive, avionics, and railway.

In automotive, ISO 26262-8 [50] defines three tool confidence levels (TCL 1-3) that depend on:

- Tool impact, related to the possibility that a malfunction of a particular software tool can introduce or fail to detect errors in a safety-related item or element being developed.
    - TI1 shall be selected when there is an argument that there is no such possibility
    - TI2 shall be selected in all other cases.
- Tool error detection, related to the confidence in measures that prevent the software tool from malfunctioning and producing corresponding erroneous output, or in measures that detect that the software tool has malfunctioned and has produced corresponding erroneous output.
    - TD1 shall be selected if there is a high degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected
    - TD2 shall be selected if there is a medium degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected
    - TD3 shall be selected in all other cases

Tool qualification in avionics currently is governed by the DO-330 standard [51]. It defines five tool qualification levels (TQL-1 to TQL-5), which are assigned to a given tool according to the software assurance level (A-D) and three criteria:

- Criterion 1: A tool whose output is part of airborne software and thus could insert an error
- Criterion 2: A tool that automates verification processes and could fail to detect an error
- Criterion 3: A tool that, within the scope of its intended use, could fail to detect an error

As a rule of thumb, Criterion 3 is related to computer-aided specification, Criterion 2 to V&V tools, and Criterion 1 to compilers.

For railway application, the only relevant standard for tool qualification is EN 50128:2011 [52], which provides engineering assistance tool requirements. Three tool classes are defined:

- T1, which is related to specification assistance (no safety impact in case of errors in this tool).
- T2, which is related to tools that if an error occurs, a safety requirement may be missed.
- T3, which is related to safe data computation.

The classes are therefore similar to those from avionics: T1 is related to computer-aided specification, T2 is related to test automation tools, and T3 to compilers.

Regarding the current support for tool qualification process, the Validas approach, tool, and services [53] appear to be the most advanced available support. Validas addresses tool chain analysis, construction of tool qualification kits, and application of tool qualification kits. It uses a model-based approach that provides high flexibility, reusability, and systematic processes, and generates the required tool qualification documents. How to use Validas and the possible integration of its open-source tools with the AMASS tool platform will be further studied in T5.2.

4. State of the Practice

This section outlines in which way(s) industry already deals with the problems of seamless interoperability, including how Industry standards “prescribe” practice, or how limitations and “proven in use” have shaped practice. As can be observed, some of the technologies reviewed in Section 3 have already stated to be introduced in practice for seamless interoperability, e.g. OSLC, whereas others have not yet.

The section focuses on the state of the practice for two main technologies: requirements management tools and medini tools (by KMT).

4.1 Interoperability of Requirements Management Tools

The REUSE Company (TRC) has created tools to assess requirement specifications based on NLP techniques. This assessment is based on analysis of three different points of view:

- Correctness: checks that the requirement statement is correct semantically and follow the grammar set by the company guide to write requirements.
- Completeness: checks that there are no gaps in the specification from many different perspectives: terminology, relationships with external interfaces, typology of the requirements, missing links between high-level requirements and low level requirements, etc.
- Consistency: checks that there are no contradictions between the requirements

These points of view take different approaches: correctness assesses an individual requirement and focuses on different characteristics found in the requirements (either in their statements or in their properties, history or attributes). Whereas the last two assess requirement specification sets as the minimum unit to be assessed focused on their transformation into a formal representation model.

These requirement specifications are stored in commercial tools, also known as Requirement Management Systems (RMS) or Requirement Management Tools (RMT), such as:

- Rational DOORS by IBM
- Visure Requirements by Visure Solutions
- Integrity by PTC
- Reqtify by Dassault Systèmes
- OSLC (not a tool as such but a technology for requirements information exchange)

TRC decided in the past to be on top of these tools to avoid redundancy problems and to focus in the quality assessment.

Thus, a requirement abstraction was designed to represent the requirements independently from their storage source: the TRC standard requirement object. The one and only way of retrieving the requirements from these RMSs is to use their proprietary APIs to manage properly.

Finally, the experience gained while developing RMSs integration may help during the development of the AMASS project. There have been some pros and some cons:

- Pros:
  - The official APIs are supported by the RMS suppliers, thus there are a customer support team behind them and they are updated and improved for each new RMS release.
  - They provide access to almost every piece of information managed in the tools.
  - This allows us to load, modify, create or delete requirements from their source without any intermediate storage. This technique allows that there are no inconsistencies and no synchronization process is needed.
- Cons:
  - A new connector must be created for each new integration. This implies studying and consuming a new APIs

  - Each time a supplier releases a new RMS version, the API may have suffered changes, so the integration needs to be tested again.

The next sections present more information about the integration of TRC tools with others and the main conclusions drawn.

4.1.1 Integrations

4.1.1.1 Integration with DOORS

The integration with DOORS has been done by using the DOORS client. This client is launched in a batch mode (as a background process without any user interface) and parameterized to execute a DXL script.
There were several functions to be implemented in DXL scripts to retrieve the necessary information about projects, folders, formal modules, link modules and requirements. Thus, several DXL scripts have been created.

These scripts retrieve the information from the DOORS database and store it in temporary files. After the DXL script execution has been finished, the control returns to the quality application and it loads the information from those temporary files and deletes them.

The pros and cons of this API integration are the following:

- Pros:
  - DXL scripts allow access to all the information available in DOORS Client with user interface.
- Cons:
  - DXL Scripts are hard to debug; the testing environment is very poor.
  - Each execution takes minimum 3 seconds. Even worse, we have experienced license-checking delays that increases batch mode response time dramatically, making it unbearable.
  - Some customers have deployments where DOORS client is not deployed in the customer local computer, but in a virtualized desktop or in a shared network path.
  - Most of the times, the DOORS client batch mode needs additional configuration to load formal modules.
    - The reason is that there are plugins on top of these formal modules.
    - These plugins are set up by additional parameters set up in the DOORS client start up.
    - These additional parameters must be also provided to the DOORS connection set up to be able to open those modules without error when DOORS is running in batch mode.
    - Any new plugin added to the environment which needs additional parameters needs to refresh the DOORS connection setup to modify these additional parameters
  - There are constant DXL memory leakages, thus the code has to be carefully designed to avoid them.
  - There are different interpretations of the RTF format used to store requirements in DOORS and the standard.
  - There are different interpretations of the OLE Objects format representation from DOORS and the standard.
  - There were some COM registration difficulties to enable to create an authoring plugin on top of DOORS Client, this time with user interface, not in batch mode.

4.1.1.2 Integration with Integrity

The integration with PTC Integrity has been done using the web service APIs that are available in the Integrity Server. Thus, there is no need to have the Integrity Client in the same computer as the Quality

Assessment tools are installed. Moreover, quite differently from DOORS, there is no need for scripts and temporary files, which makes the integration easier and faster in regard to response time.

The pros and cons of this web service API integration are the following:

- Pros:
  - Smooth integration with our desktop applications using web services
  - The Integrity data metamodel is quite flexible to allow all of our operations
  - There are several web services to support all the functionality needs, thus allowing access to all the data available in Integrity
  - No need for scripts or intermediate files
- Cons:
  - Even if these web services are official by the tool vendor, there were not enough documentation to access to all needed info and it has been discovered by test and error methodology
  - There a small gap that disables the possibility of automating the deployment of the authoring plugin on top of Integrity.

4.1.1.3 Integration with Visure Requirements

The integration with Visure Requirements (VR) has been achieved by using their proprietary API called VR Face, which is a COM object in Windows architecture. Thus, there is no need of scripts or intermediate temporary files. But, on the contrary of the integration made with Integrity, it's needed that the VR client and the VR Face is installed on the same computer as the Quality Assessment tools are installed.

The pros and cons of this VR API are the following:

- Pros:
  - This VR Face API allows a smooth integration with our desktop applications
  - This API allows access to all the requirement data available in VR client
  - The VR data metamodel is quite flexible
  - No need for intermediate files
- Cons:
  - The VR Face API is complex to understand and manage properly
  - The VR Face has some performance issues when dealing with attribute management.

4.1.1.4 Integration with Reqtify

The integration with Reqtify has been different from those done with other RMS providers because Reqtify does not have a proper API. The integration was done after some negotiations with Dassault Systèmes (DS). It was agreed to exchange information using XML files created within Reqtify under a new Reqtify license option. These XML files are not using any standard requirements format, and they do not allow our Quality Suite to perform any modification in the source document.

From these inputs the integration was done using our Excel connection already available by transforming the XML file into an Excel file with a predefined configuration to match all the information available in the XML file. All the operations that Reqtify does not allow to the Quality Suite are performed in the Excel intermediate file (which is not deleted after quitting the Quality Suite).

The pros and cons of this VR API are the following:

- Pros:
  - Easy to develop integration using the integration with Excel as an intermediate step.
- Cons:
  - Reqtify does not have a proper API: Exchange of requirement information using ad hoc XML files created by TRC-Dassault agreement
  - The metamodel is not flexible
  - There is only one spec by the project
  - Need to have Reqtify with RQS license on the same computer

- Need to have MS Excel on the same computer
- Any modification of the metamodel has to be agreed by TRC and Dassault, and then released by Dassault in the following release

4.1.1.5 Integration with OSLC

The integration with OSLC has been done by using the web services exposed by the standard OSLC definition. Thus is quite similar to the integration done for PTC Integrity.

The pros and cons of this OSLC API are the following:

- Pros:
  - Platform independent definition on requirements: OSLC RM
- Cons:
  - The only implementation of OSLC RM is DOORS Next Generation the Jazz platform
  - Lack of enough documentation
  - The metamodel API is not flexible: some mode cannot be modified
  - Performance issues by request
  - There are not API functions to perform complex operations in an atomic step, thus increasing the numbers of operations to be done and then struggling with performance issues by each request
  - Bad API Error reporting on some functions
  - Problems with the authentication mechanism
  - Lack of authoring identification

4.1.2 Conclusions

1. Each new RMS integration needs to create a new connector
2. Each new RMS release needs to perform the testing the integration
3. OSLC RM connector was implemented to avoid the above issues, provided that new RMS implements the OSLC layer
4. Each connector uses either APIs, WS or files to exchange the information between RMS and Quality Analyzer tool

4.2 Tool integration experiences in medini tools

As a safety assurance tool for the application of ISO 26262, medini analyze of KMT is encouraged to interoperate with many tools and file formats in different technologies and formats, some of these technologies out there for a long time already. The tool integrates with a couple of other analysis, but also design or RM tools. The experiences made during development and field application are outlined in this section.

4.2.1 Integration via API (COM)

Many tool vendors that – among them big players as IBM, Mathworks, Atego and ETAS) – implement rich client tools on Windows OS using .NET/C++ or similar languages. They naturally and historically support Windows COM based APIs. When integrating with IBM rational Rhapsody for example, to retrieve and access design models and information, the integration has to be done via their COM interface. Since Rhapsody offers an official Java API that in principle wraps the COM interface, the assumption (and later decision) was to utilize this API.

Experiences made:

- Rhapsody Java/COM Bridge assumes an open and running Rhapsody (active project is connected)

- Official Java integration was bridged via COM, native runtime libraries have to be integrated in the integrating tool at runtime (DLL hell), many compatibility problems when connecting different Rhapsody versions (tool stability)
- The API is rich and huge, access to almost all information are at hand
- IBM fosters the OSLC way since some years, Rhapsody has partial support but OSLC since version X, maybe will be the future also in Rhapsody

Pro:

- Official API supported by the tool supplier
- Almost all information can be accessed and even manipulated (written)
- Model files will be managed and saved by the tools, we don't have to care about migration, difference in versions etc.

Con:

- Runtime and integration problems with Java/COM bridge, may destabilize own tool
- API is proprietary and completely tool specific, so another vendor/tool yet another API/COM bridge…

4.2.2 Integration with standardized XML

There are some related official XML standard exchange formats for engineering (automotive) like RIF, MSR-MEDOC and others. A promising candidate to integrate with RM tools is RIF (the Requirements Exchange Format) as defined by the HIS group [54] and its successor ReqIF. The goal was to enable bi-directional (import/export) integration with any RM tool in a standardized way (DOORS, MKS, etc.), e.g. to link existing non-safety requirements with derived safety requirements and to push back safety requirements to RM system. The assumption was that, by using a standard file exchange format, any tool that imports or exports requirements is on the safe side and can claim interoperability with all RM tools. The most stable version RIF 1.2 at that time was used.

Experiences:

- There weren't many tools that did implement RIF v1.2, many stopped implementation at 1.0/1.1
- RIF is underspecified and gives room for interpretation, in parts it defines the structure but lacks semantics
- DOORS (Telelogic) as early adopter and member of standardization, but implemented many DOORS specific extensions (so we had to support them as well)
- DOORS was main target of the integration, as a result the RIF IO was rather DOORS specific
- RIF was introduced with DOORS 9.2, earlier version did not support RIF
- Some companies internally forbid the usage of RIF (still don't know the reason behind)

Pro:

- Standardized interchange format, fits for "openness" idea
- Import of all required information was possible
- RIF was designed as interchange format, so bi-directional use cases are explicitly addressed

Con:

- Lacked support by tool vendors
- Underspecified in semantic details, tool specific interpretation of RIF had to be implemented
- Differences in versions have to managed by ourselves
- RIF was made for supplier tool chains (pass requirements down, enrich them and pass back up) not for general roundtrip as we planned it

4.2.3 Integration via Proprietary File Format

In a few cases, it is required to read proprietary (model) files directly, rather than integrating via an API. Examples for that kind of integration are Matlab Simulink and Enterprise Architect (EA). The goal of an integration was to unidirectional import design models into the safety assurance tool, as a side effect transform the models if necessary. The assumptions were to bypass the (available) COM communication to be able to import models without having tools installed and running locally. Main reason was that both COM APIs were treated as slow and buggy. The Simulink file format was good to parse and read and out there for almost 20 years, so assumed to be stable.

Experiences:

- Many “ugly” hacks and workarounds in the Simulink file, mostly for historical reasons, fixed in the Matlab tool but still there in the files
- Simulink file format switch to XML (after 20years) recently (still proprietary!)
- EA files are Access database files, most data are stored in generic fields and columns, lots of interpretation by the EA tool
- DB usage mostly for performance reasons, EA COM API for external tools is buggy

Pro:

- Being able to read file content gives great independence on the runtime tool
- File reveals more details than the public API
- Fast, parser usually quicker than COM interface

Con:

- Many pieces of information are “interpreted” by the tools but not stored directly in the files. We have to adapt that interpretation, and even worse, we first have to “know” it, which is quite a difficult task without having the specification. In the end, it may lead to an incorrect interpretation and incorrect handling of the data.
- Proprietary file formats may change at any time, high risk to re-implement import code
- We do not get official support, many undocumented features, error prone
- Different versions in the files
- EA database not always in sync with model, may contain obsolete data which was not removed by EA due to performance reasons (ok for EA but not for us)

4.2.4 Conclusions

1. Proprietary point-to-point integration are not cost effective, regardless of the integration approach (COM, file format etc.)
2. Even standardized exchange formats do not necessarily reduce the integration effort
3. Complete data import allows self-contained projects but leads to update and merge problematic later and to adaptation to local data structures

5. Consolidation and Way Forward

In this deliverable, the partners investigated seamless interoperability, which is one of the central objectives of the AMASS project:

"Develop a fully-fledged open tool platform that will allow developers and other assurance stakeholders to guarantee ***seamless interoperability*** of the platform with other tools used in the development of CPSs."

In the Scientific and Technical Objectives (STOs) of the project, this general objective includes:

- ***Tool Integration***
- ***Collaborative Work Management***
- ***Tool Quality Assessment and Characterization***

It was necessary to gain an overview of the available technologies that could be used and further elaborated in AMASS to meet these three objectives. Such technologies form the state of the art. Furthermore, it was investigated what techniques and technologies are used today in real projects – the state of the practice. Where necessary, the investigation was done with experiments to underpin the results.

The experiments of the AMASS partners yielded a significant gap between the state of the art and the state of the practice. For historical reasons, many tools are still being implemented as rich clients using relational databases or XML files for data integration. These technologies do not naturally support change management; therefore, a version control layer is usually added. ALM tools historically have struggled to seamlessly link into configurable tool chains because the underlying technologies are geared towards monolithic rich clients, effectively creating isolated data silos.

Table 3 summarizes results obtained from the investigation of the state of the art and of the state of the practice in relevant areas of seamless integration.

Table 3. Aspects of seamless interoperability – state of the art vs. state of the practice

	State of the Art	**State of the Practice**
Authentication	Single sign-in	Sign-in for every tool
Collaboration	Live collaboration on same set of data	Diff/merge (complicated and error-prone) or locking (not really collaboration)
Change management	Complete history of changes with manually created baselines	Daily check-ins to repository
Installation	No installation required	Installation on hard-drive with installer
Data changes	Push notifications across tool borders	Manual data import and export
Data exchange	Single source of truth	COM, XML, proprietary file format
Tool integration	Standardized data bus (Tasktop Sync, ModelBus, OSLC)	Point to point

Using modern web technologies, it would be possible to close the gaps between tools and allow for a seamless integration. It should be noted that the classic software frameworks for rich clients should not be discarded altogether. Their level of maturity is not currently matched by web technologies. Therefore, it might be possible to apply web-based frameworks to enable new and powerful features as live collaboration whenever it is feasible without forcing that issue. This approach will not end up in a bloated "one size fits all" rich client with a huge technology stack, but multiple smaller and more flexible applications, some of them being web-based.

The identified gaps between the state of the art and the state of the practice have been consolidated to a number of high-level needs formulated from a user's perspective as user-stories (Table 4). Generally,

current pain points such as lack of team work, difficult configuration, and manual interactions should be removed to enable a more focused working style of the user without being interrupted to solve shortcomings of the applied tools. These user stories synthesise the way forward for seamless interoperability and will be used as starting point for the AMASS requirements to be specified in WP2.

Table 4. Way forward for seamless interoperability: User stories

ID number to reference	**User Story** short user story (or use case description or requirement about seamless interoperability).
Template	**As a** tool user working with a dedicated application **I want** to just login once (have single-sign-on) **so that** I can just use all tools I need to do my work after first login.
US_01	As a tool user working with evidence management I want to point to specify information about a Section of a given document (e.g., A System Requirement specified inside a MS Word document) so that I can refer/point out to this section for change management, traceability, etc.
US_02	As a tool user working with some functionalities (e.g., compliance management, reports, metrics) I want to get access to information from Web so that I can know this information in real-time as it is being edited by any other user
US_03	As a tool auditor I want to know any change on the data managed by the tools including authors, date and content so that I can assess its confidence and traceability
US_04	As a tool user I want to access the tools data concurrently with other users so that the integrity of the data is guaranteed and that I am aware of the concurrence modifications rules and effects
US_05	As a tool manager I want to grant access to users according to (a) tool functionality, (b) type of information (e.g., specific project, date range) so that users get access according to their profiles.
US_06	As a tool manager I want data to be readily available in non-proprietary formats.
US_07	As a tool user I want to create and enter data only once.
US_08	As a tool user I want to have metrics and measurements generated and reported.
US_09	As a tool manager I want to minimize the number of data management and lifecycle tools.
US_10	As a tool auditor I want automatic collection of lifecycle and status data in a transparent way as part of workflow.
US_11	As a tool user I want data to move through process with minimal manual intervention.
US_12	As a tool manager I want continuous analysis, verification and integration of the data.

Abbreviations

Abbreviation	Full Name	URL
ALM	Application Lifecycle Management	
API	Application Programming Interface	
ARTA	AMASS Reference Tool Architecture	
DAF	Dependability Assurance Framework for Safety-Sensitive Consumer Devices	www.omg.org/hot-topics/cdss.htm
DXL	DOORS eXtension Language	
GSN	Goal Structuring Notation	www.goalstructuringnotation.info
IoT	Internet of Things	
NLP	Natural Language Processing	
OMG	Object Management Group	www.omg.org
OSLC	Open Services for Lifecycle Collaboration	open-services.net
REST	Representational State Transfer	
RDF	Resource Description Framework	www.w3.org/RDF
SACM	Structured Assurance Case Metamodel	www.omg.org/spec/SACM
SPEM	Software & Systems Process Engineering Metamodel	www.omg.org/spec/SPEM/2.0
UI	User Interface	

References

[1] "Open Source solutions for Systems Engineering and Embedded Systems," Polarsys Open Source Solutions for Embedded Systems, [Online]. Available: https://www.polarsys.org/. [Accessed 2016].

[2] "Eclipse EMFStore," Eclipse Foundation, [Online]. Available: http://www.eclipse.org/emfstore/.

[3] "Eclipse CDO," Eclipse Foundation, [Online]. Available: https://wiki.eclipse.org/CDO.

[4] J. Mauersberger and T. Richardson, "Prototype implementation of tools for Argumentation/Compositional Certification D5.5," 2015. [Online]. Available: http://www.opencoss-project.eu/sites/default/files/D5.5_OPENCOSS_V1_1.pdf.

[5] "Eclipse Dawn," Eclipse Foundation, [Online]. Available: http://wiki.eclipse.org/Dawn.

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "A View of Cloud Computing," *Commun. ACM,* vol. 53, pp. 50-58, #apr# 2010.

[7] "Open Services for Lifecycle Collaboration," [Online]. Available: http://open-services.net/.

[8] T. Berners-Lee, "Linked Data," 27 7 2006. [Online]. Available: https://www.w3.org/DesignIssues/LinkedData.html. [Accessed 28 9 2016].

[9] C. Bizer, T. Heath and T. Berners-Lee, Linked Data - The Story So Far, International Journal on Semantic Web and Information Systems, 2009.

[10] D. Beckett, "RDF 1.1 XML Syntax," 25 2 2014. [Online]. Available: http://www.w3.org/TR/rdf-syntax-grammar/. [Accessed 2016 9 28].

[11] J. M. Alvarez-Rodríguez, J. Llorens, M. Alejandres and J. Fuentes, "OSLC-KM: A knowledge management specification for OSLC-based resources.," *INCOSE International Symposium ,* vol. 25, no. 1, p. 16–34, 2015.

[12] N. Noy and A. Rector, "Defining N-ary Relations on the Semantic Web," 12 4 2006. [Online]. Available: http://www.w3.org/TR/swbp-n-aryRelations/. [Accessed 29 09 2016].

[13] A. Mallea, M. Arenas, A. Hogan and A. Polleres, "On blank nodes," *The Semantic Web–ISWC 2011,* vol. 1, no. 1, pp. 421-437, 2011.

[14] D. Martin, J. Domingue, A. Sheth, S. Battle, K. Sycara and D. Fensel, "Semantic Web Services," *IEEE Intelligent Systems,* vol. 22, no. 6, pp. 8-15, 2007.

[15] M. García-Rodríguez, J. M. Alvarez-Rodríguez, D. Berrueta, L. Polo, P. Ordoñez de Pablos and J. E. Labra-Gayo, "Towards a Practical Solution for Data Grounding in a Semantic Web Services Environment," *J. UCS (JUCS),* vol. 18, no. 11, pp. 1576-1597, 2012.

[16] V. Castañeda, L. Ballejos, L. Caliusco and R. Galli, "The Use of Ontologies in Requirements Engineering," *Global Journal of Researches In Engineering,* vol. 10, no. 6, 2010.

[17] M. Kossman, R. Wong, M. Odeh and A. Gillies, "Ontology-driven Requirements Engineering: Building the OntoREM Meta Model," *IEEE,* no. 1, pp. 1-6, 2006.

[18] D. Gaševic, V. Devedžic and D. Djuric, "Model Driven Architecture and Ontology Development," *SpringerLink,* 2006.

[19] B. Gallina and Z. Szatmari, "Ontology-based Identification of Commonalities and Variabilities among Safety Processes," in *16 th International Conference on Product-Focused Software Process Improvement*, 2015.

[20] "Integrity," PTC Inc., [Online]. Available: http://www.ptc.com/application-lifecycle-management/integrity.

[21] "Teamcenter : Siemens PLM Software," Siemens Product Lifecycle Management Software Inc., [Online]. Available: http://www.plm.automation.siemens.com/en_us/products/teamcenter/.

[22] "Jazz Community Site," International Business Machines Corporation, [Online]. Available:

https://jazz.net.

[23] "HP Product Lifecycle Management (PLM)," Hewlett Packard Enterprise Development LP, [Online]. Available: http://www8.hp.com/de/de/business-services/it-services.html?compURI=1091473.

[24] B. Gallina, J. P. Castellanos Ardila and M. Nyberg, "Towards Shaping ISO 26262-compliant Resources for OSLC-based Safety Case Creation," in *CARS 2016 - 4th International Workshop on Critical Automotive Applications: Robustness & Safety*, 2016.

[25] B. Gallina and M. Nyberg, "Reconciling the ISO 26262-compliant and the agile documentation management in the Swedish context," in *CARS 2015 - Critical Automotive applications: Robustness & Safety*, Paris, 2015.

[26] M. Bender, T. Maibaum, M. Lawford and A. Wassyng, "Positioning Verification in the Context of Software/System Certification," in *Proceedings of the 11th International Workshop on Automated Verification of Critical Systems (AVOCS 2011)*, 2011.

[27] "Eclipse Lyo," Eclipse Foundation, [Online]. Available: http://www.eclipse.org/lyo/.

[28] "Xtext Web Editor Support," Eclipse Foundation, [Online]. Available: http://www.eclipse.org/Xtext/documentation/330_web_support.html.

[29] "mxGraph Version 3.6.0.0 – 07. September 2016," JGraph Ltd., [Online]. Available: https://jgraph.github.io/mxgraph/.

[30] "Google Realtime API: Conflict Resolution and Grouping Changes," Google Inc., [Online]. Available: https://developers.google.com/google-apps/realtime/conflict-resolution.

[31] Jong-yi Hong, Eui-ho Suh and Sung-Jin Kim, "Context-aware systems: A literature review and classification," *Expert Systems with Applications,* vol. 36, no. 4, pp. 8509-8522, 2009.

[32] U. Alegre, J. C. Augusto and T. Clark, "Engineering context-aware systems and applications: A survey," *Journal of Systems and Software,* vol. 117, 2016.

[33] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive and Mobile Computing,* vol. 6, no. 2, pp. 161-180, 2010.

[34] J. Ye, S. Dobson and S. McKeever, "Situation identification techniques in pervasive computing: A review," *Pervasive Mob. Comput.,* vol. 8, no. 1, pp. 36-66, 2012.

[35] "A survey on context-aware web service systems," *International Journal of Web Information Systems,* vol. 5, no. 1, pp. 5-31, 2009.

[36] "https://developers.google.com/awareness/overview#context-types," [Online].

[37] S. Nair, J. L. de la Vara, M. Sabetzadeh and L. C. Briand, "An extended systematic literature review on provision of evidence for safety certification," *Information & Software Technology,* vol. 56, pp. 689-717, 2014.

[38] "Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0," Object Management Group, 2008. [Online]. Available: http://www.omg.org/spec/SPEM/2.0/.

[39] K. Attwood, "Common Certification Language: Conceptual Model," 2015. [Online]. Available: http://www.opencoss-project.eu/sites/default/files/D4.4_v1.5_FINAL.pdf.

[40] "Structured Assurance Case Metamodel (SACM) Version 1.1," Object Management Group, 2015. [Online]. Available: http://www.omg.org/spec/SACM/.

[41] "Dependability Assurance Framework For Safety-Sensitive Consumer Devices (DAF) Version 1.0," Object Management Group, 2016. [Online]. Available: http://www.omg.org/spec/DAF/.

[42] "IEEE Standard Glossary of Software Engineering Terminology," IEEE, 1990.

[43] J. Cleland-Huang, O. Gotel and A. Zisman, Software and Systems Traceability, Springer Publishing Company, Incorporated, 2012.

[44] J. L. de la Vara, M. Borg, K. Wnuk and L. Moonen, "An Industrial Survey of Safety Evidence Change Impact Analysis Practice," *IEEE Transactions on Software Engineering,* 2016.

[45] I. Santiago, Á. Jiménez, J. M. Vara, V. De Castro, V. A. Bollati and E. Marcos, "Model-Driven Engineering As a New Landscape for Traceability Management: A Systematic Literature Review," *Inf. Softw. Technol.,* vol. 54, pp. 1340-1356, 2012.

[46] M. Borg, P. Runeson and A. Ardö, "Recovering from a Decade: A Systematic Mapping of Information Retrieval Approaches to Software Traceability," *Empirical Softw. Engg.,* vol. 19, pp. 1565-1616, 2014.

[47] J. Guo, N. Monaikul, C. Plepel and J. Cleland-Huang, "Towards an Intelligent Domain-specific Traceability Solution," in *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, New York, NY, USA, 2014.

[48] B. Gallina, S. Kashiyarandi, K. Zugsbrati and A. Geven, "Enabling Cross-domain Reuse of Tool Qualification Certification Artefacts," in *1st International Workshop on DEvelopment, Verification and VAlidation of cRiTical Systems, SAFECOMP Workshop*, 2014.

[49] O. Slotosch, "Model-Based Tool Qualification: The Roadmap of Eclipse Towards Tool Qualification," in *Information Technology and Open Source: Applications for Education, Innovation, and Sustainability: SEFM 2012 Satellite Events, InSuEdu, MoKMaSD, and OpenCert Thessaloniki, Greece, October 1--2, 2012 Revised Selected Papers*, A. Cerone, D. Persico, S. Fernandes, A. Garcia-Perez, P. Katsaros, A. S. Shaikh and I. Stamelos, Eds., Berlin, Heidelber: Springer Berlin Heidelberg, 2014, pp. 215-228.

[50] *ISO 26262-8:2011-11 – Road vehicles – Functional safety – Part 8: Supporting processes,* International Organization for Standardization, 2011.

[51] *DO-330, Software Tool Qualification Considerations,* RTCA & EUROCAE, 2011.

[52] *EN 50128:2012-03, Railway applications - Communications, signalling and processing systems - Software for railway control and protection systems,* CENELEC, European Committee for Electrotechnical Standardization, 2012.

[53] "Validas Tool Qualification Services," Validas AG, [Online]. Available: http://www.validas.de/toolqualification.html.

[54] „HIS (Herstellerinitiative Software)," HIS (automotive group), [Online]. Available: http://www.automotive-his.de/.

[55] K. Padira, "Investigation of Resources Types for OSLC domains Targeting ISO 26262," 2016.

[56] B. Gallina, K. Padira and M. Nyberg, "Towards an ISO 26262-compliant OSLC-based Tool Chain Enabling Continuous Self-assessment," in *10th International Conference on the Quality of Information and Communications Technology- Track: Quality Aspects in Safety Critical Systems*, 2016.

[57] J. P. Castellanos Ardila, "Investigation of an OSLC-domain targeting ISO 26262," 2016.