A reference architecture for archival systems with application to product models

Raphaël Barbau

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE BOURGOGNE

A Reference Architecture for Archival Systems with application to product models

Raphaël BARBAU

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE BOURGOGNE

THÈSE présentée par

Raphaël BARBAU

pour obtenir le

Grade de Docteur de
l'Université de Bourgogne

Spécialité : **Informatique**

A Reference Architecture for Archival Systems with application to product models

Soutenue le 5 Juillet 2013 devant le Jury :

Lionel ROUCOULES	Rapporteur	Professeur, ENSAM Aix-en-Provence
Yacine OUZROUT	Rapporteur	Maître de conférence HDR, IUT Lumière - Université Lyon 2
Benoît EYNARD	Examinateur	Professeur, Université de Technologie de Compiègne
Abdelaziz BOURAS	Examinateur	Professeur, IUT Lumière - Université Lyon 2
Sebti FOUFOU	Directeur de thèse	Professeur, Université de Bourgogne

Contents

List of Figures

Acronyms

ACTDR	Audit and Certification of Trustworthy Digital Repositories
ADM	Architectural Development Method
AIP	Archival Information Package
AP203	Application Protocol 203 – Configuration controlled 3D design of mechanical parts and assemblies
AP214	Application Protocol 214 – Core data for automotive mechanical design processes
AP215	Application Protocol 215 – Ship arrangement
AP216	Application Protocol 216 – Ship moulded forms
AP218	Application Protocol 218 – Ship structures
AP227	Application Protocol 227 – Plant spatial configuration
AP239	Application Protocol 239 – Product life cycle support
AP	Application Protocol
API	Application Programming Interface
AV-1	Overview and Summary Information
AV-2	Integrated Dictionary
AV	All Viewpoint
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CCSDS	Consultative Committee for Space Data Systems
CDM	Conceptual Data Model
CPM	Core Product Model
COP	Chain of Preservation
CPM	Core Product Model
CV-1	Vision
CV-2	Capability Taxonomy
CV-3	Capability Phasing
CV-4	Capability Dependencies
CV-5	Capability to Organizational Development Mapping
CV-6	Capability to Operational Activities Mapping
CV-7	Capability to Services Mapping

CV	Capability Viewpoint
CWA	Closed World Assumption
DEX	Data EXchange Specification
DI	Descriptive Information
DIP	Dissemination Information Package
DIV-1	Conceptual Data Model
DIV-2	Logical Data Model
DIV-3	Physical Data Model
DIV	Data and Information Viewpoint
DL	Description Logics
DM2	DoDAF Meta-Model
DoDAF	Department of Defense Architecture Framework
DoD	Department of Defense
EA	Enterprise Architecture
EAF	Enterprise Architecture Framework
ERP	Enterprise Resource Planning
FBS	Function-Behavior-Structure
FBS-PPRE	Function-Behavior-Structure Processes-Products-Resources-External effects
FEA	Finite Element Analysis
FOXML	Fedora Object XML
ID	Identifier
InterPARES	International Research on Permanent Authentic Records in Electronic Systems
ISE	Integrated Shipbuilding Environment
ISO	International Organization for Standardization
IT	Information Technology
JPEG	Joint Photographic Experts Group
LDM	Logical Data Model
LEAPS	Leading Edge Architecture for Prototyping Systems
LOTAR	LOng Term Archiving and Retrieval

LTAR	Long Term Archiving and Retrieval
MDA	Model-Driven Architecture
METS	Metadata Encoding and Transmission Standard
MIME	Multipurpose Internet Mail Extensions
MODAF	Ministry of Defence Architecture Framework
MOF	Meta-Object Facilities
MOSLA	Maturity Of Standard for Long Term CAD Data Archiving
NAF	NATO Architecture Framework
NC	Numerical Control
NIST	National Institute of Standards and Technology
OAIS	Open Archival Information System
OAIS RM	Reference Model for an Open Archival Information System
OAM	Open Assembly Model
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group
OV-1	High-Level Operational Concept Graphic
OV-2	Operational Resource Flow Description
OV-3	Operational Resource Flow Matrix
OV-4	Organizational Relationships Chart
OV-5a	Operational Activity Decomposition Tree
OV-5b	Operational Activity Model
OV-6a	Operational Rules Model
OV-6b	State Transition Description
OV-6c	Event-Trace Description
OV	Operational Viewpoint
OWA	Open World Assumption
OWL	Web Ontology Language
Part21	Clear text encoding of the exchange structure
Part28	XML representations of EXPRESS schemas and data, using XML schemas
PDF	Portable Document Format

PDI	Preservation Description Information
PDM	Product Data Management
PES	Physical Exchange Specification
PID	Persistent Identifier
PLCS	Product Lifecycle Support
PLib	Part Library
PLM	Product Lifecycle Management
PSL	Process Specification Language
PV-1	Project Portfolio Relationships
PV-2	Project Timeline
PV-3	Project to Capability Mapping
PV	Project Viewpoint
QVT	Query/View/Transformation
RAAS	Reference Architecture for Archival Systems
RDF	Resource Description Framework
RDL	Reference Data Library
RI	Representation Information
RL	Rule Language
RM	Reference Model
SIP	Submission Information Package
SOAML	Service-Oriented Architecture Modeling Language
SOA	Service-Oriented Architecture
SPARQL	SPARQL Protocol and RDF Query Language
SQWRL	Semantic Query-Enhanced Web Rule Language
StdV-1	Standards Profile
StdV-2	Standards Forecast
StdV	Standards Viewpoint
S-TEN	Intelligent Self-describing Technical and Environmental Networks
STEP	Standard for the Exchange of Product model data
SV-10a	Systems Rules Model

SV-10b	Systems State Transition Description
SV-10c	Systems Event-Trace Description
SV-1	System Interface Description
SV-2	Systems Resource Flow Description
SV-3	Systems-Systems Matix
SV-4	Systems Functionality Description
SV-5a	Operational Activity to Systems Function Traceability Matrix
SV-5b	Operational Activity to Systems Traceability Matrix
SV-6	Systems Resource Flow Matrix
SV-7	Systems Measures Matrix
SV-8	Systems Evolution Description
SV-9	Systems Technology & Skill Forecast
SvcV-10a	Services Rules Model
SvcV-10b	Services State Transition Description
SvcV-10c	Services Event-Trace Description
SvcV-1	Services Context Description
SvcV-2	Services Resource Flow Description
SvcV-3a	Systems-Services Matrix
SvcV-3b	Services-Services Matrix
SvcV-4	Services Functionality Description
SvcV-5	Operational Activity to Services Traceability Matrix
SvcV-6	Services Resource Flow Matrix
SvcV-7	Services Measures Matrix
SvcV-8	Services Evolution Description
SvcV-9	Services Technology and Skill Forecast
SvcV	Services Viewpoint
SV	Systems Viewpoint
SWRL	Semantic Web Rule Language
SysML	Systems Modeling Language
TIFF	Tag Image File Format

TOGAF	The Open Group Architecture Framework
TWR	Torpedo Weapon Retriever
UML	Unified Modeling Language
UPDM	Unified Profile for DoDAF/MODAF
URI	Uniform Resource Identifier
VDA	Verband der Automobilindustrie
W3C	World Wide Web Consortium
XML	eXtended Markup Language
XSD	XML Schema Definition

1

INTRODUCTION TO DIGITAL PRESERVATION

Preserving information means making sure this information stays understandable and accessible over time. Preservation is a complex topic that encompasses many different issues. This thesis aims to present and address two specific problems related to the preservation of digital information. Section 1.1 provides a generic presentation of digital preservation. Then, Section 1.2 presents the main problems related to the description of archival systems. Section 1.3 presents the main problems related to the representation of digital information. Finally, Section 1.4 presents the approach used to solve these problems.

1.1/ BACKGROUND INFORMATION ON DIGITAL PRESERVATION

To be preserved, information has to be recorded on a medium. The medium hosts a physical representation of the information: data. According to the Reference Model for an Open Archival Information System (OAIS RM)[1], data is defined as a reinterpretable representation of information in a formalized manner suitable for communication, interpretation, or processing. Although information is abstract, data is something concrete, that is materialized. Data has to be interpreted to be recognized as information. Data interpretation differs depending on the containing medium, as specific technology might be needed to transform the data. For example, a page on a book simply requires recognition of the characters and knowledge of the language to be interpreted. But a video tape requires additional tools to produce video and audio, which can then be interpreted by humans. Similarly, a digital file has to be first processed by a computer to be interpreted. Once data is finally interpretable, knowledge is necessary for human to add meaning to this data so that it becomes information.

The amount of knowledge required to interpret the data may vary: for instance, reading a text requires knowledge of the alphabet and of the language, while watching a landscape painting does not require any particular knowledge. Reading a children's story requires less expertise than reading a journal paper in astrophysics. Any knowledge that helps human to interpret the information should be preserved as well. Such knowledge includes descriptions of the technology involved for representing the information. But such knowledge includes also, for example, the language, information models, and specific vocabulary that allow people to correctly understand the data.

Preservation activities are needed because time affects the ability to interpret the data.

Over time, the medium that holds the data can be altered. For example, paper and ink are materials that can easily deteriorate depending on the environment. Of course, not all the media are affected in the same manner. Engraving information on stones leads to less potential degradation due to the environment. But as a trade-off, using stones is much less convenient than using paper for writing, or transporting the information: it is more voluminous and heavier. Actually the problem exists because in many occasions, the most convenient way to express the information is not the most convenient way to preserve it.

Over time, the knowledge required to interpret the data may disappear. Finding an ancient text that is left unaltered over time does not mean this text can be interpreted. If the language used to interpret the text is no longer known, it will not be possible to get any information out of the medium. This is what happened for Egyptian texts before the meaning of hieroglyphs was discovered. A similar problem exists with digital files. Simply storing a file will not be sufficient if, after a certain period of time, software able to interpret the data no longer exists, and the format cannot be migrated.

One way to deal with knowledge evolution is to express the information in different forms. For example, the information represented in the Rosetta Stone was written in three different languages [2]. This allowed archaeologists to understand the meaning of hieroglyphs. Similarly, digital information can be represented in different formats, and can be held on different media.

Over time, the technology used to store information evolves, and new technology quickly replaces old technology. So the means used to record and access the information may become quickly obsolete, and migration may be needed. The way information is recorded has an influence on its preservation. Digital information has become critical in today's world. One potential risk in the future is the inability to retrieve information from digital objects, making people enter into a Digital Dark Age [3].

Digital information can be easily reproduced and transformed thanks to computers. Computers are systems, composed of hardware and software, that automatically process digital information. Computers are widely adopted because they save people a lot of time. For example, a text stored on a hard drive can be distributed over the network and read by numerous people at the same time. From a preservation perspective, using digital information allows the use of automated tools to reproduce and migrate the data, facilitating some aspects of the preservation processes. Computers can also be leveraged to automatically look into the content to retrieve specific information. If a file is stored as a text, it is possible to quickly determine if this file contains specific words. Digital information processing can also be used to produce audio, video, and interactive contents.

However, using the digital representation has several drawbacks compared with the traditional paper-based representation. First, the ease of information creation, reproduction, and transformation has led to an increasing amount of information. Because of this increasing amount, managing the contents has become harder, despite the ability to automatically process the information. Organizations become more and more complex, and so do the activities related to the archives: submission, access, and preservation of the information.

Computers are the only means by which people can manage digital information. So, in the context of an archive, they have to be designed to allow people to submit, preserve, and retrieve information. These computers will be referred to as archival systems in the rest of the thesis. While sometimes systems can refer to people and computers, in this

thesis, archival systems refer to computers and computers applications only.

From a computer science perspective, two particular aspects need to be considered to perform digital preservation. The first one is the development of archival systems able to meet the needs of the preservers, and the second is the selection of a representation for the content.

Describing archival systems is important to demonstrate that the system can support the preservation activities, and make preserved information both understandable and accessible. Archival system descriptions include the interactions with humans and other systems, the information processed by the system, and the functions performed by the system. These descriptions can have various levels of details, from high-level views that summarize the main elements of the system, to low-level views that show precisely how each aspect is actually implemented.

Another aspect of digital preservation is the representation of the information. Information has to be stored in determined formats so that it can be processed by computers. These formats have different characteristics. Some of these characteristics facilitate the preservation, some other characteristics make preservation impossible. Some formats require expensive software to interpret the data, while other formats are self-describing and do not require any particular software. Some formats have a long lifetime and may be supported during decades, while other formats have a short life and will have to be transformed (migrated) to a new format. Some formats emphasize knowledge representation, and can represent semantic relationships among different files.

Digital preservation sometimes involves complex information that relies on large information models, and which involves many different objects and relationships. To preserve such information, it is necessary to precise exactly what information needs to be preserve, and what information model can support it. Moreover, metadata for preservation can not only refer to a document, but also to particular objects inside the document. This thesis takes as an example the preservation of one category of complex information: product models.

The following sections will present the major problems related to archival system description, and information representation.

1.2/ PROBLEMS RELATED TO ARCHIVAL SYSTEM DESCRIPTIONS

Archival systems are necessary to preserve digital information. Archival system descriptions need to show that an archival system meets the preservation needs, and facilitates the preservation and the accessibility of information. The description can be the representation of an existing archival system, or it can be the design of a future archival system.

Some of the major aspects an archival system description needs to show are the interaction between the system and people or other systems, the information managed by the system, and the functions performed by the system.

A first problem is that the interactions involving the archival system may be very complex. The archival system may have to support many different content submissions, and many different access scenarios. Each of these interactions involves different actors, information, and procedures. This problem can occur in large organizations, in which many

different contents have to be preserved. Multiple submission, preservation, and access activities may exist, even for the same content.

A second problem concerns the preserved information. The archival system may have to deal with many different kinds of content, and the archival system description has to determine what content is preserved. This content can be complex to define, especially when large information models have to be created to support this content.

For each content, it is also critical to define its adequate metadata. Metadata is data about the digital content, and it is used for different reasons. A first reason is to guarantee the authenticity and integrity of the information. A second reason is to provide information that helps interpreting the data. A third reason is to facilitate the discovery of information within the collection. For example, a search on books often uses metadata such as titles, authors, editors, or ISBNs. These attributes helps in differentiating — or categorizing — the different contents. As the amount and complexity of digital content increases, metadata has to be even more well defined to enable fast discovery. Metadata is generally managed separately from the content. Because of the large amount of content of different types, metadata can also be complex to define.

A last problem concerns how the archival system functions: what are the components, what actions it performs, how it communicates with its environment. Archival systems need to support various different functions to receive, manage, and send both preserved content and metadata for this content.

1.3/ Problems related to information representation

Another aspect of digital preservation is to decide what is the best suited representation of the information. The representation can greatly influence the ability to discover and understand the preserved data over time. Sometimes a good solution from a short-term perspective is not a good solution from a long-term perspective, as the short-term requirements are different.

From a short-term perspective, the main consideration is whether the current software can interpret the data. The software environment and the knowledge base required to understand the content are considered invariant.

From a long-term perspective, it is necessary to consider that the software environment and the knowledge base might change. The file format has to give guarantees that the content will be supported by future software or, if not, that the content can be easily processed and transformed. Another consideration is how well the content can be enriched with additional information to improve its understanding and accessibility.

Many widely used formats do not support these long-term considerations. For example, the documentation for these formats is not always publicly available or affordable. Many formats do not allow users to directly complement the data with external knowledge, such as concepts not covered by the original formats, or similar objects defined in other contents. Such features can make the preserved content more understandable by adding more semantics, and can enable semantic queries to be performed over the content.

1.4/ STRUCTURE OF THIS DOCUMENT

The previous sections presented different problems related to digital preservation. A first main problem is to find a way to describe archival systems and their environment, to make sure information is preserved properly and remains accessible. A second main problem is to find a representation that supports strong semantics, to improve the understanding and accessibility of the preserved information.

These problems are evident for product models. Chapter 2 introduces product models, and explains why archival system description is complex for product models, and why product models representation can be improved with regards to preservation. Product models will be taken as an example of information for the archival system description problems, and for the information representation problems. This chapter also presents works that can be leveraged for improving the representation of product models. Finally, this chapter presents the representations currently used and evaluations of what is needed for product model preservation.

1.4.1/ A REFERENCE ARCHITECTURE FOR ARCHIVAL SYSTEMS (RAAS)

To address the archival system description problems, this thesis proposes a method to formally describe the architecture of archival systems. This method is referred to as a RAAS. An architecture identifies the elements that constitute an organization, including the people, the systems, and the activities.

Chapter 3 presents different works that can be leveraged for describing archival systems. This chapter presents works that provide the elements that compose archival systems, and works that provide a way to describe systems.

RAAS is presented in Chapter 4. This chapter proposes the representation of concepts specific to archival systems along with generic concepts used for describing systems. RAAS focuses on the high-level software design common to any preservation solution, and it does not cover detailed software or hardware implementations. RAAS proposes a generic way to describe archival systems.

In Chapter 5, RAAS is demonstrated in specific case, for the preservation of product models. This chapter presents various aspects of a product model archival system, including the different activities and information defined to allow understanding and accessibility of product models. The archival terminology is used in the description to better understand the system from a preservation perspective.

1.4.2/ SEMANTIC REPRESENTATION OF PRODUCT MODELS

To address the product model representation problems, Chapter 6 proposes a new representation for product models. This new representation is based on the translation of existing product models into a knowledge representation language. This new representation aims to improve the preservation and the accessibility of product models by allowing semantic relationships to be added to existing models. This chapter shows how this new representation can benefit the product models presented in Chapter 5.

1.4.3/ FUTURE WORK

Finally, Chapter 7 presents the conclusions on this work. It will discuss how RAAS can be completed, and how the product model representations can be more leveraged.

The approach and the contributions presented in this thesis can be leveraged in different ways. RAAS allows the use of the preservation terminology to describe information systems. The reference architecture can be extended to describe more aspects of the preservation, such as the entire organizational unit in charge of the preservation, the security aspects, or the risk management. The idea of bringing concepts from a domain (preservation in this case) into enterprise architecture can be used in other situations, and help to bridge the gap between specific business problems and implementation of a solution.

RAAS can be used with various types of content besides product models, for example, medical records.

2

INTRODUCTION TO PRODUCT MODEL PRESERVATION

This chapter explains why archival system descriptions and information representations are complex for product model preservation. This chapter also presents the current representations of product models, as well as alternative representations for better addressing product model preservation.

According to the Standard for the Exchange of Product model data (STEP), a product is a thing or substance produced by a natural or artificial process [4]. The notion of "product" can refer to pens as well as it can refer to software. A product model is an abstract representation of a product. In general, product models refer particularly to the representation of electro-mechanical products. In this thesis, product models refer more specifically to formal, computer-interpretable representations of products.

Product models are of particular interest for many organizations. Some organizations want to preserve product models to reuse existing knowledge. Also, some organizations have the legal obligation to preserve their product models, to demonstrate the compliance of their product with the regulation. For example, in the aerospace domain, airplane manufacturers have to make the airplane designs available for as long as the products are in service [5]. For this reason, manufacturers in the aerospace industry have been involved in developing recommendations to address the preservation of product models[6].

Section 2.1 presents the problems related to product model archival systems. Section 2.3 presents the problems related to product models representation with regards to preservation.

2.1/ PROBLEMS RELATED TO PRODUCT MODELS ARCHIVAL SYSTEMS

A key element to demonstrate the ability to preserve product models is to describe archival systems. As explained in Chapter 1, archival systems are necessary to preserve digital information. An archival system description determines the activities supported by the system, the information processed by the system, and the functions of the system. The archival system description can be used to describe an existing archival system, or to drive the development of a new archival system.

Product model archival systems are particularly complex to describe. As the next paragraphs will explain, these archival systems may have to support various activities and

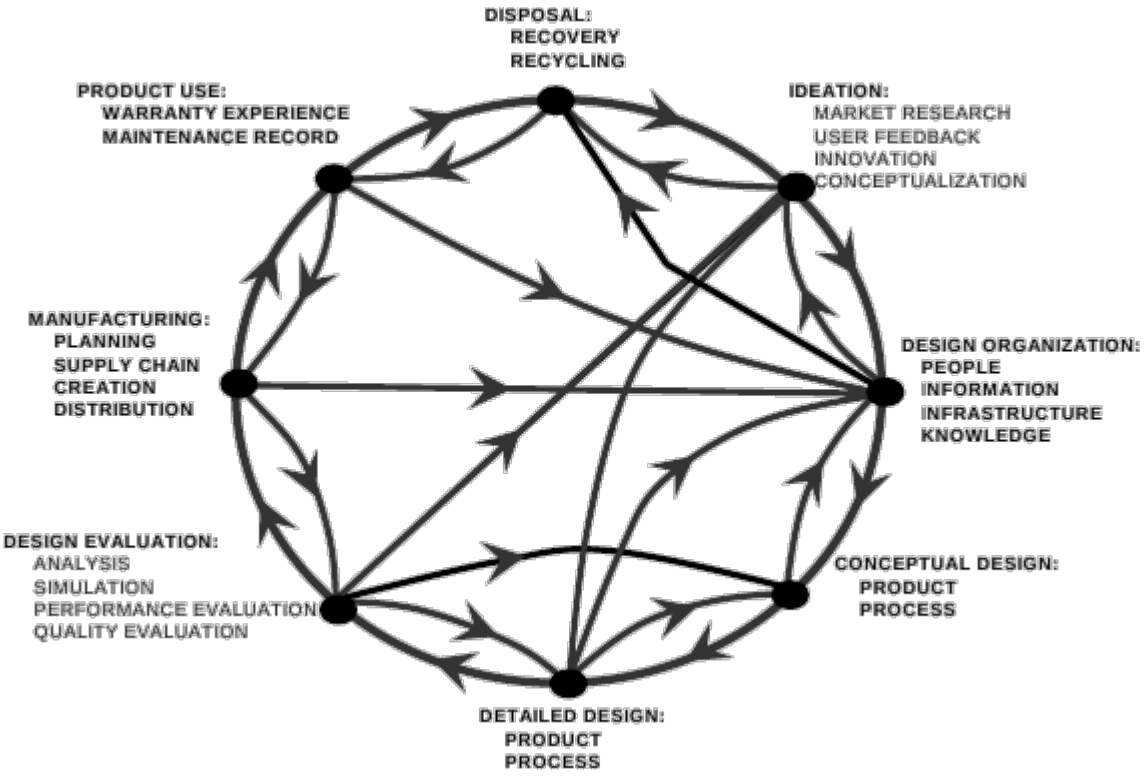

Figure 2.1: Overview of the product lifecycle

various types of content, and the metadata for this content is not simple to define.

In a business environment, an archival system typically interacts with business functions. In the case of product models, these business functions exchange product models with the archival system. Figure 2.1 shows the business context of product models, taken from a paper on Product Lifecycle Management (PLM) [7]. This Figure shows the various exchanges of product information between different product lifecycle stages: ideation, design organization, conceptual design, detailed design, design evaluation, manufacturing, use, and disposal. The different stages are placed in a circle, and the arrows represent information exchange that may occur between the different stages. As products may have a very long lifetime, there can be a long time between the creation of product information and its consumption.

In a long-term perspective, the archival system must act as an intermediate system to enable these exchange to occur over time. So, at each lifecycle stage, the archival system may have to support many different submission activities, and many different consumption activities. These activities can concern different types of product information. Moreover, even at one particular lifecycle stage, product information may be located in different systems, and sometimes in different organizations (e.g. collaborative design). This product information coming from different sources also has to be integrated for a better management and access. This complexity has to be addressed to create an effective preservation solution.

One way to categorize product information access is to consider three levels [8]. The first level is called Reference. This level is simply about rendering product information (e.g. accessing product design and tests to prove compliance, accessing maintenance infor-

mation and product design for failure analysis). A second level is called Reuse. This level is about reusing the product information to modify it (e.g. accessing product requirements and specifications to redesign a product, accessing product design to generate NC code). A third level is called Rational, and it contains all the information that led to the creation of the product. Each of these levels requires more detailed product information than the previous.

An archival system description should not only represent these access cases, but also the submission and preservation activities that make the long-term access possible. The access cases actually determine what content should be preserved, and what metadata should be added.

The definition of metadata for product models is challenging, especially to effectively organize these product models. One single product may be composed of many parts, and each part may have different versions, each version may have different viewpoints (e.g. engineering, business, marketing), and each viewpoint may have associated product information.

Different solutions have been developed to organize product information. During the design stage of the product, product information is managed by Product Data Management (PDM) systems. PLM aims to manage product information from the entire product lifecycle. PDM and PLM provide a core model to which product information can relate. For example, if a part is described in two different product models, they should both relate to the same element. The concepts reused among various product models (such as product definitions, product relationships, configuration management, activities) are included in PDM and PLM information models, and are sometimes added within the product models as metadata [9]. The definition of PLM information models is a complex task. Organizations have different ways to manage their product information, and product information is managed by different information models at different lifecycle stages. For example, the integration between the engineering product models and the business product models is problematic [10].

2.2/ PROBLEMS RELATED TO PRODUCT MODEL REPRESENTATIONS FOR PRESERVATION

The notion of model implies a particular formalism. Before the use of computers, product models such as geometrical representations were stored on paper or microfilms. Formalism was then defined in terms of human interpretation. The emergence of computers has allowed product models to be represented as digital content. Formalism in the context of digital content is defined in terms of computer interpretation. Product models refer to product information represented in a computer-interpretable way. For instance, a scanned engineering drawing is a product model for humans, but it is considered as an image for computers. The content can be interpreted only by humans as it does not contain specific product knowledge that a computer can recognize. Some product models have a more formal representation of the product, in which the product knowledge is structured. For example, a Computer-Aided Design (CAD) file contains the formal definition of the product structure (parts and relationships among parts) and its shape. These product models can be interpreted not only by humans, but also by computers. These representations are converted into visual representations that human can understand. Nowadays, a large

portion of product models are created in a digital form [11].

As computer-interpretable representations, product models rely on product information models. These product information models define the concepts and relationships that will describe the product. Different product information models have been developed, most of the time to describe the product under different perspectives, using different concepts. For example, the Standard for the Exchange of Product model data has defined over thirty different product information models (called Application Protocol) to represent different aspects of products. While many product information models focus on representing geometry information, various efforts have developed models that cover other types of product information. The Core Product Model (CPM)[12] and the Open Assembly Model (OAM)[13] include beyond-geometry information, such as function and behavior. Other efforts include the Function-Behavior-Structure (FBS)[14] model and the Function-Behavior-Structure Processes-Products-Resources-External effects (FBS-PPRE)[15] model.

Product models are actually represented according to a format, which follows the product information model. Multiple formats may also exist to represent the same aspects of a product. Because formats have different characteristics, some of them are more relevant than others to represent product models from a preservation perspective. An important problem is to determine what product model representation should be selected to guarantee that the product data can be understood and accessible over time.

A large number of product models is represented using proprietary formats. This is because product models are generally created using proprietary software, which supports native formats. A first issue with proprietary formats is the obligation to buy specific software to interpret the data correctly. Because documentation is not necessarily available or affordable, it is complicated for software companies or even for preservers to develop software that support those formats. This restriction can prevent the users from interpreting the product models, and prevent the development of tools to either extract information from the product models or to migrate the product model into another format. A second issue with proprietary formats is their frequent change. Backward compatibility of the software is not always guaranteed, and hence product models may require frequent migrations to stay interpretable. Emulation is a possibility that consists in preserving the software environment over time, so that the data can be interpreted as it was initially [16]. Although this solution works for a consumer to access the information, the data will not be reusable in the current environment, which is sometimes necessary. Also emulation can be an expensive approach.

To overcome the problems of proprietary formats, standard formats have been developed. Standard formats are managed by a community of experts that includes software companies and consumers of the format. Standard formats are generally accessible to anyone, which facilitates their implementation. They also tend to not change as often, as the agreement of a community of experts is needed. Standard formats are widely recognized as a better way to store product models from a preservation perspective. However, standard formats are not always fully supported by software vendors who prefer their own proprietary formats. Standards also lag behind proprietary formats in terms of new features.

One issue with product model formats is that they do not totally leverage the possibilities of computer interpretation. The computer interpretation is thought in terms of software support, but not in terms of knowledge representation. The formats do not support di-

rectly enriching the information with external knowledge, such as domain-specific vocabularies or other product models. A knowledge representation approach allows product information to be more semantically defined in product models. This approach can, for example, support domain-specific concepts that are not supported by generic product information models. This approach can also integrate product models by establishing semantic relationships among them. Knowledge representation is beneficial for several reasons. It leverages the computer interpretation by enabling semantic queries to make product information more accessible and understandable. These semantic queries can use domain-specific knowledge and the integrated view of product information. As a result, knowledge representation offers a good alternative or complement to the current product models representations.

Product model representations are further studied in the following section.

2.3/ CURRENT PRODUCT MODEL REPRESENTATIONS FOR PRESERVATION

This section presents works by others on representation methods for product models. Product model representations have to be evaluated according to specific criteria to determine how well suited they are for preservation. The main objectives for these representations are how well they capture the knowledge, how well they facilitate the understanding and discovery of the information, and how well they have chance to stay interpretable over time.

Lubell et al. [8] have defined metrics for engineering information representation. These metrics include language, processable expressiveness, content and interface. The authors also recalls 7 sustainability factors defined by the Library of Congress:

- Disclosure: availability of documentation and tools
- Adoption: widespread use
- Transparency: human readability
- Self-documentation: presence of metadata
- External dependencies: reliance on specific hardware/software
- Impact of patents: necessity of license to interpret the data
- Technical protection mechanism: limitations that would impair the ability to migrate the content

This section presents STEP[4], a standard for representing and exchanging product models. Then, it discusses various additional efforts to improve the representation of product models in the context of preservation.

2.3.1/ STANDARDS FOR PRODUCT MODELS

Product models are often exchanged among companies (or even within the same company) by different systems. Interoperability issues occur when these systems are not able to interpret the data exchanged. The origin of these issues are systems not supporting the same format. These interoperability issues are of particular concern for an archival system, since the preserved information comes from some systems, and need to be available for other systems. Moreover, the archival system itself needs to interpret the data

correctly, for information discovery purposes or for format migration purposes.

ISO 10303, also called Standard for the Exchange of Product model data[4] is an international standard for product models exchange[17]. STEP defines data schemas called Application Protocols (APs) that focus on a particular exchange context. The most successful uses of STEP are for CAD interoperability, in which the Application Protocol 203 – Configuration controlled 3D design of mechanical parts and assemblies (AP203) and the Application Protocol 214 – Core data for automotive mechanical design processes (AP214) are widely used for exchanging CAD models. Data schemas are written in EXPRESS, a language created specifically to support the needs of STEP. Two methods are defined for exchanging data: ISO 10303-21, referred to as Part21, or 10303-28. referred to as Part28. STEP is the product models format recommended by LOng Term Archiving and Retrieval (LOTAR)[1] and Verband der Automobilindustrie (VDA)[2]. STEP was selected because all the documentation necessary to implement the standard is open. STEP supports the current needs of the aerospace and automotive industry, which is the representation of 3D CAD PDM information. Moreover, the STEP formats do not evolve too often, and backward compatibility is guaranteed.

Regarding the sustainability factors defined earlier, proprietary formats generally offer low disclosure (specification not available), low transparency (binary files), low self-documentation (binary files), strong external dependencies (proprietary software), unknown impact of patents, and unknown technical protection mechanisms. The adoption is limited to the user of specific software. STEP formats offer high disclosure (open formats), high transparency (ASCII or eXtended Markup Language (XML) files), high self-documentation (various metadata available), low external dependencies (supported by multiple software), no impact of patents, and no technical protection mechanisms. The adoption however is quite limited, as most people still use native proprietary CAD formats[18].

The use of STEP for product model preservation has been investigated in different works. The following paragraphs present some of them.

Kassel and David [19] considered product model preservation for shipbuilding. They described the formats available for ship product models: vendor-specific formats, generic standard CAD formats such as STEP AP214, or shipbuilding standard formats such as the Application Protocol 215 – Ship arrangement (AP215), Application Protocol 216 – Ship moulded forms (AP216), Application Protocol 218 – Ship structures (AP218), and Application Protocol 227 – Plant spatial configuration (AP227).

Kassel and Briggs [20] explained the issues with preserving ship product model data. Although ship-specific APs have been created to support the shipbuilding specificities, no translators are commercially available. They proposed the use of more generic APs , such as AP214 for geometry and Application Protocol 239 – Product life cycle support (AP239) for non-graphical information, to represent this information in a neutral format. AP239 is particularly important because it allows product models to refer to concepts not defined in the original information model. These concepts can be more specific to the product domain being described, so product knowledge can be better represented.

Briggs et al. [21] studied the use of AP239 to link logistics and design data. This work is part of the Integrated Shipbuilding Environment (ISE) project, which aims at enabling

[1] http://www.lotar-international.org
[2] http://vda.de/en

interoperability among US shipyards. The ISE-6 projects aims at coordinating the use of AP239, S1000D (technical publications) and shipbuilding standards (AP212, 215, 216, 218, 227). As part of the ISE-6 project, they described the benefits of Product Lifecycle Support (PLCS) for ship data exchange: holistic view of the product lifecycle, data exchanges between the acquisition and logistic communities. A product model archive may have to provide the same functions.

The Integrated Shipbuilding Environment Consortium (ISEC) [22] presented the context of product models the shipbuilding area: various Integrated Data Environment are used to support design, manufacturing, and logistics. Different models are exchanged among each of these environments. It is necessary to integrate on one side acquisition product model data and on the other side lifecycle support product model data. The paper demonstrates the feasibility of using PLCS to exchange logistic and design data.

STEP provides open standard formats for many types of product models. However STEP has also a few drawbacks. Even though the information model is good enough to meet the main needs, the implementation of STEP using the EXPRESS language and ASCII files (Part21) has some limitations. First of all, these languages are almost only used by STEP. Their limited adoption may prevent the availability of tools to interpret the data in the future.

Another issue with EXPRESS and Part21 is the limitation of the languages regarding strong semantics. Strong semantics enables a better expression of the information, for example by using external ontologies to define knowledge more precisely. This capability is not directly available in STEP. The current mechanisms that enable semantics rely on dedicated tools rather than on the languages. One example is the Reference Data Library (RDL) mechanism, which consists of an Web Ontology Language (OWL) ontology that extends the information model of AP239. External tools are needed to integrate the product data and the ontology, since the integration is not part of the languages.

2.3.2/ ALTERNATIVE REPRESENTATIONS OF PRODUCT MODELS

Besides STEP-based approaches, different efforts have proposed alternative representations for product models.

Ondrejcek et al. [23] has worked on two different areas of engineering information preservation. First, they addressed the preservation of paper-based 2D drawings by extracting information from them. These drawings are scanned and then stored, but since they are just images, no semantic or computer-processable engineering information are represented within. By using Optical Character Recognition techniques, they worked on the recognition of various information written at specific locations in the drawing. Recognized information includes product number, approval, or date. They then attached this metadata using Resource Description Framework (RDF)[24]. This solution improves the discovery of some legacy engineering information, even though it is only applicable to 2D drawings.

Patel et al. [25] proposed strategies for the preservation of CAD files. Because of the closed nature of proprietary formats and the large size of CAD files, the authors proposed a an approach called Lightweight Models with Multilayered Annotations (LiMMA). This approach combines open and lightweight CAD formats with product lifecycle information. Existing lightweight formats are presented and evaluated according to model fidelity, metadata support, security features, file size, software support and openness.

Regli et al. [26] proposed a framework for geometric models. They considered geometric models as the core of product data management. Their goal was to have a representation for geometric models that will stay interpretable over time. They provided four aspects to consider for preservation: file formats, logical object encoding, object metadata and organizational workflows.

Their proposed approach to store geometric models uses three different representations: a discrete mathematical representation, a standard representation, and a native representation. They used ontologies to represent metadata, and to capture the relationships among the files as well as other annotations. Finally, to represent process and workflow information, the authors proposed the use of Process Specification Language (PSL)[27]. They used a large corpus of raw engineering data to demonstrate their approach. From this corpus, they inferred the underlying workflow used by the engineers during the inspection activity. Then they connected the subprocesses of the inspection activity with the engineering files. The assumed use case was the re-creation of the part.

Product information models such as CPM and OAM could also be considered for preservation. They are publicly documented, and an OWL representation has been proposed[28].

One approach to replace or complement STEP by leveraging stronger semantic is to determine a way to transform the STEP data into languages such as OWL. Two main works have been developed in this direction.

The Intelligent Self-describing Technical and Environmental Networks (S-TEN)[29] project is an effort funded by the European Community. One of its objectives is to “exploit the Semantic Web for scientific and engineering applications.” This project describes a bi-directional translation between EXPRESS and OWL. S-TEN focuses on translating subsets of APs. Hence in the S-TEN project no AP is covered in full. The translated part of the information models are also sometimes modified, either to take advantage of the use of OWL, or as an improvement. The S-TEN approach translates selected modules, which only covers a subset of STEP APs. This restricts the range of instance files that can be automatically translated to OWL. Further, the OWL schemas are manually modified after translation.

Zhao and Liu [30] proposed a methodology to represent EXPRESS models in OWL and Semantic Web Rule Language (SWRL), a rule-based language for OWL. Their report describes the approach in two parts. In the first part, the mapping between EXPRESS schemas and OWL and SWRL is discussed. SWRL adds more capabilities to OWL by permitting rule-based inference over the ontology. The different steps involved in the process are described. The authors also present a set of tools that can query and reason over the ontology. Zhao and Liu also translated procedural code contained in the EXPRESS schemas. The procedural code specifies algorithms that can compute derived attributes or to check the validity of data. Some aspects of the EXPRESS language are not properly supported. For instance, the translation of ordered lists in EXPRESS was not proposed. Automated tools doing the entire translation are announced, but they were not available.

In addition to the previous research works, some research has been done under the ISO with the recent AP239. AP239 provides a mechanism for enriching product data by using external classification. External classification enables users to define controlled vocabularies that are used to specialize instances of generic entities of the AP239 EXPRESS schema. The use of OWL is recommended, to define controlled vocabularies. This mech-

anism is known as Reference Data Library.

OWL as a format for product models was also proposed for the CPM and OAM[28].

To summarize this chapter, STEP is selected as the format to be used for product models preservation in industry recommendations like LOTAR and VDA. However, STEP has drawbacks such as its limited use, and the lack of strong semantics. There is a need for connecting STEP models to other models, such as those developed for beyond-geometry information. To overcome these drawbacks, this thesis relies on OWL to represent product models. The thesis proposes a way to automate the translation from any native STEP data into OWL, so that the information model that makes the strength of STEP are kept, while the drawbacks due to the use of EXPRESS and Part21 are overcome. The OWL representation is not intended to totally replace the original representation, but to complement it. Expected benefits of the OWL representation are that it facilitates the integration of product models, and that it makes it possible to semantically enrich product knowledge.

Product models preservation faces two problems: the description of archival systems is complex, and product model formats do not facilitate the understanding and the accessibility of product knowledge. This thesis aims to provide solutions for these two problems. First, the Reference Architecture for Archival Systems (RAAS) presented in Chapter 4 should be able to support the context of product model preservation: the preserved product models, the multiple activities that submit and consume product models, and the complex metadata needed to organize and support the preservation of product models. Chapter 5 intends to demonstrate this support. Second, the proposed representation of product models, presented in Chapter 6, intends to enhance the understanding and the accessibility of product models, by allowing semantic relationships to be established among product models.

3

LITERATURE REVIEW OF ARCHIVAL SYSTEM DESCRIPTION METHODS

Digital preservation issues have been addressed by different communities. The library community is a major contributor, since one of its primary goals is to manage information (books and periodicals) over a long period of time. The scientific community has also addressed the preservation issues by proposing solutions for scientific data. In the engineering world, the problem is mainly being addressed by developers and maintainers of products with long lifecycles, such as automotive and aerospace companies.

This chapter covers research related to the description of the archival system. The chapter is split into two sections. Section 3.1 addresses what needs to be represented in archival system descriptions. The presented efforts define concepts and requirements for archival system. Section 3.2 presents efforts that have targeted product model preservation. Section 3.3 addresses methods for describing systems that can be used for archival systems.

3.1/ CONCEPTUAL FRAMEWORKS AND REQUIREMENTS FOR ARCHIVAL SYSTEMS

This subsection covers two sources for archival system concepts and requirements. The Reference Model for an Open Archival Information System (OAIS RM) introduces the major concepts to be used for describing and comparing archives. The Audit and Certification of Trustworthy Digital Repositories (ACTDR) presents requirements that an archive must meet to be declared trustworthy. These requirements should be considered during the design of the archival system.

3.1.1/ CONCEPTUAL MODELS FOR ARCHIVAL SYSTEMS

The Reference Model for an Open Archival Information System (OAIS RM) (ISO 14721) [1] proposes a conceptual framework for describing and comparing archives. It defines the terminology related to information preservation. The OAIS RM presents the external actors that interact with the archive, an information model that shows what information is managed by an archive, and a functional model that shows the generic activities performed by an archive. An archive is defined as an organization (people and systems) that intends to preserve information and have it accessible.

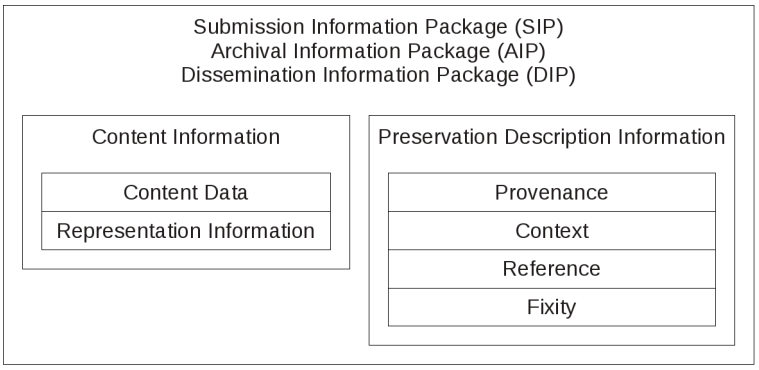

Figure 3.1: OAIS Information Packages

The OAIS RM was developed by the Consultative Committee for Space Data Systems (CCSDS), which include space agencies from many different countries. The OAIS RM is the result of a consensus among the different space agencies regarding what an archive should provide.

Three types of actors are defined in the OAIS RM. Producers send information to the archive for preservation. Consumers retrieve preserved information from the archive. Managers set the policy for the OAIS. The designated community is defined as the consumers for which the preservation is intended. The OAIS RM does not include the actors responsible for developing and maintaining the archive itself, such as archivists.

The OAIS RM proposes an information model that represents what information is stored and processed by the archive. Because the preserved content alone is not sufficient to ensure its preservation and accessibility, additional information is needed. Contents and their additional information are encapsulated into information packages (see Figure 3.1). Different kinds of information packages are defined depending on the context. The Submission Information Package (SIP) refers to the package the producer sends to the OAIS. The Archival Information Package (AIP) refers to the package the archive stores. The Dissemination Information Package (DIP) refers to the package the archive delivers to the consumer. Each information package has some packaging information. Content data refers to the raw data that represents the information to preserve. Representation Information (RI) corresponds to everything that explains how to interpret and understand the content data. Preservation Description Information (PDI) is an aggregation of four types of information:

- Context information: relationships between the content and the other contents or the environment,
- Provenance information: history and ownership of the content,
- Fixity information: information to make sure the content is not altered,
- Reference information: identification of the content.

Some of the PDI is derived from the information packages to generate Descriptive Information (DI), which will help consumers discover the preserved information.

The OAIS RM also describes the main functions performed by an archive (see Figure 3.2). Each function is further detailed into smaller functions. Ingest represents the function that receives the SIP from the producer, and generates the AIP as well as the DI. The Data Management function is in charge of managing the DI, and querying this data whenever consumers attempt to locate specific information. The Archival Storage func-

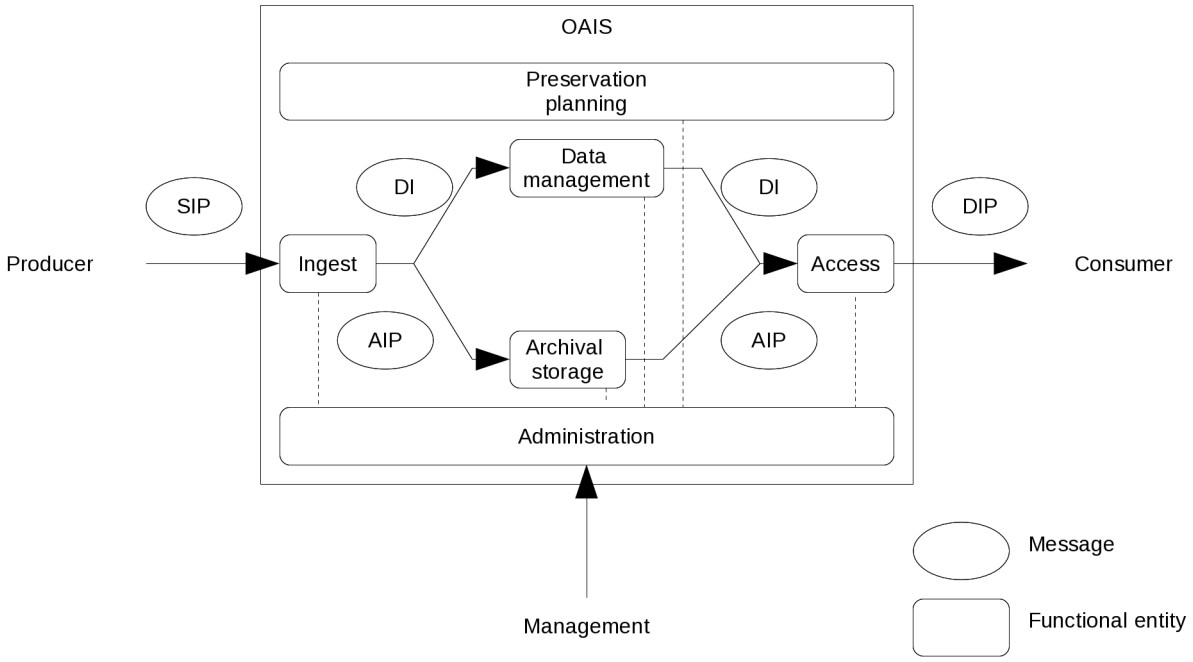

Figure 3.2: OAIS Functional Model

tion does the actual physical preservation of the AIP. Access provides services to the consumers so that they can retrieve the wanted information. The Preservation Planning function is in charge of making sure the information can be interpreted over time. The Administration function concerns the application of the overall strategy by the supervision of the other functions. The arrows among actors and OAIS functions represent information exchanges.

Finally, the OAIS RM defines responsibilities for an archive to meet in order to be considered an OAIS:

- The OAIS must negotiate for and accept appropriate information from the producer. This is achieved by creating an agreement that explains what information is expected.
- The OAIS must obtain enough control of the information provided and to the level needed to ensure long-term preservation. This includes having the legal right (e.g. intellectual property) and technological possibility (e.g. well documented format) to migrate the content to preserve the information.
- The OAIS must determine which communities should become the designated community and should understand the information provided. The OAIS must also ensure that the information to be preserved is independently understandable to the designated community. This is to determine whether the information managed by the OAIS (content plus additional information) can be interpreted by the foreseen consumers.
- The OAIS must follow documented policies and procedures that 1) ensure that the information is preserved against all reasonable contingencies, 2) ensure that the information is disseminated as authenticated copies of the original. The OAIS should have and publish migration procedures, and should monitor the designated community.

- The OAIS must make the preserved information available to the designated community.

Both information and functional models are conceptual: they are meant to be independent from any implementation technology or content domain, and they are not directly implementable. So, the OAIS RM should be seen as an aid for developing an archive. To develop an archive based on the OAIS RM, it is necessary to tailor the information and functional models according to an actual context: who are the producers and consumers? What information should be preserved? What metadata should be present ? What formats are accepted? etc.

As the OAIS RM is independent of any implementation strategy, it does not prescribe whether a function is implemented by computers or not.

The OAIS RM should be the cornerstone of the archival system design. An important aspect of the archival system design should be to refer to the terminology defined in the OAIS RM, so that it is possible to clearly see how functions and information are implemented.

The OAIS RM is not meant to be used for certification purpose. However, there was still a need for criteria to evaluate an archive.

3.1.2/ CERTIFICATION OF ARCHIVES

Audit and Certification of Trustworthy Digital Repositories (ACTDR)[31] is a recommended practice delivered by the CCSDS for the certification of an OAIS. ACTDR is based on a document called Trustworthy Repositories Audit and Certification, which was developed by the Research Libraries Group and the National Archives and Records Administration.

The certification concerns three different areas: the organizational infrastructure, the digital object management, and the infrastructure and security risk management. Each area is composed of requirements, and each requirement contains an example of how to demonstrate that the organization meets that requirement.

The organizational infrastructure part contains requirements related to the commitment of the organization to preserving information. It relates to various administrative aspects, such as governance (mission statements, strategic plans), staff, policies and financial aspects.

The digital object management part is about the management of the content. This part contains requirements for the main functions described in the OAIS RM: ingest, preservation planning, archival storage, data management, and access.

Finally, the last part is about the infrastructure and security risk management.

ACTDR focuses on the management of an OAIS in the context of an organization. Regarding the archival system description, ACTDR provides a few clues about what is expected for an archival system description to show that it can actually preserve information and make it accessible. But as in the case of the OAIS RM, specific aspects such as security and risk management are not considered for the topic of this thesis.

In addition to the recommendations produced by the CCSDS, various efforts have tried to build upon existing preservation efforts around the world, in an attempt to identify and

develop good archival practices.

3.1.3/ DEVELOPMENT METHOD FOR THE ARCHIVAL SYSTEM

Developing an archive requires many steps, from the definition of the requirements to the actual implementation. One important work that has comprehensively addressed this aspect is the International Research on Permanent Authentic Records in Electronic Systems (InterPARES) program [32].

InterPARES is a international multi-year program that aims to "develop the knowledge essential to the long-term preservation of authentic records created or managed in digital form". To do so, InterPARES builds upon the OAIS RM, ACTDR, and many other preservation efforts. This program is split into three projects.

InterPARES 1 (1999-2001) focused on the preservation of inactive (no longer needed for day-to-day business) electronic records. The idea was not to bring solution, but to clearly identify and articulate the preservation problems.

InterPARES 2 (2002-2006)[33] aimed at defining the concepts, criteria, and methods to ensure the creation and maintenance of accurate and reliable records, and the preservation of authentic records. The project looked at records from various scientific, artistic, and governmental sectors. The scope of the projects includes the generation, selection, preservation of records. Significant outputs of the project are principles and guidelines for the creators and the preservers, and also an activity model called Chain of Preservation (COP). The creators guidelines targets the individuals in charge of creating or maintaining digital materials. The preservers guidelines targets the people responsible for the long-term preservation of digital records.

The COP defines the steps in the creation, maintenance, and preservation of digital records. The COP considers four main record activities: managing the framework, managing the records creation, managing records in a recordkeeping system, and preserving the selected records. The last three activities involve respectively three different systems: the record-making system, the recordkeeping system, and the permanent preservation system. In the product model world, a record-making system wound correspond to any tool used to create product models (such as Computer-Aided Design (CAD)), a recordkeeping system would correspond to Product Data Management (PDM) systems, and a permanent preservation system would correspond to an archival system. The design of the different systems is included in the COP. The COP can be seen as a comprehensive methodology for addressing preservation within an organization, including how to develop archival systems.

The following section present preservation requirements that emanate from the product model world.

3.2/ PRODUCT MODEL SPECIFIC REQUIREMENTS

This thesis studies the preservation of product models. As a specific kind of information, product models have particular requirements regarding preservation. This section will present some requirements that have been identified in various works. Industry recommendations are a first source of requirements. However, if these recommendations

provide requirements, they are not likely to provide implementation guidance or description guidance for the archival system.

3.2.1/ PRODUCT MODEL PRESERVATION IN THE GERMAN AUTOMOTIVE INDUSTRY

Verband der Automobilindustrie (VDA) is the German automotive industry association that includes manufacturers and suppliers. VDA publishes various recommendations, but the one of particular interest to us is about Long-Term Archiving of digital Product Data (VDA-4958) [34], published in 2005. This recommendation was developed by the CAD/ Computer-Aided Manufacturing (CAM) working group. Motivations for preserving product models are to provide proof in the event of legal dispute, to show that safety regulations were applied, or simply to maintain institutional knowledge. The VDA recommendation focuses on CAD files and PDM data.

The recommendation is divided into 4 parts. The first part gives the overview, requirements and general recommendations. The requirements for the preserved documents include:

- having unique identifiers for the document
- keeping the history of the document
- identifying the authors and persons in charge
- ensuring its authenticity

The second part describes the processes related to the archival system. These processes are derived from the functions given in the OAIS RM. The few variations include the validation of the data through the concept of validation properties, and the use of digital signatures for authenticity. Validation properties are calculated using the product information, and they are independent from the format used to represent this information. For example, the center of gravity for a product geometry is a validation property. Because this property does not depend on the format, it can be used to make sure all the representations of a product geometry are identical, especially during file migration.

The third part is about the data models for long-term preservation. It specializes the OAIS RM information model for 3D CAD models and associated non-geometric information. This part addresses the identification of content to preserve at three different levels of abstraction: conceptual, logical and physical. These conceptual models include characteristics such as geometry, dimensions, tolerances, and classifications that are then mapped to standard implementation models such as Standard for the Exchange of Product model data (STEP) Application Protocol 214 – Core data for automotive mechanical design processes (AP214) for geometric information.

The fourth part is about certification of the archive workflows. The certification has ten assessment areas: general description of the organization, user-oriented description of the long-term preservation solution, technical long-term preservation solution and migration, security, technical operation, processes, employee qualifications, tests, maintenance, and integration of technical and organizational measures. The VDA certification requirements are more specific than those of ACTDR.

VDA focuses mainly on what product information to preserve for the automotive industry. It covers both geometrical and non-geometrical (e.g. configuration management) characteristics present at the design stage. VDA proposes conceptual, logical, and physical

models to represent product information. Moreover, VDA emphasizes the validation and authenticity aspects. No guidance is proposed to preserve and access other types of product models, and the recommendation does not focus on metadata in the Product Lifecycle Management (PLM) context.

Regarding the archival system design, the key aspects to learn from VDA are that information related to the document authenticity should be supported, as well as validation properties and procedures.

3.2.2/ PRODUCT MODEL PRESERVATION IN THE AEROSPACE INDUSTRY

The LOng Term Archiving and Retrieval (LOTAR) project [6] is an ongoing international effort that addresses the storage, retention and retrieval of 3D-CAD and PDM data. It is led by both the Aerospace Industry Association in the USA and the AeroSpace and Defence Industries Association of Europe. The final result of the effort will be a set of recommended practices published both in the USA and in Europe. These recommended practices are split into 4 different parts: the basic parts, the common process parts, the support process parts, and the data domain specific parts.

The *basic* parts introduce the standard by covering the main business requirements, authentication and certification of the archive, and a system architecture framework. The *basic* parts also define concepts, terms, and references used across all the LOTAR recommendations.

The *common process* parts present the main activities of submission, preservation, and access. These parts are the same as in VDA: Overview Data Flow, Data Preparation, Ingest, Archival Storage, Retrieval and Removal.

The *support process* parts cover the administrative activities related to the archive. No recommendations have been published as of 2012, but the scope will include testing, auditing, preservation planning, data management and administration.

Finally, the *data domain specific* parts provide solutions for preserving the following types of information: CAD 3D models, PDM, composite, electrical harness tubing and systems engineering. Only the CAD 3D models part has been released as of 2012. LOTAR has defined a set of requirements that must be met prior to standardization. For example, a requirement for 3D-CAD is to have a format that supports the formal representation of geometric dimensioning and tolerancing. The LOTAR PDM parts will be required to address specific use cases, such as type certification[5]. Each data domain specific part describes a set of requirements that drives the preservation, and recommended specific data formats.

3.2.3/ ACADEMIC RESEARCH ON PRODUCT MODEL ARCHIVES

Academic research tends to provide cross-domain solutions to product models preservation. Recent works have been motivated by the transition to digital representation.

A workshop on Long Term Knowledge Retention was held at National Institute of Standards and Technology (NIST) in 2006[35]. The goal was to identify challenges, research and implementation issues in the digital preservation of information, with an emphasis on design and manufacturing. The workshop recommendations included the following:

- build a registry of engineering formats to be used as common source or representation information for engineering archives
- determine how best to capture workflows of business and manufacturing process
- collect and preserve case studies so that they are available to inform the design of engineering archives

A second workshop on Long Term Knowledge Retention was held in Bath in 2007[36]. The main conclusions of this workshop were the necessity of a CAD data preservation demonstration, the need for file migration tools.

Lubell et al. [8] identified the following requirements for long-term preservation of engineering informatics:

- representation methods for product and process information
- strategy for predicting access requirements over the long-term for digital objects
- suitability criteria and metrics for engineering formats
- registry for classifying engineering information
- and extension of the OAIS RM

From the user access perspective, Lubell et al. identified three different levels of access: reference, reuse and rationale. Reference is the ability to read the data. Reuse is the ability to modify and re-engineer the information. Rationale is knowing the intent. Each successive level of access requires a higher level of semantics. For example, a format meant for visualization may not contain information accurate enough to manufacture a product.

Lubell [37] presented the requirements for metadata needed to preserve product data. Metadata is crucial for ensuring a correct preservation and access over time. This paper explains the use of metadata defined in the OAIS in the context of product models.

Lubell et al. [38] identified some issues in engineering informatics preservation: variety of data types, data accuracy, short product model formats life. The authors propose the generation of descriptive metadata during the ingest of ship product model. The description of the ingest activity and descriptive information should be present in the archival system design.

Brunsmann and Wilkes [39] studied the incorporation of long-term preservation ideas into PLM. This included service and operation knowledge, design rational, innovation process, design history, design constraints, design reviews, and product classifications. Knowledge and representation languages evolve over time. Information needs to be viewed, reused, migrated, and available for searching.

Design knowledge needs to be migrated to overcome semantic and syntactic evolution. Brunsmann and Wilkes propose an architecture for the information models of a PLM system. The PLM data can be composed of multiple information models. PLM knowledge is attached to this PLM data, and it may refer to external ontologies such as Part Library (PLib)[40] or format registries. Their approach is to either annotate PLM data, or to establish relationships between internal and external concepts.

Because current PLM repositories do not offer preservation functionalities, Wilkes et al. [41] also proposed a long-term preservation architecture for managing PLM information (such as requirement, analysis, rational, operation). Their focus was on long-term access of design data for reuse, and on long-term access to representation information.

Wilkes et al. defined some requirements for digital preservation systems in PLM:

- the preservation functions should be integrated with the engineering function
- it should be possible to access the archival system through services
- the preservation system should be independent from the PLM system
- the system should be able to transform native files to a neutral format, and preserve both
- the system should store metadata
- the system should validate the data
- the system should enforce intellectual property rights and restrict the access to data when necessary
- the system should leverage metadata to enable search and retrieval
- the system should support migration
- the system should be able to refer to external taxonomies
- the system should be able to establish relationships with external objects (e.g. in Enterprise Resource Planning (ERP) systems)

The authors proposed an architecture that relies on the communication between a PLM system and the archival system. Many of the requirements they have defined in this architecture are useful for any kind of archival system, and will be reused in this thesis.

This section provided various requirements for archival system descriptions. To summarize, the following aspects should be present in these descriptions:

- reference to the OAIS concepts
- identification of the certification criteria addressed
- identification of the standards or other information models required to understand the data
- definition of the content and the metadata (such as descriptive information)
- description of the archival system functions
- description of the ingest and access activities
- communication mechanisms of the archival system

The next section presents system description methods that can be used to describe archival systems.

3.3/ ARCHIVAL SYSTEM DESCRIPTIONS USING ENTERPRISE ARCHITECTURE

This section focuses on the method chosen to describe the archival system. The objective is to select a method that can be used to describe archival systems according to the concepts defined in the OAIS RM, and to provide certification evidence for ACTDR. This part introduces the Enterprise Architecture (EA), which addresses the description of systems within an organization. Then, different Enterprise Architecture Frameworks (EAFs) that are used to develop systems architecture are presented. A particular look is taken on Department of Defense Architecture Framework (DoDAF), an EAF that emphasizes the formal description of systems.

EA provides the description of different components of an enterprise (business organization) and their interconnections [42]. EA intends to deal with the complexity of the enterprise and to show how high-level goals are implemented. EA was initially developed to help manage different information systems within a large organization. One major ben-

efit of EA is to facilitate the change of information systems. The description of a particular system is called an architectural description [43].

EA is well suited for describing an archival system that interacts with many other components of the enterprise. As explained in Chapter 2, this is the case for enterprises involved in any activity related to complex products. The archival system has to accept product models from various heterogeneous sources, and it may disseminate these models to many different groups of people (designers, manufacturers, maintainers...) that could be within the organization, or outside of the organization (suppliers and customers).

3.3.1/ ENTERPRISE ARCHITECTURE FRAMEWORKS

The actual description of an architecture is addressed by EAFs. EAFs provide generic tools for developing architectural descriptions. These frameworks provide descriptions at different levels of abstraction: from a high level — meant for decision makers — to a low level — meant for software architects and developers. Different EAFs exhibit different characteristics and approaches [44, 45].

Because the structure of an enterprise is complex, EAFs decompose this structure into different pieces. One example of decomposition approach is the Zachman Framework [46], which uses a matrix of 6 types of abstractions (data, function, network, people, time, motivation) by 5 levels of abstraction (contextual, conceptual, logical, physical and out-of-context). This decomposition allows stakeholders to quickly get the information they want. Other EAFs do not use the same decomposition, but rather provide different viewpoints for different stakeholders.

Some EAFs provide a detailed methodology for developing the enterprise architecture. This is especially the case for The Open Group Architecture Framework (TOGAF)[47], which is known for its ADM (Architecture Development Method). Methodologies are useful for transitioning from a current state (“As-Is”) to a desired state (“To-Be”).

Some EAFs provide a language and a set of views to formally describe an enterprise. This formal description offers three benefits: providing good semantics to architectural elements, seeing how a particular element is used under different perspectives, and facilitating the exchange and the modification of the architecture. DoDAF[48] and the similar Ministry of Defence Architecture Framework (MODAF)[49] define a metamodel that enables formal architectural description. Modeling languages based on these metamodels have been developed[50], for instance by extending Unified Modeling Language (UML)[51] or Systems Modeling Language (SysML)[52] so that they can describe an architecture. UML and SysML do not actually provide a full range of enterprise-specific concepts, but they are well suited to represent software systems.

Other EAF include the Generalized Enterprise Reference Architecture and Methodology (GERAM), developed by the IFIP-IFAC Task Force [53, 54, 55], and adopted as an Appendix of ISO15704:2000 [56] is a generalized EAF for enterprise integration and business process engineering. GERAM defines all the components required for use in enterprise engineering. Other well-known reference architectures are the Purdue Enterprise Reference Architecture (PERA)[57], CIMOSA[58], and GRAI-GIM[59].

The different EAFs currently available are converging on a common set of best practices. The current version of TOGAF (9.1) has a metamodel, and the current version of DoDAF (2.0) includes a methodology. This evolution reduces the fundamental differences among

the EAFs.

Becker et al. [60] have proposed an approach to develop preservation solutions using EA. Their approach is to study the application of an existing development methodology, the Architecture Development Method of TOGAF, in the context of digital preservation. So, their approach does not address the representation of preservation concepts, but rather identifies standards and recommendations that should drive the development of the archival system.

3.3.2/ The DoD Architecture Framework

DoDAF is an EAF that focuses on giving tools to describe of architectures. Other EAFs have been derived from DoDAF, for example the Ministry of Defence Architecture Framework (MODAF)[49] and the NATO Architecture Framework (NAF)[61]. DoDAF was created to meet the needs of US Department of Defense, but it is still generic enough to be applied in any organization.

DoDAF is composed of two main parts. First, a metamodel that defines the architectural elements necessary to describe an architecture. Second, a set of views that address different concerns. A view covers a particular subset of the metamodel. The views are organized in viewpoints. Figure 3.3, which is taken from the DoDAF specification, shows these viewpoints:

- All Viewpoint (AV) describes the “overarching” aspects of an architectural description. In other words, this viewpoint describes all what the models of the architectural description have in common.
- Capability Viewpoint (CV) describes the capability requirements, definitions, and deployment.
- Operational Viewpoint (OV) describes the activities required to conduct operations.
- Services Viewpoint (SvcV) describes the services provided to support the activities.
- Systems Viewpoint (SV) describes the systems and their interconnections.
- Data and Information Viewpoint (DIV) describes the data used during the activities, by the systems or services.
- Standards Viewpoint (StdV) describes the policy, standards, guidance, constraints and forecasts that apply to the activities, systems or services.
- Project Viewpoint (PV) defines the projects and how they relate to the capabilities.

The metamodel of DoDAF 2, called DoDAF Meta-Model (DM2), has 3 layers: the Conceptual Data Model (CDM), the Logical Data Model (LDM) and the Physical Exchange Specification (PES). The CDM is composed of high-level non-technical concepts such as Activities, Capabilities, Data, Information, Persons, Resource, and Service. The LDM provides a more technical and logical representation. DM2 is presented as a core that can be extended by user communities. The concepts defined in DM2 are then used across all models, allowing the same object to be represented according to different perspectives.

DM2 was implemented as an extension of UML/acsysml, called Unified Profile for DoDAF/MODAF (UPDM). UPDM provides a vocabulary for modeling architectural descriptions that conform to DoDAF. UPDM models can be created using all the major UML tools. The reliance on UML provides a way to use an Model-Driven Architecture (MDA) approach to software development, while the reliance on SysML provides a way to represent requirements.

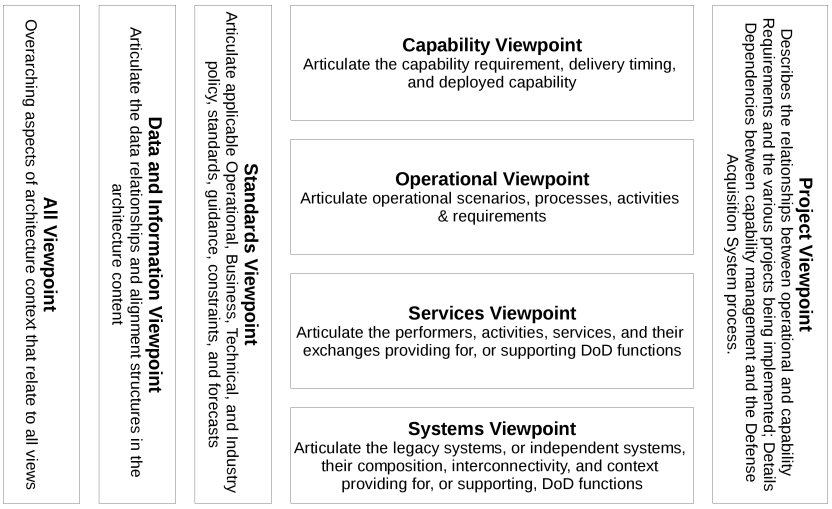

Figure 3.3: DoDAF Viewpoints, as in the DoDAF 2 specification

This chapter described the different efforts that can be leveraged to describe archival systems. The OAIS RM has conceptual and generic information and functional models that can guide the implementation of archival systems. The ACTDR points out important aspects of the archive that should be validated to demonstrate the archive's trustworthiness, and that should be part of archival system designs. The other efforts provide additional requirements and guidance for archival system descriptions.

This thesis will address the formal description of archival systems using EA in Chapter 4. This formal description will rely on DoDAF to provide a basis for describing architectures. The DoDAF terminology will be specialized according to the archival terms defined in the OAIS RM. The ACTDR will guide a selection of views to demonstrate that the archival system has the necessary concepts to preserve information. Even though the archival system description does not target a particular content, the specific requirements of product models will be taken into consideration. The result of this work is a reference architecture to guide and constrain the description of an archival system within an organization. Chapter 5 will then present an application of the reference architecture for product models preservation.

4

A REFERENCE ARCHITECTURE FOR ARCHIVAL SYSTEMS (RAAS)

Chapter 3 identified the need for describing archival systems and their environment. This environment includes different elements of an organization, such as persons, other systems, activities, or information that have any kind of relation with the archive. Enterprise Architecture (EA) addresses the description of these aspects. So, the EA approach makes it possible to represent the preservation solution within a complex environment, where multiple systems have to communicate with the archival system, and multiple activities have to be supported.

This chapter presents a Reference Architecture for Archival Systems (RAAS). A reference architecture aims to "guide and constrain the instantiations of solution architectures" [62], which in this case corresponds to archival systems. It is important to note that RAAS covers the description of the archival system, and it does not constitute a development method for the archival system. The term "architectural description" refers to the description of the structure of an actual system or organization. An architectural description is generally composed of views, which are expressed using a particular language defined in a metamodel. RAAS proposes an extension of this metamodel to support the preservation concepts. Then, RAAS proposes a set of views that describe important aspects of the archival system.

RAAS uses the core metamodel and views defined in Department of Defense Architecture Framework (DoDAF) to provide the necessary tools to describe systems. Section 4.1 presents the goal, scope, and organization RAAS. Section 4.2 presents the archival terminology that will be used to describe archival systems. Section 4.3 presents a set of views that can be used to describe important aspects of an archive. RAAS will then be used in Chapter 5 for describing the architecture of a product model archive.

4.1/ INTRODUCTION OF RAAS

The purpose of RAAS is to guide and constrain the description of archival system architecture. This description includes both the design of the information system, and the representation of its business environment. The scope does not include administrative or management activities. Moreover the description focuses on the preservation domain specifically, rather than other domains such as security and risk management. RAAS aims at being generic enough to be used for any archival solution.

Section 4.1.1 presents how RAAS is meant to be used. Section 4.1.2 presents how

RAAS relates to preservation recommendations. Section 4.1.3 explains the organization of RAAS.

4.1.1/ USE OF RAAS

The main objective of RAAS is to support the formal description of archival systems designs. To do so, RAAS provides a specific terminology for archival systems, as well as a set of views to describe important aspects of the archival system. RAAS can be used either to design a future archival system, or to represent an existing one. The archival terminology makes it possible to keep track of the preservation concept during the system design stage. Also, this approach makes it easier to understand how the preservation strategy is actually implemented. For example, it becomes easy to find out all the ingest activities, as all these activities will be related to the same concept.

RAAS supports aspects that are common to any solution: interactions involving archival systems, the composition of archival systems, and information processed by the archival system. Depending on the context, archival system descriptions may include additional aspects that are not covered by RAAS. For example, an archival system description can provide information regarding:

- the security management
- the organizational structure in charge of the archive
- the monitoring activities
- implementation details (more detailed models)

RAAS is meant to be as generic as possible, which means the defined concept and views are assumed to be present in any archival system. RAAS can be extended or modified to support more specific contexts. For example, a company interested in homogenizing its archival system can provide more company-specific details on how all its archival systems should be described. Another example would be to include aspects that are currently not in the scope of RAAS. The previous paragraph listed some aspects that could be included in an extended reference architecture. Section 4.1.2 lists a few Open Archival Information System (OAIS) functions that are not in the scope of the thesis.

4.1.2/ HOW RAAS RELATES TO PRESERVATION RECOMMENDATIONS

RAAS relies mainly on two recommendations: the Reference Model for an Open Archival Information System (OAIS RM) and the Audit and Certification of Trustworthy Digital Repositories (ACTDR).

The OAIS RM provides concepts for describing archives. However, OAIS implementations have no way to relate the system design back to the OAIS RM. This standard terminology is the main source of inspiration for the archival terminology defined in RAAS. An enterprise architecture terminology defines generic concepts to describe system. The goal of RAAS is to add archival-specific terminology to better define archival system architectures, using a common terminology. This archival terminology as added as specialization of the generic architectural terminology. RAAS relates to the OAIS RM in the same way as the Service-Oriented Architecture (SOA) reference architecture relates to the SOA reference model[63].

Figure 4.1 summarizes the relationships between the OAIS functions and information and RAAS. More information about the OAIS functions is available in Section A.1. This Figure is further explained in the following paragraphs.

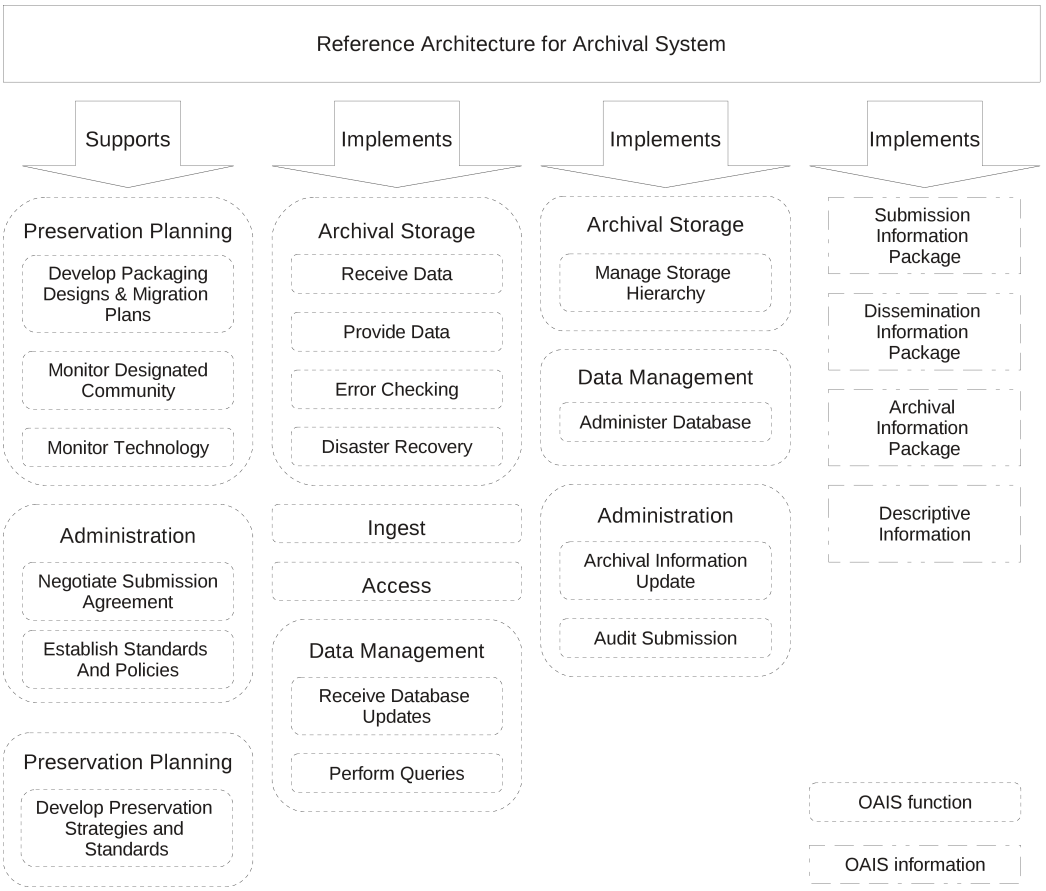

Figure 4.1: Relation between OAIS functions and RAAS

The main parts of the OAIS RM are a functional model, and an information model.

The functional model contains different types of functions. Some functions describe how the archive processes the information, other functions describe how the archive is developed and managed. While RAAS focuses on the computer system, the OAIS RM does not state what functions are performed by computers, and what functions are performed by humans. The relationships between the OAIS RM functions and architectural descriptions resulting from RAAS are explained below.

The boxes on the very left depict OAIS functions related to the development or the maintenance of the archive: Develop Packaging Designs and Migration Plans, Monitor Designated Community, Monitor Technology, Negotiate Submission Agreement, Establish Standards and Policies, Develop Preservation Strategies and Standards. RAAS can be used to support these functions by describing the design of the archival system. Here is a list of documents that are produced by these functions and that are supported by RAAS:

- Submission agreement definition, including identification of the producers, composition of the Submission Information Package (SIP) (content Representation Information (RI), and Preservation Description Information (PDI)), standards used (doc-

umentation standards and format standards), information rights
- identification of consumers and their competences
- Information packages definition
- Standards used

The boxes in the center-left side present OAIS functions that process information: Receive Data, Provide Data, Error Checking, Disaster Recovery, Ingest, Access, Receive Database Update, and Perform Queries. These functions are in the scope of RAAS. Unlike the OAIS RM, RAAS makes assumptions about functions performed by computers. This is necessary because an architectural description has to differentiate what is implemented as software, and what is not.

The boxes in the center-right side present OAIS functions that consist of human activities that involve interactions with the archival system. These functions are to make sure the content stays understandable and accessible over time. The following functions are concerned: Manage Storage Hierarchy, Administrate Database, Archival Information Update, and Audit Submission.

The other functions were not covered for various reasons: some are not about the archival systems interactions (Disaster Recovery, Physical Access Control), some are too specific (Customer Service, Manage System Configuration, Generate Report, Activate Requests). These aspects could be addressed in a future work that would complement RAAS.

Finally, the boxes on the very right side represent the information packages and their content: Submission Information Package, Archival Information Package, Dissemination Information Package, and Descriptive Information. RAAS supports representing the implementation of these models.

ACTDR is the second source of inspiration for RAAS. ACTDR provides a set of criteria for a repository to be declared trustworthy with regards to long-term preservation. ACTDR has a more organization-oriented point of view. For example, the organizational context is taken into consideration (e.g. organizational infrastructure requirements).

ACTDR covers many different aspect of the archive, from the management to the implementation. RAAS covers some of these aspects by proposing architectural views that can serve as evidence for ACTDR certification. The set of views presented in Section 4.3 states which ACTDR criteria can be addressed by each view. Some aspects of the ACTDR are not covered by RAAS, for example the organizational structure of the OAIS, the financial aspect, or the risk management aspect. The detailed list of criteria addressed by this thesis is available in Section A.2.

To summarize, Figure 4.2 shows an overview of the approach. RAAS is composed of two parts. First, it has an extension of the DoDAF metamodel to support the archival vocabulary defined in the OAIS RM. DoDAF provides a generic core vocabulary, to which archival terminology are added. Some of these concepts are provided by the OAIS RM. Second, it has a selection of DoDAF views to describe various aspects of the archival system. This selection presents what aspects of the archival system should be represented, and how some of them can serve as evidence for ACTDR certification. RAAS can guide archival system developers to develop architectural descriptions of archival systems.

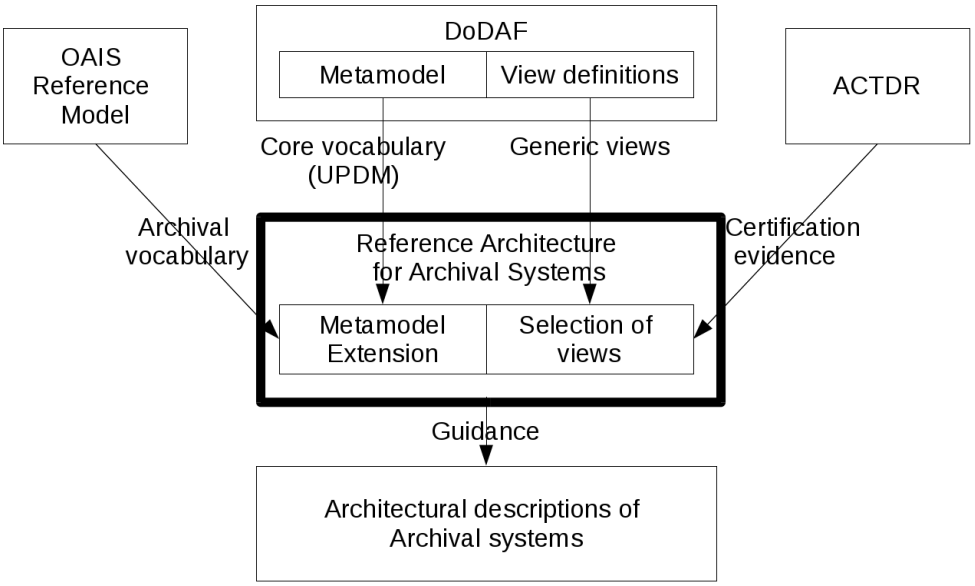

Figure 4.2: Overall presentation of the approach

4.1.3/ ORGANIZATION OF RAAS

RAAS is composed of two parts. The first part presents the archival terminology added to the DoDAF metamodel to include archival-specific terms. The second part presents a selection of DoDAF views to show how these terms can be used within an architectural description.

To be more precise, Unified Profile for DoDAF/MODAF (UPDM)[50] is used as implementation of the DoDAF metamodel. UPDM is developed as a Unified Modeling Language (UML) profile. The presentation of the archival terminology is organized as follows: a first part presents concepts taken from the OAIS RM, and a second part presents additional concepts not present in the OAIS RM but that would still be present in any archival system description. The extension is presented in Section 4.2.

The selection of views includes an abstract realization of the archive's architectural description. The views are divided into four parts that each serves a different purpose: overview, ingest and access, preservation, and systems and services (see Figure 4.3). These parts are presented in the following paragraphs.

The Overview part provides a summary of the archive. This part is meant for those who want to have a quick overview of the archive. It does not need to give a detailed description of the activities or implementation, but it should contain the scope of the archive, the motivation for preserving information, and mission and goal of both the archive and the enterprise.

The Ingest and Access part focuses on the ingest and access activities. This part is meant for the preservers to ensure consistency between what consumers expect from the archive and what is provided by the producers. The reason for grouping both ingest and access activities is to facilitate the comparison between what content is expected in input and output. The same thing applies to the metadata. For example, the descriptive information used in the access scenarios to locate the content has to be sent to the archive during the ingest. The ingest activity starts when the producer selects a particular

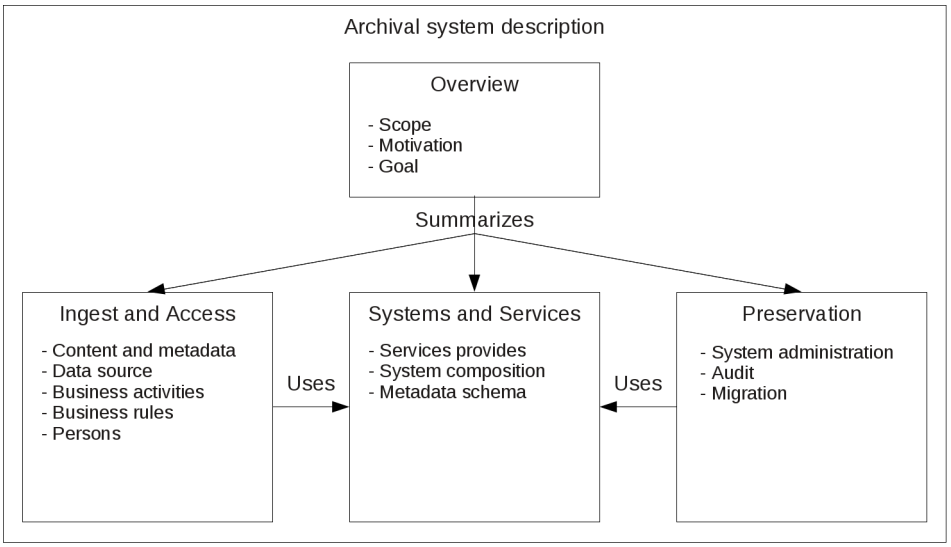

Figure 4.3: Organization of archival system descriptions

content item, and ends when this content is given to the archival system for preservation. The access activity starts when the consumer needs to acquire a particular content, and ends when the content is delivered. The Ingest and Access part identifies:

- the content ingested and accessed, and its associated metadata
- the systems from where the content originates
- the persons that perform the activities and their skills, to be monitored
- the details of the activities
- the standards (operational, business, technical, or industrial) and business rules

The Preservation part focuses on the interactions with the archival system performed by the preservers or maintainers:

- Archival Information Package (AIP) update
- disposal

The Systems and Services part is intended for the software developers. It provides a technology-dependent implementation of the archival system regarding the content management. RAAS supports the representation of:

- the services provided to the producers, consumers, and managers
- the subdivision and implementation of the archival systems (hardware and software)
- the connections and information flows between these subsystems
- the structure of the AIPs (stored by the archival storage system) and the metadata schema (managed by the data management system)

The selection of views is presented in Section 4.3.

4.2/ REPRESENTATION OF THE ARCHIVAL TERMINOLOGY

As stated earlier, RAAS relies on the DoDAF metamodel, or more precisely the UPDM profile, and the views used to describe architectures.

To understand our approach, knowledge of the profiling mechanism and of the UML metamodel is required[64]. UML defines a metamodel, which is composed of metaclasses such as Class, Property, Operation, or Activity. These metaclasses are then instantiated to create UML models. As a UML profile, UPDM defines stereotypes that are a way to categorize or specialize the UML metaclasses. A stereotype can then apply to an instance of the metaclass it extends. In UPDM, this specialization mechanism provides a way to develop architectural description using a specific enterprise terminology. For example, the metaclass Class can be specialized into new concepts, such as Capability, EnterpriseGoal, or Competence. These new concepts are defined as stereotypes that have Class as a metaclass. The goal of RAAS is to provide a more archival-specific terminology for describing archival systems.

This section presents the archival terminology defined in RAAS. This archival terminology consists in archival-specific stereotypes that specialize the UPDM stereotypes. Section 4.2.1 presents the stereotypes representing OAIS RM concepts. Section 4.2.2 presents additional stereotypes that would be present in a typical architectural description of archival system.

The archival terminology is organized by UPDM elements. For each UPDM element that is specialized, relationships with other UPDM elements are given. Not all the possible relationships are enumerated, but only those used in this thesis. A more complete description of the possible relationships is available in the UPDM specification[50]. Moreover, specific constraints that cannot be represented in the models are added on the archival terminology.

Each diagram depicts UML stereotypes, which represent either UPDM elements or the concepts added as part of RAAS. The UPDM concepts are displayed with a gray background, while the concepts added as part of RAAS are depicted with a white background. Each stereotype displays, between brackets, the UML metaclass that it extends. This section uses the same notation as in the UPDM specification. UML dependencies stereotyped by *metaconstraint* establish constraints among the UML metaclasses that are extended by the stereotype. The source of the metaconstraint defines the element being constrained, the *umlRole* property defines which property of the UML metaclass is being constrained, and the target of the dependency defines the restricted value.

For example, Figure 4.4 presents three stereotypes: *Details*, *ExchangedElement*, and *EntityItem*. *Details* extends the *Dependency* metaclass, *ExchangedElement* extends the *DataType* metaclass, and *EntityItem* extends the *Class* metaclass. Two metaconstraints are defined for *Details*. A first one goes from *Details* to *ExchangedElement* and has “supplier” as *umlRole*. This means that a UML element stereotyped by *Details* must have a UML element stereotyped by *ExchangedElement* as supplier. A second one goes from *Details* to *EntityItem* and has “client” as *umlRole*. This means that a UML element stereotyped by *Details* must have a UML element stereotyped by *EntityItem* as client. Section A.4 provides a subset of the UML metamodel, to cover the UML elements and their properties that are presented in this chapter.

Figure 4.4: Notation used for stereotypes

4.2.1/ REPRESENTATION OF OAIS CONCEPTS

This section presents concepts defined in the OAIS RM, and which are added as specialization of UPDM concepts.

Content and information packages Content and information packages describe information exchanged during archival-related activities. Content is the information that is meant to be preserved. Information packages encapsulate content information as well as additional information required to ensure a long-term preservation and accessibility. The SIP, AIP, and Dissemination Information Package (DIP) represent the information packages respectively as they are received, preserved, and disseminated. In UPDM, the concept *ExchangeElement* represents a resource exchanged during an activity. *ExchangeElement* is represented as a *DataType* in UML. The content and the information packages are defined as specializations of *ExchangeElement*:

- *SubmissionInformationPackage*
- *ArchivalInformationPackage*
- *DisseminationInformationPackage*
- *Content*

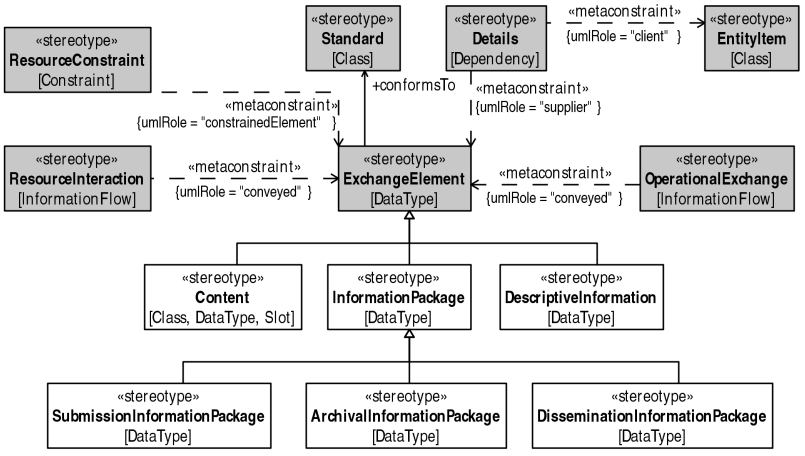

Figure 4.5: Extension of ExchangeElement

The *Content* stereotype does not only represent documents, it may also represent information contained within documents.

As it can be seen in Figure 4.5, an *ExchangeElement* can have a *Details* relationship to an *EntityItem*. *EntityItem* extends the *Class* metaclass, and the *Details* relationship aims to show how a piece of information is implemented by a logical structure (class or data type). *Details* extends the *Dependency* metaclass and has an *ExchangeElement* as supplier, and an *EntityItem* as client. An *ExchangeElement* can also be the subject of a *Constraint* and may have to conform to a particular *Standard*. Finally, an *ExchangeElement* can be conveyed during an activity (*OperationalExchange*) or during an interaction between resources (*ResourceInteraction*).

The following constraints are defined for the new elements:

- A *SubmissionInformationPackage* is exchanged between a *Producer* and an

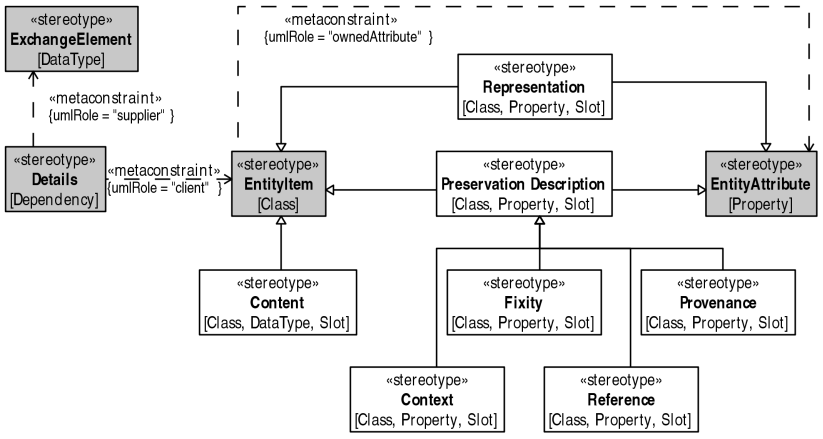

Figure 4.6: Extension of EntityItem

Archive
- A *ArchivalInformationPackage* is exchanged between a *Manager* and an *Archive*
- A *DisseminationInformationPackage* is exchanged between a *Consumer* and an *Archive*
- All the *SubmissionInformationPackage*, *ArchivalInformationPackage*, and *DisseminationInformationPackage* have a *Details* relationship that shows how the information package is implemented
- A particular content has at least one *ContentFormat*,
- *SubmissionInformationPackage*, *ArchivalInformationPackage*, and *DisseminationInformationPackage* have at least one *PackagingStandards*,
- A *Content* may have *ContentValidation*, *ContentModification*, or *ContentAccess* constraints.

Representation information and presentation description information Within information packages, the OAIS RM defines the different types of information that can be attached to the content. These are PDI and Reference information, as seen in Section 3.1.1. PDI is further detailed as being *Reference* information, *Provenance* information, *Fixity* information, and *Context* information.

In UPDM, these types of information can be seen either as *EntityItems*, or as *EntityAttributes*. *EntityItem* extends the *Class* metaclass, and *EntityAttribute* extends the *Property* metaclass. An *EntityItem* has *EntityAttributes* as owned attributes. It is possible to refer to PDI and representation information either for the definition of a particular type, or for the use of a type. Here are the specializations of *EntityItem* and *EntityAttribute*, as shown in Figure 4.6:

- *Representation*
- *Reference*
- *Fixity*
- *Context*
- *Provenance*

The following constraints are defined for the new elements:

- *Representation*, *Reference*, *Fixity*, *Context*, and *Provenance* are directly or indi-

rectly related to a *Content*.

Actors The OAIS RM defines three categories of actors that exist in the environment of the archive. These categories are *Producers*, *Consumers*, and *Management*. Management corresponds to the persons that set the policy of the archive. So, the preservers and other persons in charge of applying the policy are not considered. In terms of interaction with the archival system, the persons who interact with the archival system during the preservation phase should be represented. UPDM defines the notion of *Responsibility* to denote the particular role of a person. Here are the specializations of *Responsibility*, also displayed in Figure 4.7:

- *ProducerRole*
- *ConsumerRole*
- *PreserverRole*

Responsibilities are assigned to specific *Posts*. *Posts* are positions occupied by people in a company. *Competency* extend the *Class* metaclass. *RequireCompetence* extends the *Dependency* metaclass, and has a *Responsibility* as client and a *Competence* as supplier. *Post* and *Competency* extend the *Class* metaclass.

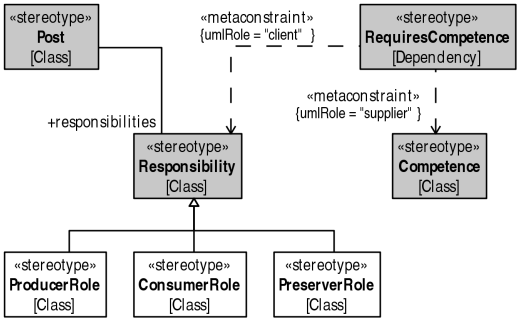

Figure 4.7: Extension of Responsibilities

The following constraints are defined for the new elements:

- A *ProducerRole* should have at least one *IngestCompetence*
- A *ConsumerRole* should have at least one *AccessCompetence* and one *UnderstandCompetence*
- A *PreserverRole* should have at least one *AccessCompetence*

Producer and consumers can also be seen as *Nodes* instead of physical persons. A *Node* is a logical abstraction, meaning that it may correspond to people or systems. The following specialization of nodes are added:

- *Producer*
- *Consumer*
- *Preserver*
- *Archive*

As shown in Figure 4.8, *Nodes* can be connected to *OperationalActivities* by *IsCapableOfPerforming*. *OperationalActivity* extends the *Activity* metaclass, and *IsCapableOfPerforming* extends the *Dependency* metaclass. *IsCapableOfPerforming* has an *OperationalActivity* as supplier and a *Node* as client. *Implements* extends the *Dependency*

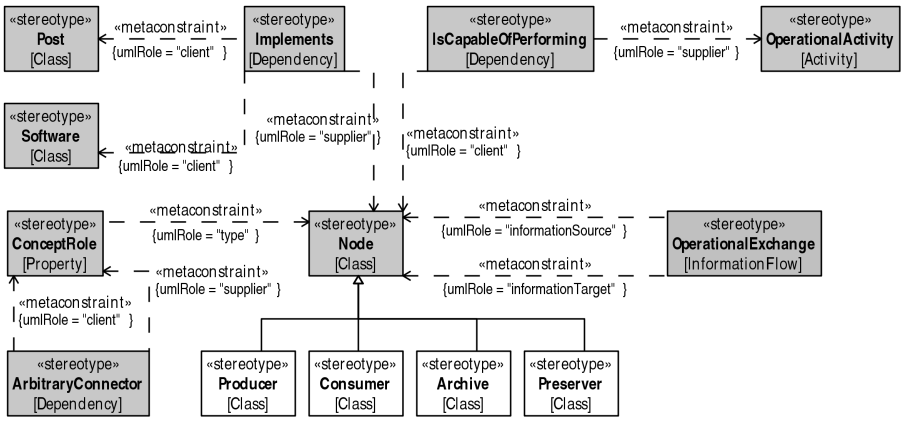

Figure 4.8: Extension of Nodes

metaclass and has a *Node* as supplier and a *Post* or a *Software* as client. Both *Posts* and *Software* extend the *Class* metaclass. An *OperationalExchange* represents an exchange of information between two *Nodes*. *OperationalExchange* extends the *InformationFlows* metaclass and has a *Node* as *informationSource* and *informationTarget*. *ConceptRoles* are connected to each other by *ArbitraryRelationships*. *ConceptRole* extends the *Property* metaclass, and *ArbitraryRelationship* extends the *Dependency* metaclass.

The following constraints are defined for the new elements:

- A *Producer* and an *Archive* should have at least one *OperationalExchange* that carries a *SubmissionInformationPackage*
- A *Consumer* and an *Archive* should have at least one *OperationalExchange* that carries a *DisseminationInformationPackage*
- A *Preserver* and an *Archive* should have at least one *OperationalExchange* that carries a *ArchivalInformationPackage*

Activities As a result of the three types of actors identified, three types of activities can be identified. The interactions between the archive and the producers, consumers, and management constitute respectively ingest, access, and management activities. In addition, the activities that are within the OAIS are also defined, in particular the preservation activities that include update and disposal of the preserved content. All of these activities are defined as specialization of *OperationalActivity* in UPDM:

- *IngestActivity*
- *PreservationActivity*
- *AccessActivity*

As seen in Figure 4.9, an *OperationalActivity* extends the *Activity* metaclass and has many relationships. *Performs* extends the *Dependency* metaclass and has a *Post* as client, and an *OperationalActivity* as supplier. *MapsToCapability* extends the *Dependency* metaclass and has an *OperationalActivity* as client, and a *Capability* as supplier. *SupportsOperationalActivity* extends the *Dependency* metaclass and has an *OperationalActivity* as supplier and a *ServiceInterface* as client. *IsCapableOfPerforming* extends the *Dependency* metaclass and has an *OperationalActivity* as supplier and a *Node* as client. *OperationalActivities* can be the subject to *OperationalConstraints*.

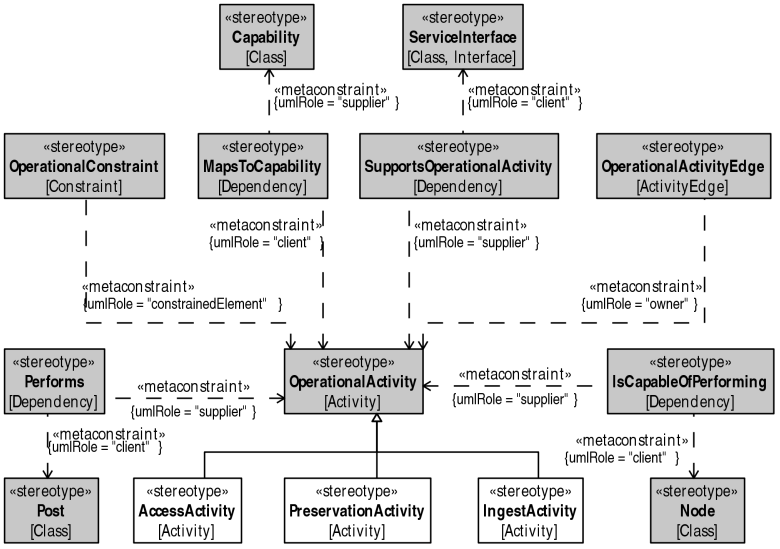

Figure 4.9: Extension of operational activities

The following constraints are defined for the new elements:

- An *IngestActivity* is connected to an *IngestCapability* and is performed by a *Post* with a *ProducerRole*
- *PreservationActivity* is connected to a *PreserveCapability* and is performed by a *Post* with an *ArchiveManagerRole*
- *AccessActivity* is connected to an *AccessCapability* and is performed by a *Post* with an *ConsumerRole*

OAIS functions The OAIS RM defines various functions that an OAIS performs. There are two kinds of functions: those intended to be performed by humans, and those intended to be performed by computers. Because it is meant to be generic and implementation independent, the OAIS RM does not prescribe what functions are performed by computers and what functions are not. On the other side, UPDM makes the distinction between these two kinds of functions, and calls them respectively *OperationalActivities* and *Functions*. The *Functions* that are likely to be implemented by systems are the following (see Figure 4.10):

- *IngestFunction* ingests the content
- *DataManagementFunction* manages the metadata
- *ArchivalStorageFunctions* manages the preserved content
- *AccessFunctions* makes the preserved content accessible

Each of these types uses OAIS functions:

- *ReceiveSubmission* receives the SIP
- *QualityAssurance* makes sure the SIP is correct
- *GenerateAIP* generates an AIP from the SIP
- *GenerateDescriptiveInformation* extracts the descriptive information from the SIP
- *CoordinateUpdates* synchronizes the storage of the SIP and the descriptive information updates
- *ReceiveData* receives a AIP and stores it

- *ProvideData* retrieves a stored AIP
- *ReceiveDatabaseUpdate* updates the database with descriptive information
- *ErrorChecking* checks that a stored AIP is not altered
- *ArchivalInformationUpdate* modifies an AIP to preserve the information
- *PerformQueries* queries the descriptive information to discover preserved content
- *CoordinateAccessActivities* manages the consumers queries and asks for a stored AIP
- *GenerateDIP* generates a DIP from an AIP
- *DeliverResponse* gives a response to the consumer

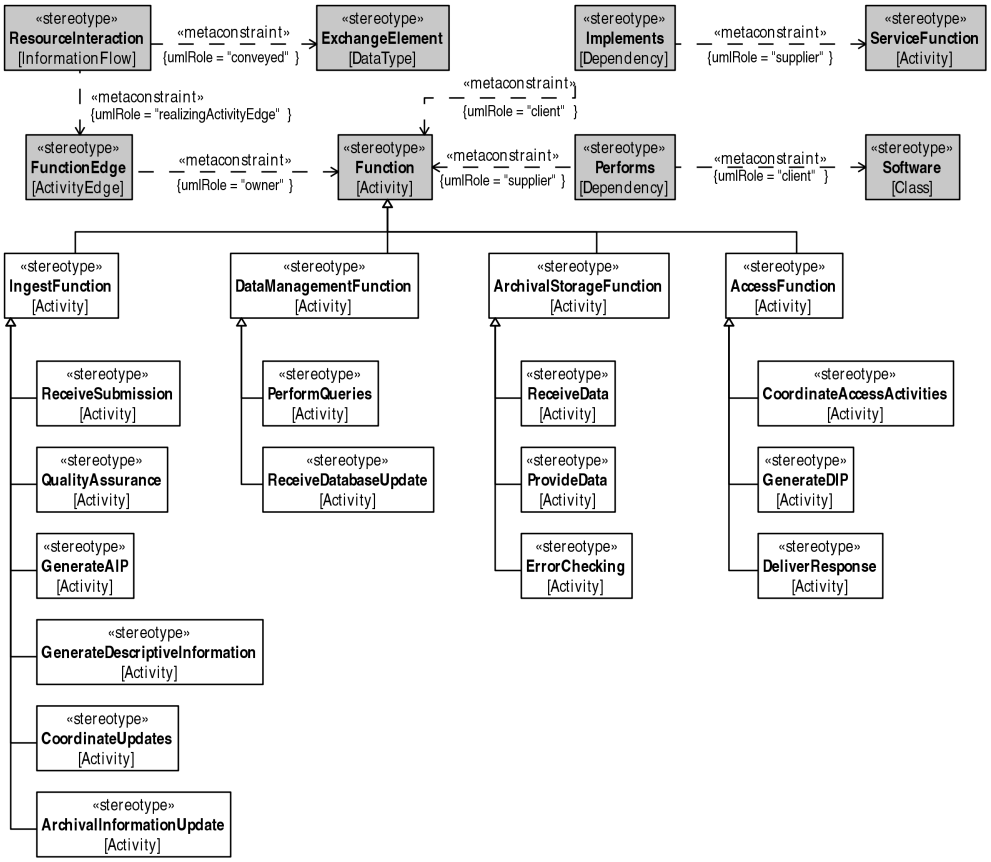

Figure 4.10: Extension of Function

Function extends the *Activity* metaclass. *Performs* extends the *Dependency* metaclass and has a *Function* as supplier and a *Software* as client. *Implements* extends the *Dependency* metaclass and has a *ServiceFunction* as client, and a *Function* as supplier. A *Function* may own *FunctionEdges*, which can be involved in *ResourceInteractions*. *ResourceInteraction* extends the *InformationFlow* metaclass and conveys *ExchangeInformation*.

4.2.2/ REPRESENTATION OF ADDITIONAL CONCEPTS

In addition to what is defined in the OAIS RM, several concepts were added because they will be required within an architectural description.

Capabilities A capability corresponds to the ability to achieve a particular goal. For an archive, the following high-level goals are defined as specialization of *Capability* (see Figure 4.11):

- *SubmissionCapability* is the ability to send information to the archive for preservation,
- *IngestCapability* is the ability for the archive to accept information to be preserved,
- *PreserveCapability* is the ability to make the preserved content stay understandable,
- *MakeAccessibleCapability* is the ability to allow consumers to discover and retrieve the preserved information,
- *AccessCapability* is the ability to access preserved information from the archive.

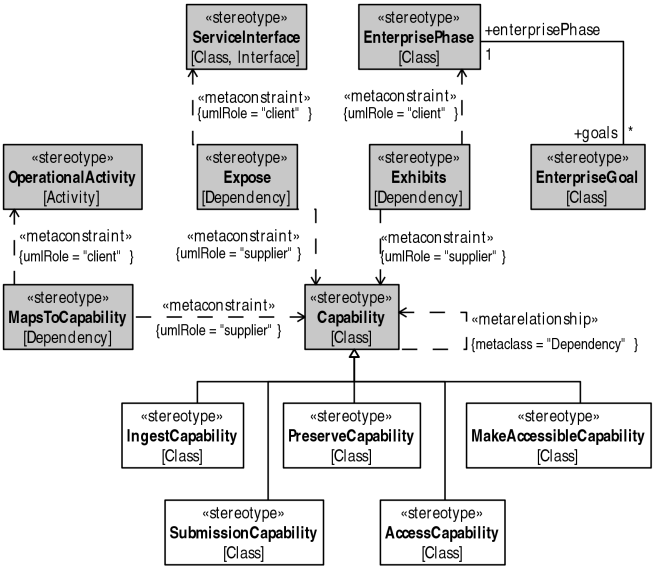

Figure 4.11: Extension of capabilities

EnterprisePhase extends the *Class* metaclass, and corresponds to a particular phase in the life of the enterprise. During this phase, the enterprise has many *EnterpriseGoals* and exhibits several *Capabilities*. *EnterpriseGoal* extends the *Class* metaclass. *Exhibits* extends the *Dependency* metaclass, with an *EnterpriseGoal* as client, and a *Capability* as supplier. *Capabilities* extends the *Class* metaclass. *MapsToCapability* extend the *Dependency* metaclass and has a *Capability* as supplier and an *OperationalActivity* as client. Finally, *Expose* extends the *Dependency* metaclass and has a *Capability* as supplier and a *ServiceInterface* as client.

The following constraints are defined for the new elements:

- *SubmissionCapability* should be mapped to by at least one *IngestActivity*,
- *IngestCapability* should be exposed by an *IngestServices*,
- *PreserveCapability* should be exposed by a *ManagementServices*,
- *MakeAccessibleCapability* should be exposed by an *AccessServices*.
- *AccessCapability* should be mapped to by at least one *AccessActivity*

Services Services constitute another concept that is important to an implementation. Nowadays many software development approaches use SOA to expose the functions performed by systems, and UPDM supports this approach by defining the notion of *Service*. In the case of the archive, there are different services to consider, as seen in Figure 4.12:

- *IngestServices* are the services exposed to the producers for the ingest
- *ManagementServices* are the services exposed to the preservers to make sure the content stays interpretable
- *AccessServices* are the services exposed to the consumers for accessing the preserved content.

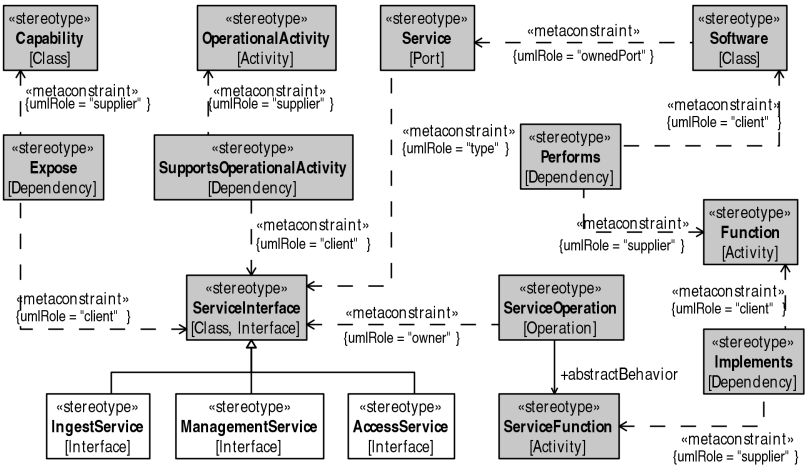

Figure 4.12: Extension of services

ServiceInterface extends the *Interface* metaclass. *Expose* extends the *Dependency* metaclass and has a *Capability* as supplier, and a *ServiceInterface* as client. *SupportsOperationalActivity* extends the *Dependency* metaclass and has an *OperationalActivity* as supplier and a *ServiceInterface* as client. *Service* extends the *Port* metaclass, is owned by a *Software*, and has a *ServiceInterface* as type. A *ServiceInterface* has *ServiceOperations*, which extend the *Operation* metaclass and is abstracted by *ServiceFunctions*. *Implements* extends the *Dependency* metaclass with a *ServiceFunction* as supplier and a *SystemFunction* as client.

The following constraints are defined for the new elements:

- *IngestService* expose an *IngestCapability*
- *ManagementService* expose a *PreserveCapability*
- *AccessService* expose an *AccessCapability*
- These services may have an *ArchiveAccess* constraint

Competences Competence, or skill, is another UPDM concept. *Competences* can be particularly useful to determine what skill or knowledge is needed to perform ingest or access activities. The following competences are identified (see Figure 4.13):

- *IngestCompetence* corresponds to the ability to prepare a SIP
- *AccessCompetence* corresponds to the ability to query and retrieve information from the archive

- *UnderstandCompetence* corresponds to the ability to understand a particular content

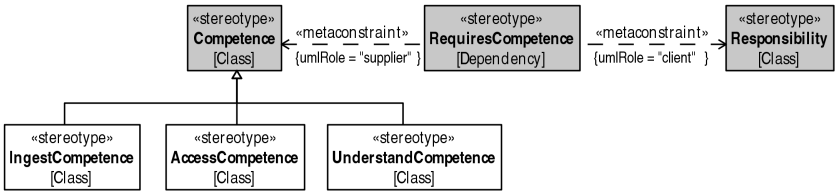

Figure 4.13: Extension of competences

Competences and *Responsibility* extend the *Class* metaclass. *RequireCompetence* extends the *Dependency* metaclass and has a *Responsibility* as client, and a *Competence* as supplier.

The following constraints are defined for the new elements:

- An *IngestCompetence* should be related to at least one *ProducerRole*,
- An *AccessCompetence* should be related to at least one *ConsumerRole* and one *ArchiveManagerRole*,
- An *UnderstandCompetence* shoud be related to at least one *ConsumerRole*.

Constraints Then, UPDM includes the notion of constraint that can apply to many UPDM concepts. Two types of constraints are actually used: *OperationalConstraints* applies to *OperationalActivities*, and *ResourceConstraints* applies to *ExchangeElements*. Both constraints extend the *Constraint* metaclass. The proposed approach includes representing the following constraints (see Figure 4.14):

- *ContentAccess* constraints apply to a content and restrict the access activities
- *ContentModification* constraints apply to a content and define what manipulations the archive can perform
- *ArchiveAccess* constraints apply to the services and express who is allowed to access the archive
- *ContentValidation* constraints apply to the content and define what makes it valid

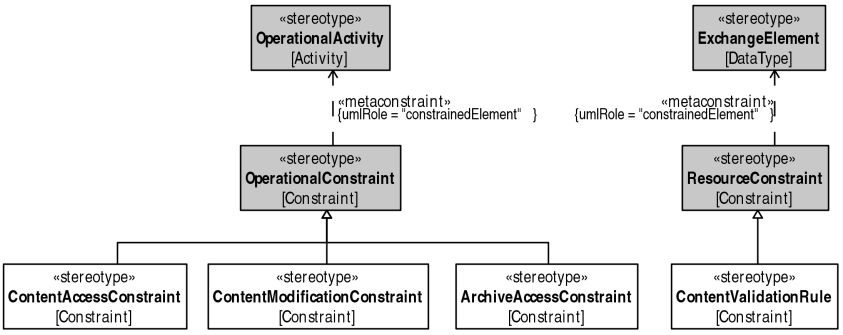

Figure 4.14: Extension of constraints

The following constraints are defined for the new elements:

- *ContentModification*, *ContentValidation*, and *ContentAccess* apply to *Content* only

- *ArchiveAccess* applies only to *IngestServices*, *AccessServices*, or *ManagementServices*

Standards Finally, UPDM defines the notion of standard against which UPDM elements should comply. Standard in this use is not restricted to international standards, but is defined broadly as a formal agreement. As a result, it can be used to represent the following aspects (see Figure 4.15):

- *ContentStandards* are formats used for the content
- *MetadataStandards* are formats used for PDI and the descriptive information
- *PackagingStandards* are formats used for the information packages

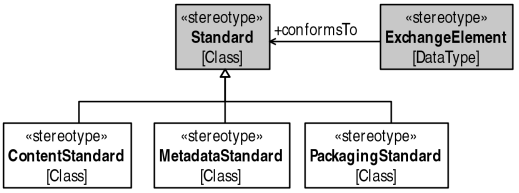

Figure 4.15: Extension of standards

Standard and its subtypes extend the *Class* metaclass and are connected to *ExchangeElement*.

This extension if subject to a set of constraints:

- *ContentStandards* apply only to *Content*,
- *PackagingStandards* apply only to either *SubmissionInformationPackage*, *ArchivalInformationPackage*, or *DisseminationInformationPackage*.

This section presented an extension of UPDM concepts to support concepts specific to archives. Section 4.3 shows how these new concepts can be actually used within DoDAF views.

4.3/ ARCHIVAL SYSTEM DESCRIPTION

This section presents a selection of views that can be used to describe the archival system architecture. This selection consists of DoDAF views applied to show a particular aspect of the archival system or its environment. In UPDM, DoDAF views are implemented by UML diagrams. A goal of this selection of views is to show a way to use the archival terminology previously defined, and to relate it to other UPDM concepts. A second goal is for these views to demonstrate how the archival system operates, so that it can be used for certification.

In this section, the four parts previously identified (Overview, Ingest and Access, Preservation, and Systems and Services) are presented. Figure 4.16 depicts these four parts, and lists the DoDAF views that compose each part. The following sections further detail each part, and provide more explanations on the views.

As explained in Section 3.3.2, DoDAF views are organized in viewpoints:

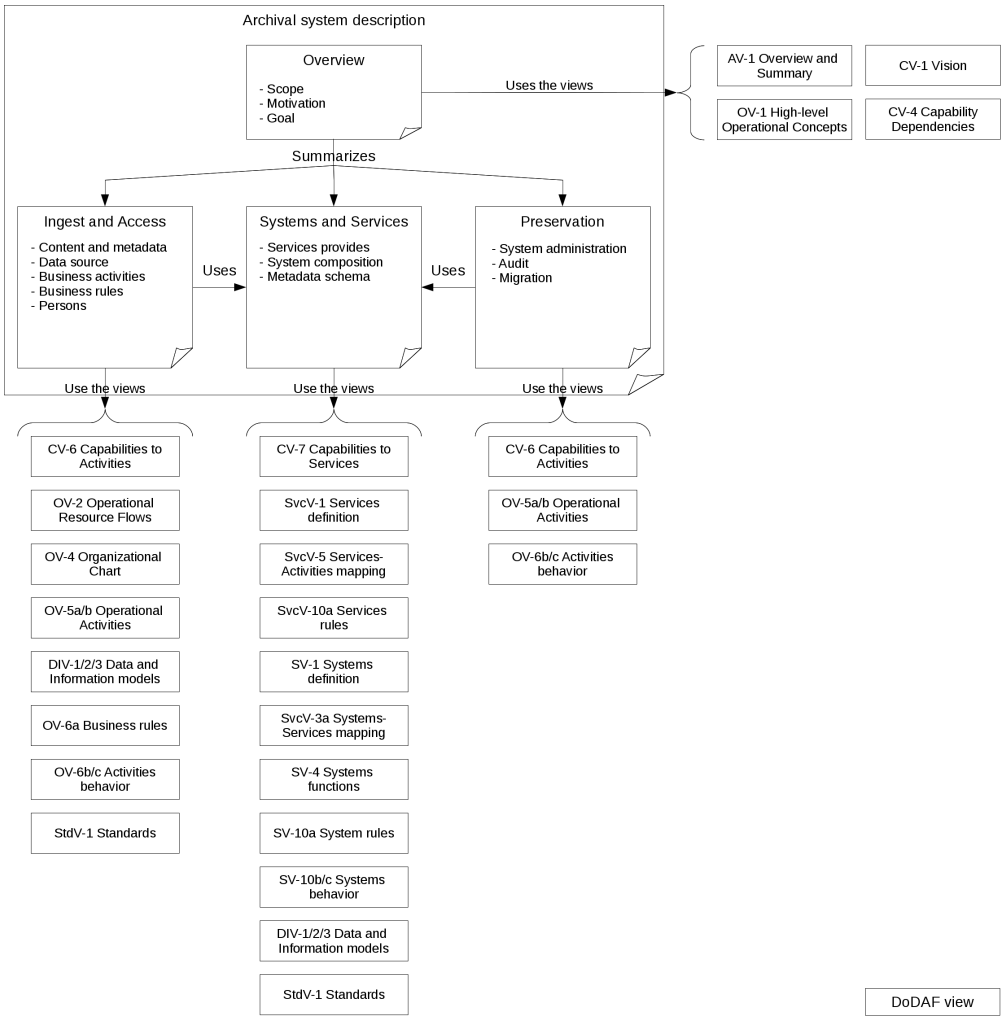

Figure 4.16: DoDAF views used in RAAS

- All Viewpoint (AV) describes the overarching aspects of an architectural description. In other words, this viewpoint describes all what the models of the architectural description have in common.
- Capability Viewpoint (CV) describes the capability requirements, definitions, and deployment.
- Operational Viewpoint (OV) describes the activities required to conduct operations.
- Services Viewpoint (SvcV) describes the services provided to support the activities.
- Systems Viewpoint (SV) describes the systems and their interconnections.
- Data and Information Viewpoint (DIV) describes the data used during the activities, by the systems or services.
- Standards Viewpoint (StdV) describes the policy, standards, guidance, constraints and forecasts that apply to the activities, systems or services.
- Project Viewpoint (PV) defines the projects and how they relate to the capabilities.

The views are implemented as UML diagrams, which display UML elements stereotyped by UPDM concepts.

When a view can serve as evidence for ACTDR, the relevant section of ACTDR is indicated as well as the role of the view.

4.3.1/ OVERVIEW OF THE ARCHIVAL SYSTEM

The Overview part gives a quick summary of the archive. It does not contain details about the implementation, but it defines the scope, the motivation, goal of the archive.

Overview and Summary Information (AV-1) is used as a high-level textual summary to explain what is in focus of the architectural description. AV-1 defines the scope, mission, purpose, and context of the architectural description. It explains why the archive is needed, and what objectives it fulfills. For example, it can present what types of information must be preserved over time. It also documents the standards used to guide the archive (the OAIS RM and ACTDR in this case). RAAS provides the generic scope, goals, and standards that should be used in the architectural descriptions. AV-1 can be completed by recommendations more specific to the context of the archive, or recommendations addressing different domains such as security or risk management.

ACTDR evidence: mission statement [31, Section 3.1.1], preservation strategic plan [31, Section 3.1.2]

High-Level Operational Concept Graphic (OV-1) is used to represent a graphical or/and textual description of the operational concepts. OV-1 is meant to be used in conjunction with AV-1 to provide an executive summary of the architectural description. An OV-1 presents a particular mission or scenario. OV-1 should identify the major activities or systems that require information preservation to achieve their objective, and the major activities or systems that produce this information. The systems and/or organizations communicating with the OAIS should be depicted.

Figure 4.17 shows an example of OV-1 composed of two data sources, the archive, and a target stakeholder. These entities are classified as *ConceptRole*, and are related to each other by *ArbitraryRelationships*: the two data sources communicate with the archive, and the archive communicates with the target stakeholder. These stereotypes are introduced in Figure 4.8.

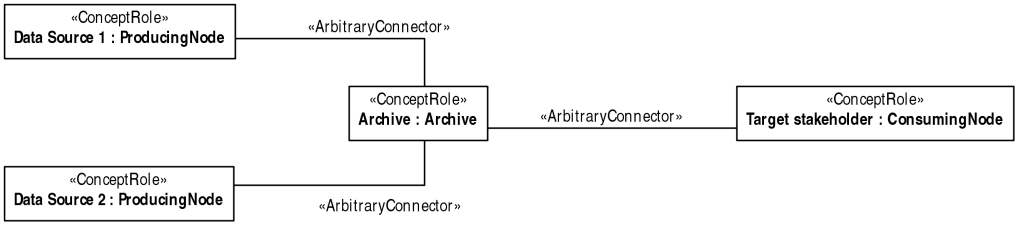

Figure 4.17: OV-1 view of the mission

ACTDR evidence: mission statement [31, Section 3.1.1], preservation strategic plan [31, Section 3.1.2]

Vision (CV-1) is used to provide a high level view of the capabilities. CV-1 connects the mission and goals of the enterprise with capabilities. In RAAS, CV-1 presents a formal definition for the mission, goals, and capabilities that require information preservation. In DoDAF, an architectural description is related to a particular *EnterprisePhase*. During this phase, the enterprise has *missions* and *EnterpriseGoals*. It also has particular *Capabilities* to achieve the goals. Figure 4.18 shows the relationships among these entities. These stereotypes are introduced in Figure 4.8.

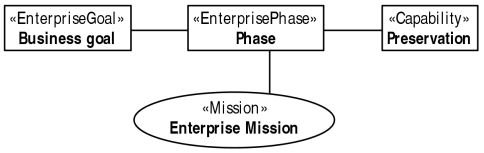

Figure 4.18: CV-1 view of the enterprise vision

ACTDR evidence: mission statement [31, Section 3.1.1], preservation strategic plan [31, Section 3.1.2]

Capability Dependencies (CV-4) can specify the dependencies among the capabilities. In RAAS, it is used to link the business capabilities with the preservation capabilities. These capabilities are then realized by the systems, services and operations described in the architectural description.

Figure 4.19 shows how the capabilities are connected. The *Preservation* capability depends on the *Preserve information and make it accessible* capability. This capability is actually composed of various capabilities: *InformationIngest*, *InformationPreservation*, and *InformationAccessibility*. Each of these capabilities are also applied the corresponding stereotypes presented in Figure 4.11.

InformationIngest depends on the *ProvideInformation* capability, which may correspond to a business function that produced the target information. On the other side, *UseInformation* may correspond to a business function that depends on the *InformationAccessibility* capability.

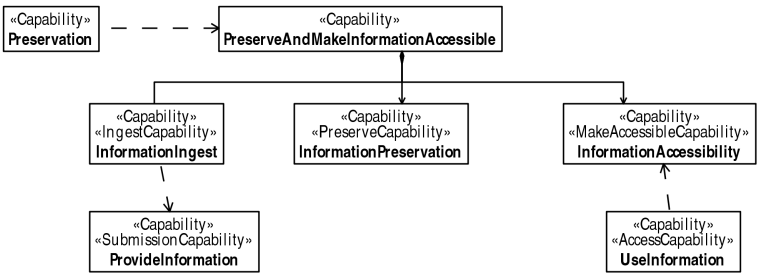

Figure 4.19: CV-4 view of the capability dependencies

4.3.2/ INGEST AND ACCESS ACTIVITIES

The Ingest and Access part focuses on the business activities related to the production and consumption of content. Grouping the information in this way ensures the content needed during the access is present during the ingest. The descriptive information needed to locate the content also needs to be present during the ingest. The Ingest and Access activities may be part of other business functions, when preservation meets a business requirement.

The models in this section provide a formal definition of the ingest and access activities including the systems, the persons (and their skills), the information exchanged (resource flows), and the sequence of actions. An archive typically deals with many different types of information, and each kind of information may have one or more ingest and access activities.

The details about the access activities constitute the user requirements. It allows to determine what information needs to be preserved, what the designated community is, and what descriptive information is needed. Not all the possible access scenarios can be predicted, but the most critical should be covered. The details about the ingest activities constitute the submission agreement between the producers and the archive. It tells what content and metadata the archive accepts, and who the producers are.

Capability to Operational Activities Mapping (CV-6) is used to describe how the capabilities are achieved through operational activities. In the context of RAAS, it describes how the capabilities relate to the production or consumption and defined in the Overview part relates to actual *OperationalActivities*.

Figure 4.20 shows this connection, using the stereotypes displayed in Figure 4.11. The *IngestActivity* maps to a *ProvideInformation* capability, and the *AccessActivity* maps to the *UseInformation* activity.

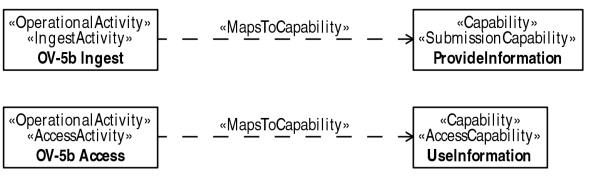

Figure 4.20: CV-6 view of the mapping between capabilities and ingest/access activities

Operational Resource Flow Description (OV-2) is used to represent the information

flows during the activity. An OV-2 presents *Nodes* and the information exchanges between these nodes. A *Nodes* is a logical entity that performs activities and that has a physical location. An *OperationalExchange* is an information flow between these nodes. In the context of RAAS, the *Nodes* represent the producers and consumers that interact with the archive. The *OperationalExchange* represents the information exchanged such as SIP and DIP. The *OperationalActivities* related to these *Nodes* are also given.

Figure 4.21 shows an example of OV-2. Three *Nodes* are depicted: a *ProducingNode*, a *ConsumingNode*, and the *Archive*. These nodes are applied the stereotypes presented in Figure 4.8. The SIP and DIP exchanged between the nodes are applied the stereotypes presented in Figure 4.5. The *IngestActivity* and the *AccessActivity* are also depicted in this diagram.

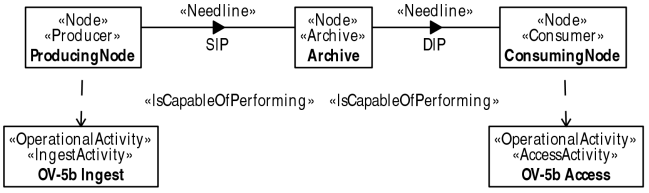

Figure 4.21: OV-2 view of the information flows within the archive

Organizational Relationships Chart (OV-4) is used to represent the *Organization* units and the posts. In RAAS, OV-4 represents the *Posts* involved in the production and consumption of the information, their *Responsibilities* (production/consumption), and the *Skills* associated with the *Responsibilities*. Information may be produced or accessed by different group of people. It is necessary to know who the producers are, as part of the submission agreement. Similarly, the consumers — especially the designated community — need to be identified. The skills, training and education associated with the roles can be included in the model. They can represent what skills are required for the producer to generate the SIP, and what skills are expected for the consumers to understand the information.

Figure 4.22 shows how the organization units and roles may be represented. The Figure represents two distinct *Organizations* that have two different posts. One post is for the *Producer*, and it requires the competence to create information. This post is associated with the *IngestActivity*. The second post is for *Consumer*, and it requires the competence: *AccessInformation* and *UnderstandInformation*. This post is associated with the *Access-Activity*. The posts are applied the stereotypes presented in Figure 4.7. The competences are applied the stereotypes presented in Figure 4.13

ACTDR evidence: designated community and knowledge base identification [31, Section 3.3.1] used for monitoring [31, Section 4.3.2]

Operational Activity Decomposition Tree (OV-5a) and **Operational Activity Model (OV-5b)** are used to detail the operational activities. Specifically, they can be used to decompose the ingest and access activities. For example, an activity of ingesting information includes selecting the right information, adding the required metadata, creating a SIP, and ingesting the SIP. The activity of accessing information for reuse can be decomposed into searching wanted information, and retrieving this information.

Figures 4.23 and 4.24 show examples of respectively an ingest and an access activity. The activities can display the *Nodes* (i.e. performers) defined in OV-2. The first activity

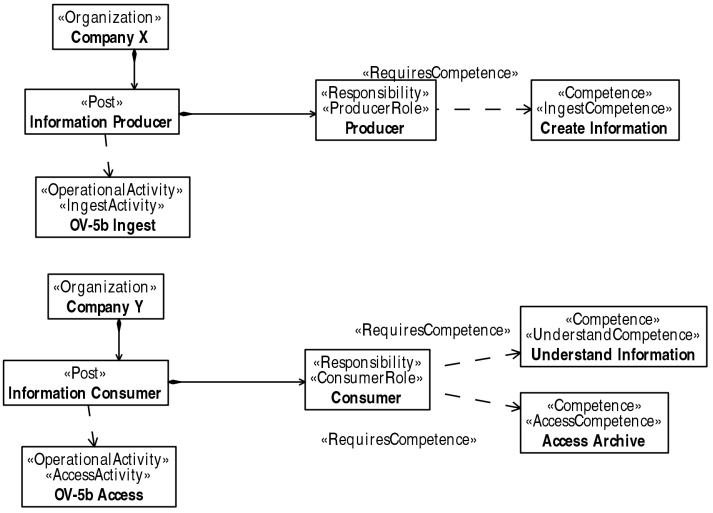

Figure 4.22: OV-4 view of the organizational chart

consists of a set of actions that are needed to prepare a SIP and send it to the archive. The second consists of a set of actions that are used to retrieve preserved information from the archive.

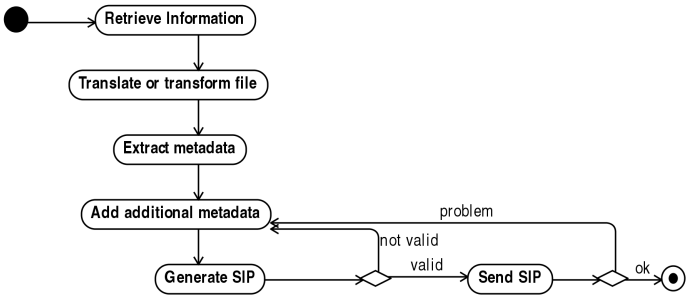

Figure 4.23: OV-5b view of the ingest activities

Conceptual Data Model (DIV-1), **Logical Data Model (DIV-2)** and **Physical Data Model (DIV-3)** focus, at three different levels of abstraction, on the information to be represented. DIV-1 is a conceptual information model focusing on the business concepts. DIV-2 is derived from DIV-1 to provide a logical representation, so that the information model can be represented using common computer languages. Finally, DIV-3 is a physical representation of the information using a particular language.

Information models are used to represent the content of SIPs, AIPs, and DIPs. Although the OAIS RM has a generic and conceptual description of these information packages, the actual implementation of these information packages is quite different. The implementation requires a precise and structured definition of the metadata to be given. This metadata, as defined in the OAIS RM, includes PDI and RI. PDI includes reference information (unique identifier in the context of the archive), provenance information (identification of the producer, history, authenticity), context information (relationship between the content and the environment) and fixity information (integrity information). Provenance information should make it possible to know precisely who the producer is. Fixity may

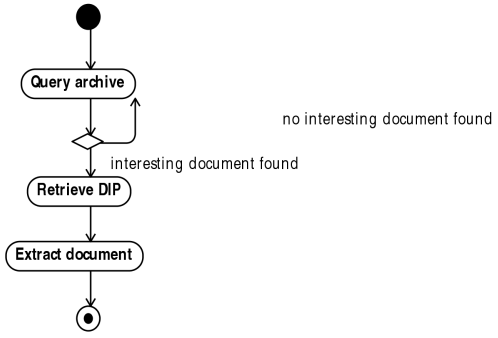

Figure 4.24: OV-5b view of the access activities

include checksum, or more advanced integrity information to make sure different representation share have the same information property. Additional metadata includes the rights associated to the content, or even the security aspect. The descriptive metadata, derived from the PDI, contains what is needed for accessing the preserved content. Each access scenario may have different requirements in terms of metadata. However, all this metadata needs to be present in the SIP and AIP. RI represents not only file formats, but any content that can help the designated community to understand the content. Most of the data defined conceptually may be already implemented in some fashion by existing information models or standards (e.g. Dublin Core[65]).

The approach RAAS recommends is to create a DIV-1 based on the conceptual information model of the OAIS RM. Then, DIV-2 and DIV-3 can be derived from DIV-1. The structure at the conceptual and logical level is likely to differ, so the relationship between an entity at the conceptual level and its representation at the logical level cannot be formally defined. However, RAAS makes it possible to attach an OAIS concept to elements of both conceptual and logical models.

Figure 4.25 shows the conceptual model for an information package. An information package may be a SIP, an AIP, or a DIP. Each information package has its respective stereotype applied, according to Figure 4.5. Each of these information package is conceptually different, but they all may contain the same kind of content. The Figure also presents different types of metadata that are associated with a content, and that are tagged with the kind of PDI it is as depicted in Figure 4.6: the *identifier* is *ReferenceInformation*, the format is *RepresentationInformation*, the version, the authors, and the owner are *ProvenanceInformation*, and the checksum is *FixityInformation*.

ACTDR evidence: Collection policy [31, Section 3.1.3], content information and information properties [31, Section 4.1.1], SIP definition [31, Section 4.1.2], and descriptive metadata definition [31, Section 4.5.1] including provenance [31, Section 4.1.4], reference [31, Section 4.2.4], fixity [31, Section 4.4.1.2], representation [31, Section 4.2.5] information

Standards Profile (StdV-1) contains the technical, operational, and business standards, guidance and policy. The information might be related to systems or information described in different architectural descriptions. For example, if a data source system sends information to the archive, the software and technology it uses to communicate should be watched. The data formats used to represent the information are also included. The model can represent the data exchange standards used during the ingest and access ac-

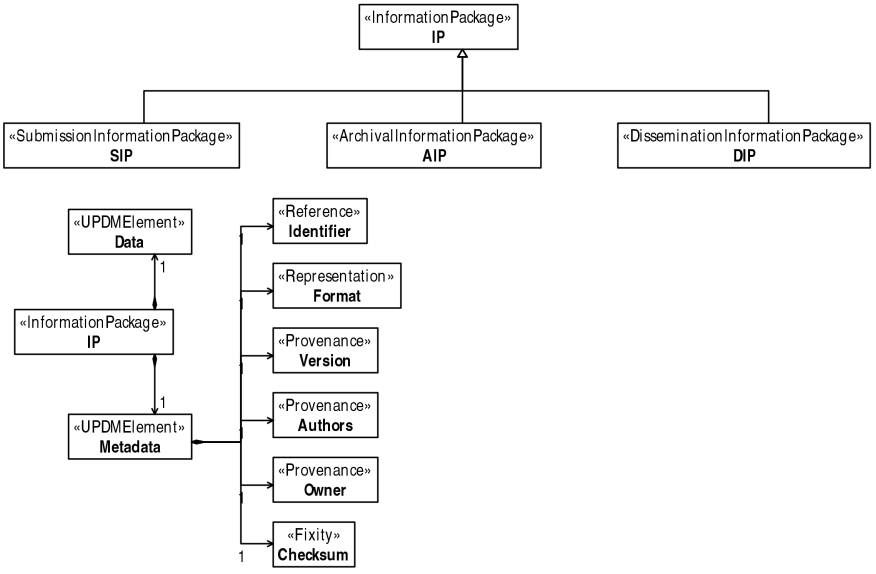

Figure 4.25: DIV-1 view of the conceptual information model of a Submission Information Package

tivities: the formats for the content, the packaging information, or any metadata standard. In particular, the information and standard required to parse any SIP are necessary. From the business aspect, the standards or policies rule the business activities.

Figure 4.26 shows an example of standards used for the following elements, stereotyped as presented in Figure 4.15: the SIP uses the FOXML[66] *PackagingFormat*, the metadata uses the Dublin Core *MetadataStandard*, and the data uses a particular *ContentStandard* as format.

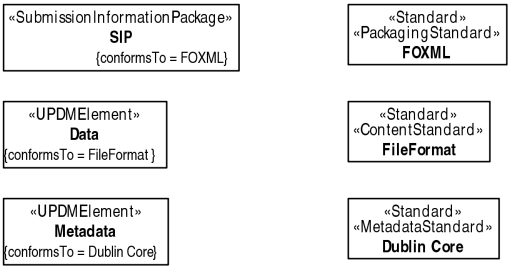

Figure 4.26: StdV-1 view of the standards used during the ingest and access activities

ACTDR evidence: SIP parsing technologies [31, Section 4.1.3], content formats used [31, Section 4.3.2]

Operational Rules Model (OV-6a) is used to represent the mission-oriented (business) rules that constrain the operations. The rules can apply to the mission, the performers, the activities, or the information flows. Such rules can specify:

- which content the archive must preserve and make accessible to consumers
- who is allowed to access the content
- what restrictions apply to the content (reproduction, modification, extraction)

Figure 4.27 shows an example of rules that apply to the entities. In this example, the rules constrain the *Nodes* by mandating who the producer should be, they constrain the access activity to assert who is allowed to access the content, and they assert that the archive must have the right to modify the content.

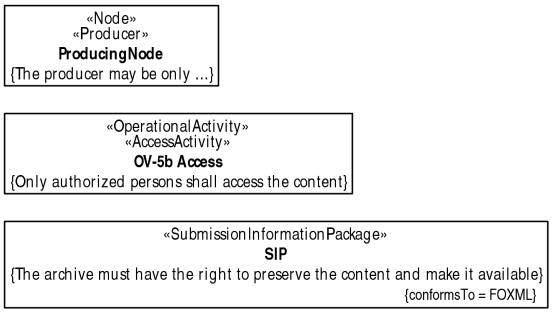

Figure 4.27: OV-6a view of the business rules for the ingest and access activities

ACTDR evidence: intellectual property rights [31, Section 3.5.2]

State Transition Description (OV-6b) and **Event-Trace Description (OV-6c)** are used to describe specific sequences and events regarding the operational activities. OV-6b targets one activity, and shows how this activity reacts to events by acting and changing state. OV-6c is used to represent the communication between activities, including what information is exchanged. A detailed ingest activity includes getting the content, extracting the metadata, packaging the content with the metadata, transforming the content from one representation to another, and sending the package to the archive. A detailed access activity includes querying the archive and getting the content. OV-6b can describe the different state of the ingest and access activities while an OV-6c can describe precise scenarios.

4.3.3/ MANAGEMENT ACTIVITIES

This part covers the preservation activities that interact with the archive. These preservation activities include update and disposal of the preserved content. Whereas ingest and access activities may be tied to non-preservation business activity, these activities are specific to the preservation. In the OAIS RM, these activities are considered as part of the OAIS.

In this part, **CV-6** describes how the capabilities related to the preservation defined in the Overview part relate to actual *OperationalActivities*.

Figure 4.28 shows this connection: the archival information update (migration) activity and the disposal activity are performed during the preservation information. The stereotypes used in this Figure are defined in Figure 4.11.

OV-5b is used to describe the steps involved in the preservation activities, which include *Disposal* of the AIP (see Figure 4.29), and the *UpdatingArchivalInformation* for the migration of the AIP (see Figure 4.30). The disposal activity includes getting a particular AIP identifier, and requesting the archive to dispose the content (AIP and associated descriptive information). The migration includes getting an AIP, performing migrations or adding new information, and sending the new package back to the archive.

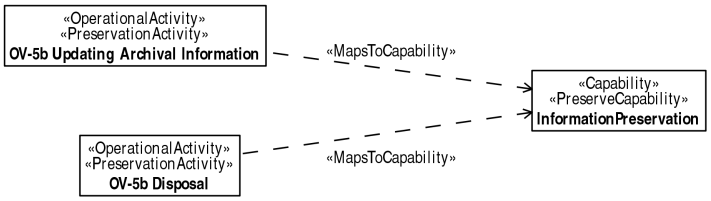

Figure 4.28: CV-6 view of the mapping between capabilities and preservation activities

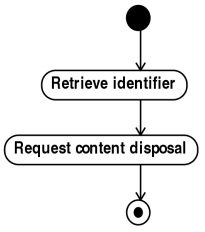

Figure 4.29: OV-5b view of the content disposal activity

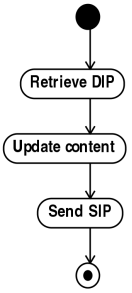

Figure 4.30: OV-5b view of the content migration activity

ACTDR evidence: processing procedure [31, Section 4.1.1]

OV-6b and OV-6c can represent in further details the sequence of events for the preservation activities. OV-6b can represent the generic approach to the disposal and migration, while OV-6c can represent the actual disposal and migration scenarios, involving the communications between the preservers and the archive in a technical fashion. Also, OV-6c can represent the events that trigger disposal and migration activities.

4.3.4/ ARCHIVAL SYSTEMS AND SERVICES

This part covers the internal composition of the OAIS, and the services it provides. Both systems and services provide functions, but there are differences between these two. Systems are considered in a "point-to-point" approach (explicit communications between system A to system B), whereas services are considered in a "net-centric" distributed approach (services publishers and subscribers). The services viewpoint can be used to define the services that the archival system proposes to the external performers (producers, consumers, preservers). Services are the way by which performers interact with the archival system. RAAS assumes that the archival system implementation uses the SOA.

The OAIS RM presents different types of functions performed by the archive. Some of these functions automatically process the information, and this thesis assumes that these functions will be implemented by computers. These functions are represented by the SV, and are exposed to the other systems and people by services, represented by the SvcV. The SV and SvcV models will be implemented by hardware and software.

Capability to Services Mapping (CV-7) is used to show the capabilities presented in the Overview part that are realized by services. Figure 4.31 shows an example of CV-7. The *InformationIngest* capability is made possible by an *IngestService*, the *InformationAccessibility* capability is made possible by an *AccessService*, and the *InformationPreservation* capability is made possible by a *ManagementService*. The stereotypes used are presented in Figure 4.12.

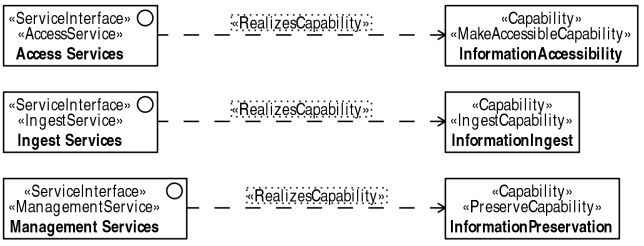

Figure 4.31: CV-7 view of the mapping between capabilities and services

Services Context Description (SvcV-1) is used to represents the services, service items and their interconnections. In DoDAF, the services support the operations and capabilities of the architectural description. SvcV-1 presents the interactions between services and humans, including the information exchanged. In this section, SvcV-1 shows the services provided by the archive to support the ingest, access, and management activities. Three types of *ServiceInterfaces* are defined, one corresponding to each kind of activity (see Figure 4.32).

ACTDR evidence: ability to list preserved content [31, Section 4.1.6], ability to show the

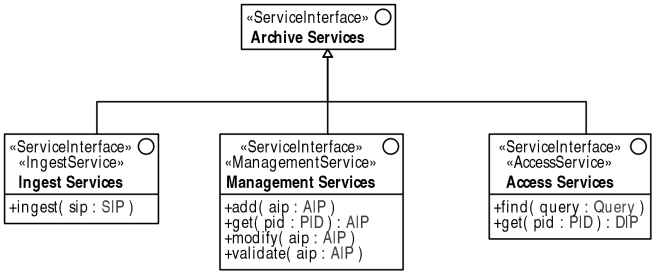

Figure 4.32: SvcV-1 view of the services provided by the archive

history of digital objects [31, Section 4.1.8]

Operational Activity to Services Traceability Matrix (SvcV-5), as shown in Figure 4.33, describes the mapping between the *OperationalActivities*, described in the ingest, access, and management activities, and the *ServiceInterfaces* provided by the OAIS. This model is needed because different activities may call the same service in the OAIS. The ingest services are used by the ingest activities, the accesses services are used by the access activities, and the management services are used by the preservation activities.

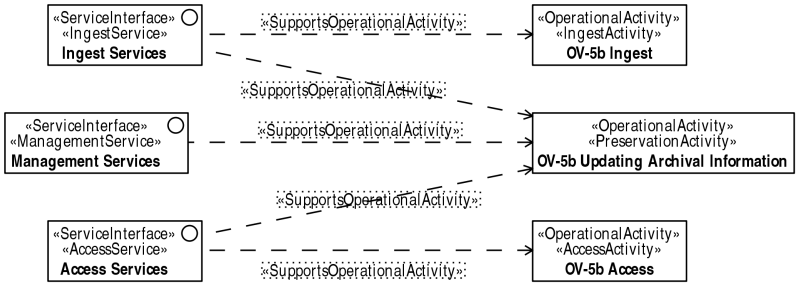

Figure 4.33: SvcV-5 view of the mapping between archive services and activities

Services Rules Model (SvcV-10a) is used to represent the constraints associated to the service. In this section, the constraints can include who is using the services (which producers, which consumers), or what data — SIP or DIP — is accepted. For the ingest and access services, most of the constraints are already stated in the ingest and access activity descriptions. Figure 4.34 shows an example of constraints on the archive services. These rules include who is authorized to use each service.

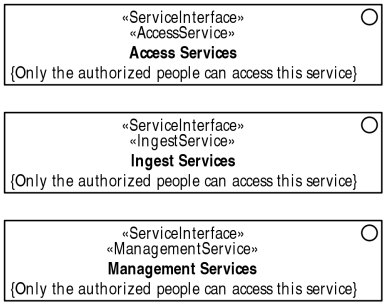

Figure 4.34: SvcV-10a view of the constraints on the archive services

ACTDR evidence: access policy [31, Section 4.6.1]

System Interface Description (SV-1) is used to represent a high level description of the systems, their compositions, and their interconnections. In this section, SV-1 represents the subsystems of the archival system, and the performers. SV-1 represents the physical architecture, whereas OV-2 represents the logical architecture. Figure 4.35 shows an example of physical architecture based on the OAIS RM. Three different software are used: the *ArchivalStorageSoftware*, the *DataManagementSoftware*, and the *ArchiveSoftware*. The *ArchiveSoftware* exposes the services previously presented. SV-1 could also capture the connections among several archives.

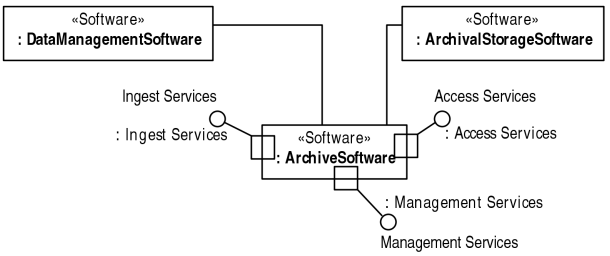

Figure 4.35: SV-1 view of the archival system

Systems-Services Matrix (SvcV-3a) is used to describe the relationships between the services provided by the OAIS and the related OAIS system components. In our case, all the services communicate only with the archive software.

Systems Functionality Description (SV-4) is used to represent the system functions. In RAAS, it is used to show what actions are are executed, and by which component. Two different functions are presented: an *IngestFunction* (Figure 4.36) and an *AccessFunction* (see Figure 4.37). These functions are taken from the OAIS RM. Different system functions can be defined according to the OAIS RM, and applied the stereotypes defined in Figure 4.10.

ACTDR evidence: SIP verification [31, Section 4.1.5], AIP generation [31, Section 4.2.2], AIP disposal [31, Section 4.2.3], Error checking [31, Sections 4.2.8, 4.2.9, 4.4.1.2], Receive data [31, Section 4.4.1], Generate DIP [31, Section 4.6.2].

Systems Rules Model (SV-10a) represents the various constraints that apply to the performers, information flows, functions and data related to the system. There are some categories of rules that apply to any kind of content, but some rules will depend on the content. As examples of rules, SIP and the AIP have to be checked for completeness and correctness. Integrity rules for each kind of model can be documented in this view (checksums or more advanced methods). Rules also make sure the AIP, and not just the content, is well formed (following particular standards or information models). These rules may be similar to the ones used for the SIP. All the operations performed on the AIP need to be clearly documented.

Systems State Transition Description (SV-10b) and **Systems Event-Trace Description (SV-10c)** describe the states, events and transitions associated with the system. These models can detail the functional entities as defined in the OAIS RM, but with more specific actions. As previously explained, specific content sometimes requires specific processes. So, in addition to generic processes, the following specific processes could be defined when needed:

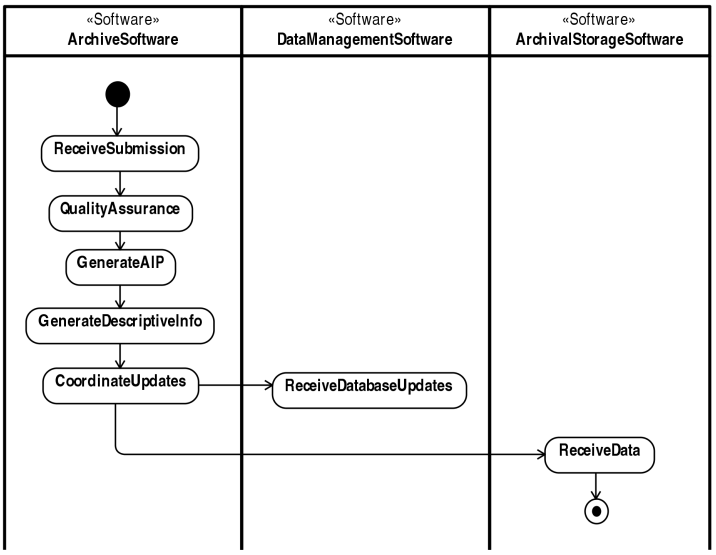

Figure 4.36: SV-4 view of the ingest function

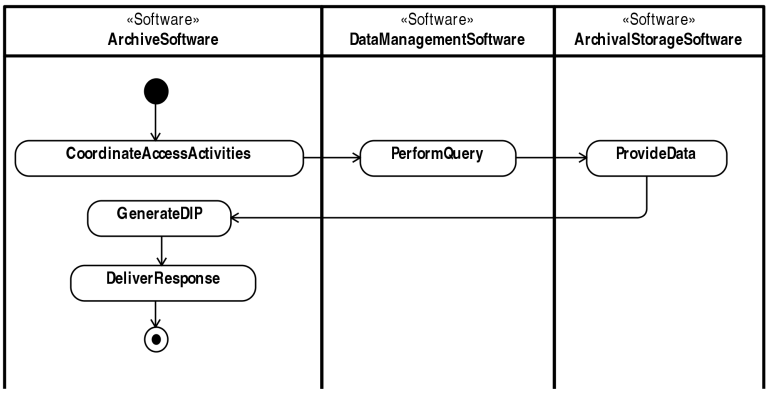

Figure 4.37: SV-4 view of the access function

- transformation from SIP to AIP, and disposition of SIP
- SIP validation
- Identifier resolution

SV-10b can represent the change of state for each subsystems, while SV-10c can represent a scenario of exchange between different subsystems.

The following Figures show how the components of the archive respond to different activities: ingest (Figure 4.38) and access (Figure 4.39).

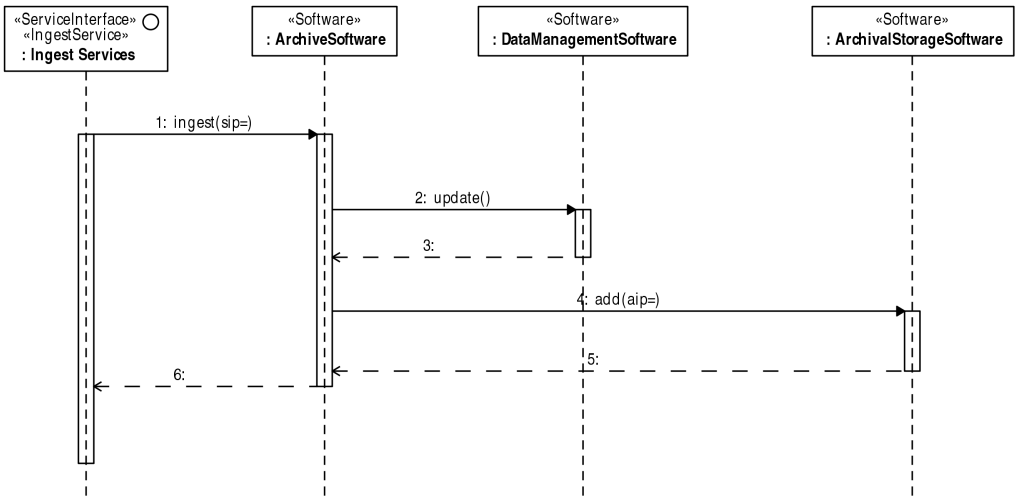

Figure 4.38: SV-10c view of the ingest sequence

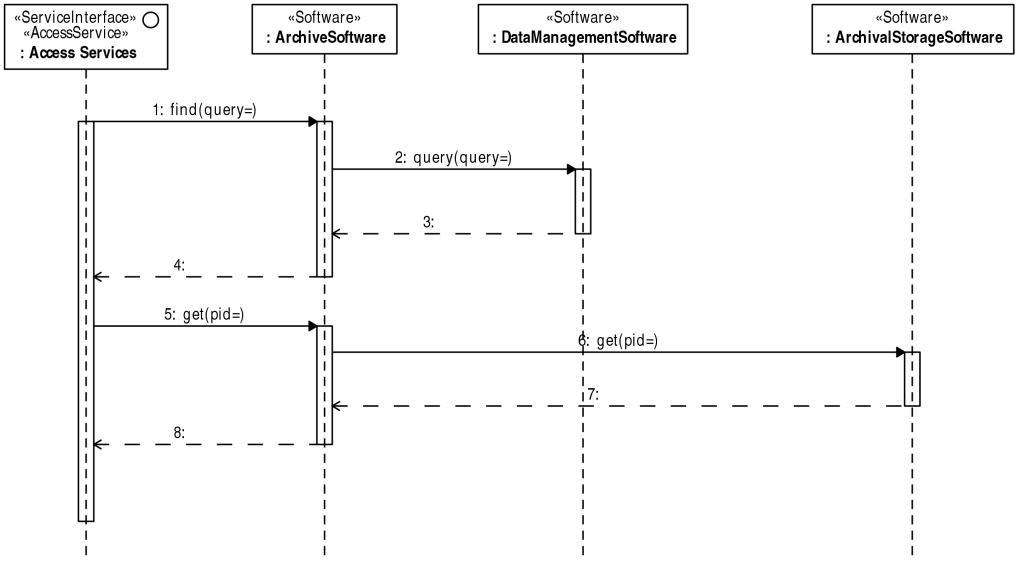

Figure 4.39: SV-10c view of the access sequence

Also, in order to satisfy the requirements for certification, the following procedures need to be supported:

- changing the AIP identifier,
- supporting the change of “preservation status”,

- checking the integrity of the collection as a whole.

DIV-1, **DIV-2** and **DIV-3** should represent the metadata schema for the archive (descriptive information). This descriptive information is deduced from the access scenarios previously identified. Different access scenarios involve different content items and even sometimes different metadata for the same content. DIV-1 can be used to represent the metadata at the conceptual level. At the logical levels, the DIV-2 metadata model can rely on information models optimized for a particular domain.

In addition to the metadata schema, it is necessary to represent the AIP. The content of AIP differs from the descriptive information because it contains additional information required to preserve the content over time, such as checksums to verify the integrity. The AIP also contains additional information including, for instance, the status of preservation, intellectual property rights, and ownership.

ACTDR evidence: AIP definition [31, Sections 4.2.1, 4.2.6] and descriptive metadata definition [31, Section 4.5.1] including provenance [31, Section 4.1.4], reference [31, Section 4.2.4], fixity [31, Section 4.4.1.2], representation [31, Section 4.2.5]

StdV-1 is used to define the standards necessary to parse the AIP. The same implementation can be used for all the AIPs, SIPs, and DIPs.

4.4/ SUMMARY AND APPLICATION OF RAAS

This chapter presented a Reference Architecture for Archival Systems, called RAAS, to formally describe archival systems. RAAS focused on the description of the information system in charge of managing preserved information, as well as the interactions of this system with its environment. RAAS supports the representation of archival systems within complex environments, and it also supports the representation of complex information.

RAAS supports the formal description of various concepts related to archival systems. These concepts are displayed in Figure 4.40. RAAS intends to represent:

- the activities that involve the archival system,
- the constraints that apply to the activities,
- the functions performed by the archival system,
- the information involved in the activities and functions,
- the posts (or roles) that perform the activities, as well as the associated responsibilities and competences associated required to perform the activities
- the services provided by the archival system, how the activities use these services
- the capabilities that the services expose

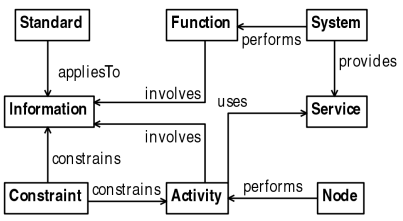

Figure 4.40: Enterprise concepts covered by the reference architecture

The next chapter demonstrates RAAS by describing various aspects of a product model archival system. Product models are a kind of complex information that need to be preserved, and that may exist in a complex environment.

5

USING RAAS FOR PRODUCT MODELS PRESERVATION

This chapter presents the description of a product model archival system. This description relies on the previously introduced Reference Architecture for Archival Systems (RAAS), which guides the description of archival systems and their environments. This description aims to represent the main aspects of the archival system, such as information, activities, system functions, services, and interactions. This description should refer to the generic concepts defined in the Reference Model for an Open Archival Information System (OAIS RM), and provide proofs that can be used for the certification of the archival system.

Section 5.1 provides background information to understand the motivation and the objective of the archival system description. Section 5.2 presents the architectural description of the product model archival system, according to the RAAS presented in the previous chapter. Section 5.3 provides conclusion on the architectural description.

5.1/ INTRODUCTION TO THE PRODUCT MODEL PRESERVATION CASE

The product models targeted in this chapter are actual product data that concerns a ship called Torpedo Weapon Retriever (TWR). The product data for this ship was made available by the Navy for educational purpose. To restrict the scope of the archival system, the objective is to show how the product models can be preserve and made accessible to support the maintenance of a particular part of the ship. While the use case concerns actual ship data, the same approach can be reused for other types of products.

Maintenance activities often need product data that was created a long time ago. In the case of ships, which may stay in service for more than 30 years, such data may have been produced decades ago. For example, a maintenance activity may require location of the part, photos, and maintenance procedure. Product data and product metadata may be complicated to represent and to manage. Product models are formal representation (not necessarily visual) of products.

The product model dataset originates from a system called Leading Edge Architecture for Prototyping Systems (LEAPS). LEAPS is a Product Data Management (PDM) system that stores various product data, such as product structure, part connections, system breakdowns, and file management. The files managed by LEAPS include various en-

gineering drawings, 3D Computer-Aided Design (CAD) files, photos, part manuals, and catalogs. As this chapter targets product models in particular, only the CAD files and the PDM information are in scope.

The creation of a product model archival system faces different issues. A first issue is to determine what information needs to be preserved. Product models are originally located in a different system, in a format that may not be suitable for preservation. The format has to provide guarantees that the data can be interpreted by both consumers, and preservers when migration is needed. Product models in the shipbuilding domain are characterized by interoperability issues due to the formats used[19, 21]. These interoperability issues have to be addressed for product models to be preserved and made accessible.

So, product information has to be identified and transformed into a more suitable format. Among the product information that needs to be identified, metadata is of vital importance. Such metadata aims to make sure product models can be preserved, understood, and made accessible.

This chapter presents the architectural description of a product model archival system. This architectural description include: the specification of ingest and access activities, the high-level design of the system to support these activities, and the preservation activities. The description of the access activities aims to identify what information needs to be preserved, and what metadata can help accessing this information. The description of ingest activities aims to determine how the preserved information has to be submitted to the archive. The high level system design addresses how the archival communicates with its environment, and what functions it performs. The preservation activities aims to make sure the basic preservation activities are supported.

The archival system description provides an abstraction that makes it easier to understand how preservation is addressed. The archival system description connects the preservation concepts to their actual implementation. The description also provides traceability from business requirements to system design.

5.2/ ARCHITECTURAL DESCRIPTION OF THE PRODUCT MODEL ARCHIVAL SYSTEM

This section presents the architectural description of a product model archival system, according to the Reference Architecture for Archival Systems presented in Chapter 4. The preserved content is composed of ship design information, which has to be made available over time to support maintenance activities.

The architectural description is organized in four parts, according to the organization presented in Figure 4.3: Overview, Ingest and Access, Preservation, and Systems and Services. The diagrams (also called views) shown in this description are done using MagicDraw 17 UML. The architectural description uses the Unified Profile for DoDAF/-MODAF (UPDM) profile and the archival-specific stereotypes introduced in Chapter 4.

5.2.1/ OVERVIEW

This section presents an overview of the architectural description. It briefly provides the motivation, scope of the archival system and the main external entities and activities that interacts with this system.

Overview and summary This architectural description focuses on describing the structure of a product model archival system developed to support a maintenance scenario. The description targets the development of the information system, the ingest and access activities, and the common content management. This architectural description relies on the archival-specific terminology and the views presented in Chapter 4. Ships in service are required to have regular maintenance activities, which are performed to make sure the ship can operate correctly. Product models are needed to effectively support the maintenance activity. But as ships may stay in service for decades, these product models need to be preserved correctly.

Operational concepts Figure 5.1 shows the main operations related to the archive. The mission of the archive is to preserve product models received from a PDM system called LEAPS, preserving them, and making them available to maintainers. The diagram depicts different *ConceptRole*: the LEAPS system, the archive, and the maintenance. These concepts are related by *ArbitraryConnectors*. These stereotypes are introduced in Figure 4.8.

Figure 5.1: OV-1 view for the overview of the maintenance scenario

Vision Figure 5.2 presents the vision of the enterprise. The mission of the Navy being to “maintain, train and equip combat-ready Naval forces capable of winning wars, deterring aggression and maintaining freedom of the seas.” In this particular example, the high level *EnterpriseGoal* is to have the ship supporting the mission of the Navy. So, during a particular *EnterprisePhase*, this goal can be reached thanks to the *MaintainShip* capability. These stereotypes are presented in Figure 4.8.

Concerning the Audit and Certification of Trustworthy Digital Repositories (ACTDR) evidence, this diagram reflects the involvement of the archival system as part of the mission of the organization [31, Section 3.1.1].

Dependencies Figure 5.3 shows the dependencies between the different capabilities. To achieve the *MaintainShip* capability, the following dependent capabilities are needed: *AccessProductModels*, *MakeProductModelsAccessible*, the *PreserveProductModels*, and *IngestProductModels*. Each of these capabilities corresponds to a set of activities or services that implement the capability. This diagram provides a high-level view of what the archival system does and supports. In a more complex example, when

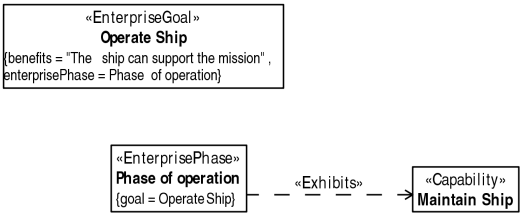

Figure 5.2: CV-1 view for the enterprise vision

the archive has to support different types of content, the diagram can display how the archival system supports other missions. The stereotypes used are as presented in 4.11.

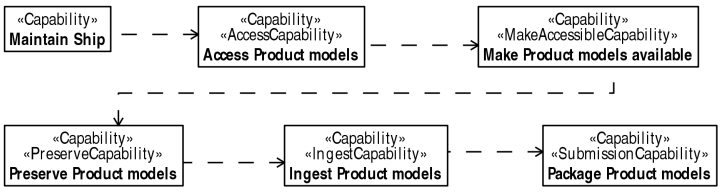

Figure 5.3: CV-4 view for the capability dependencies

5.2.2/ INGEST AND ACCESS ACTIVITIES

This part details on the ingest and access scenarios that are related to the archival system. The ingest activities consist in taking content from a data source, packaging it, and sending the package to the archival system. The access activity is a business activity that the archival system supports by giving access to preserved information.

Mapping the ingest and access capabilities to activities The capabilities related to the ingest and access to product models are mapped to specific activities, as seen in Figure 5.4. In this scenario, one activity is an *IngestActivity*, and one activity is an *AccessActivity*. The ingest activity is the ingest from the LEAPS system, and the access activity is the maintenance activity. In a more complex example, different ingest and access activities may be defined. For example, the product model archive may have to get information from different data sources: PDM systems, Enterprise Resource Planning (ERP) systems, logistics systems, or maintenance systems. Similarly, different access activities may be supported, depending on how the preserved product models are meant to be used. The stereotypes used in this diagram are defined in Figure 4.9.

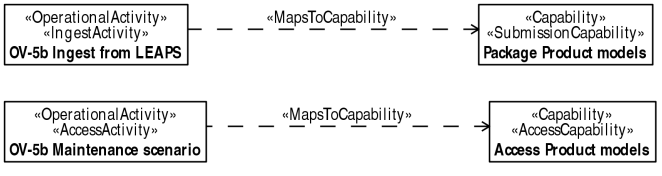

Figure 5.4: CV-6 view for the mapping between preservation capabilities and activities

Information flows Figure 5.5 presents information flows related to the archival system. Three distinct operational *Nodes* are identified. A node can be seen as an abstraction of people and systems, who exchanges information. The product models are initially managed by the *LEAPS* system, which serves as *Producer*. They are then packaged and sent to the archival system as *LEAPS SIP*. Then, during their preservation within the archival system, product models are accessed by the *Maintenance* and *Logistics* nodes, who serve as *Consumer*. The product models are transmitted as *Maintenance DIP*. The stereotypes used are defined in Figure 4.8.

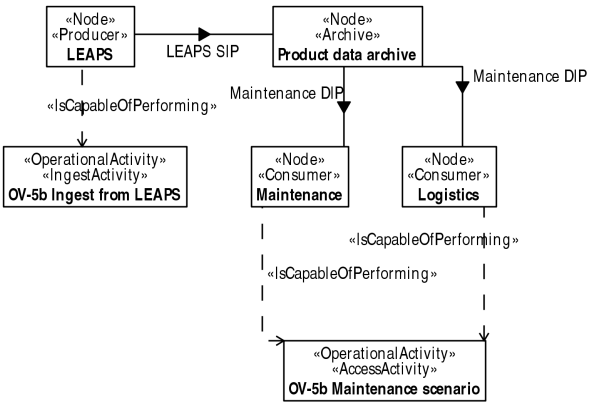

Figure 5.5: OV-2 view for the ingest and access of product models

Composition of the SIP/DIP Figures 5.6 and 5.7 give a conceptual description of the content of the *LEAPS SIP* and *Maintenance DIP*, categorized by different kinds of content. The Figure also shows what information is meant to be represented by the files: the significant properties. The *LEAPS SIP* contains a list of files:

- 2D drawings provide geometry and tolerance information for the designed parts
- specifications provide material and size information,
- CAD files provide 3D geometry representation,
- user manuals provide various installations and maintenance procedures,
- pictures provide visual representation,
- PDM information provide the bill of materials, system and zone breakdowns, document management, and parts connections.

The different types of file are identified as *ExchangeElements* and as *Content*, which means they are the target of the preservation. The stereotypes used in this diagram are presented in Figure 4.5.

PDM Information contains information related to the other files (e.g. relationships to the parts), so it serves both as content and as metadata.

The *Maintenance DIP*, in this example, consists of *CAD files*, *Photos*, *User manuals*, *Part specification*, and *PDM information*.

Note that the preservation of photos, user manuals, and specifications is out-of scope, as they are not product models. However their metadata, which includes relationships among these files and product data, consists in product models, so it is in the scope.

Regarding the ACTDR evidence, this diagrams explicitly defines the collection policy[31, Section 3.1.3], as well as the content and information properties being preserved[31, Section 4.1.1]

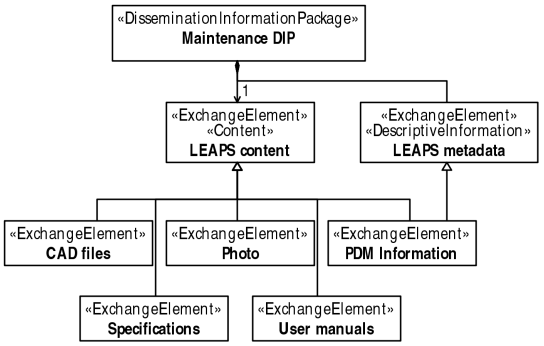

Figure 5.6: DIV-1 view of the maintenance DIP

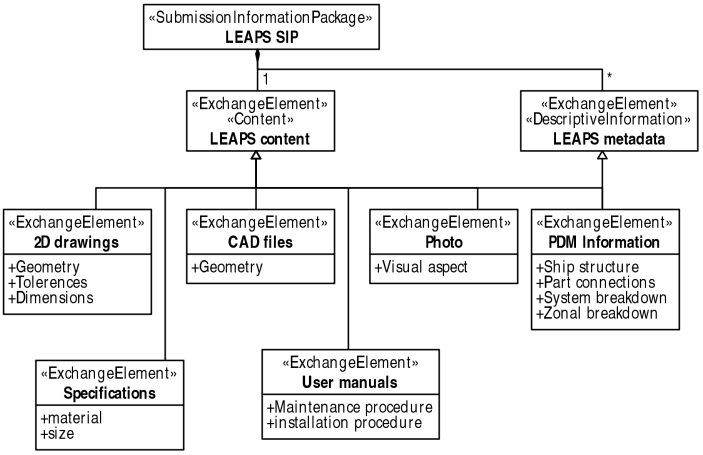

Figure 5.7: DIV-1 view of the LEAPS SIP

Structure of a SIP/DIP The generic structure of the Submission Information Package (SIP) and Dissemination Information Package (DIP) is described in Section 5.2.4.

Conceptual metadata model The notion of metadata may be confusing. Metadata is data about data. In the context of an archival system, metadata can be about documents (files) or about their content. This distinction is important because, while some metadata actually refer to the documents (e.g. checksum or file type), some metadata actually refer to objects defined within documents, and generic solutions don't provide enough granularity to capture this kind of information.

Another confusing aspect is that metadata can be managed separately from the contents, but it can also be a content itself.

As a particular type of information, product models have their own specific metadata, especially metadata to organize the content. Product metadata is important to support

because it will be used to discover product models. A confusion comes from the fact that product models contains product metadata. The question then is what data should be considered as metadata and extracted from the content.

All preserved contents have the same core of metadata. For all the following diagrams, the type of metadata is stereotyped according to what presented in Figure 4.6.

The generic metadata presented in this use case is shown in Figure 5.8. The *LEAPS content* is represented as a *Document*, which is connected to various concepts representing the metadata. The metadata includes author (*Provenance*), format (*Representation*), persistent identifier (*Reference*), label (*Reference*), creation date (*Provenance*), owner (*Provenance*), rights (*Provenance*), and checksum (*Fixity*). This metadata actually relates to files, not to the objects within the files.

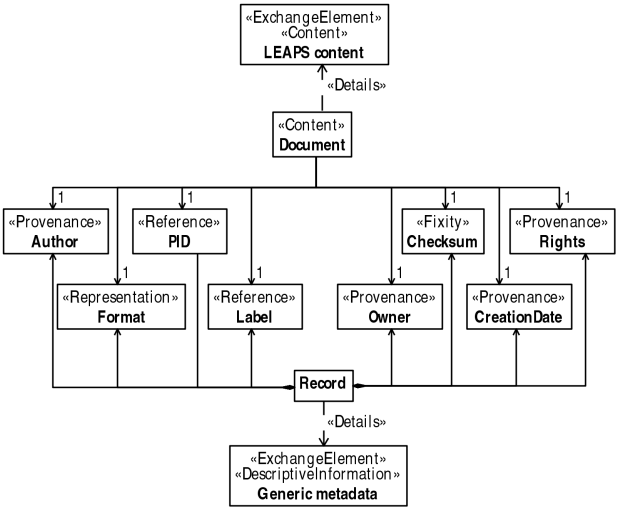

Figure 5.8: DIV-1 view of the generic product models metadata

Product models have particular needs regarding metadata. The following diagrams show, for each kind of content, what information will serve as metadata. For example, any data that may be referred to by external data.

Figure 5.9 shows the content for the *LEAPS file*. As it is PDM information, it can be seen both as content and metadata. The following concepts are presented: parts, connections among parts, system breakdown, compartment breakdown, zonal breakdown, and documents. Systems breakdowns, compartment breakdowns, and zonal breakdowns contain respectively several systems, several compartments, and several design zones. Parts may have several ends, which connect them with each others. A *Document* may be attached to systems, parts, compartments, or design zones.

Figure 5.10 shows the metadata for the CAD document. The CAD file itself is represented, as well as the product it describes. But the CAD file also includes additional *Provenance* information: the design supplier, the design owner, and the creator of the design.

Figure 5.11 shows the metadata for the engineering drawings. Engineering drawings are assigned to a particular location in the ship. The location is represented as a zone breakdown element.

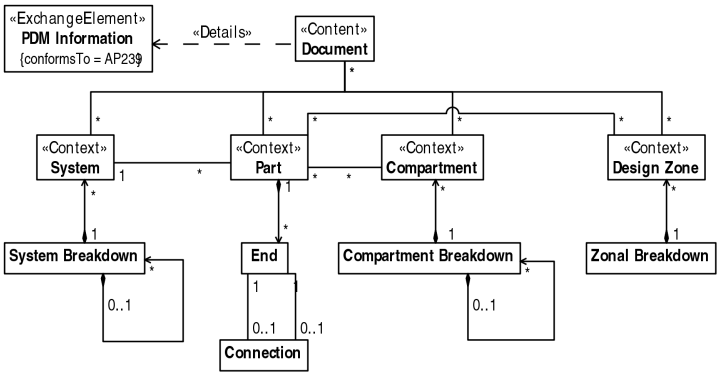

Figure 5.9: DIV-1 view of the conceptual metadata for LEAPS information

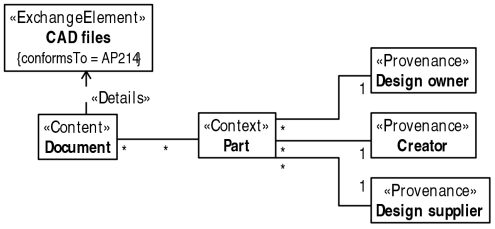

Figure 5.10: DIV-1 view of the conceptual metadata for CAD files

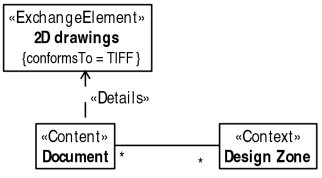

Figure 5.11: DIV-1 view of the conceptual metadata for engineering drawings

Figure 5.12 shows the metadata for the specification. The specification is connected to the actual parts for which it gives the specifications. The metadata also include additional *Provenance* information such as the manufacturer of the part.

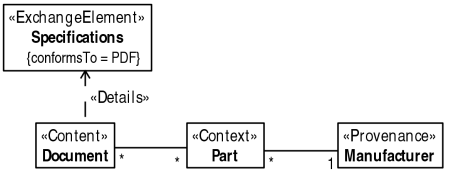

Figure 5.12: DIV-1 view of the conceptual metadata for specifications

Figure 5.13 shows the metadata for the photo. The file is connected to the actual part it depicts.

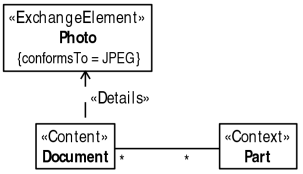

Figure 5.13: DIV-1 view of the conceptual metadata for photos

Figure 5.14 shows the metadata for maintenance procedure. The maintenance procedure is connected to the maintenance activity, which relates to a particular part.

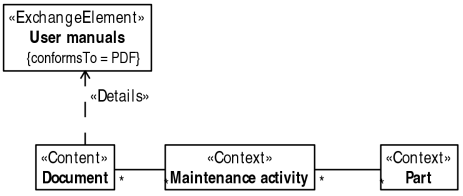

Figure 5.14: DIV-1 view of the conceptual metadata for manuals

Regarding ACTDR certification, these diagrams show explicitly the descriptive metadata[31, Section 4.5.1], including Provenance[31, Section 4.1.4], Reference[31, Section 4.2.4], and Fixity[31, Section 4.4.1.2] information.

Logical metadata model Once the conceptual metadata is defined, it is necessary to identify or develop implementable information model to represent this metadata. These logical models are presented in the following paragraphs.

Dublin Core[65] is a good candidate for representing the generic metadata presented in Figure 5.8. Dublin Core is a widely accepted and used set of metadata for digital resources [67, 68].

Regarding the information model in Figure 5.9, Product Lifecycle Support (PLCS) is chosen to represent product metadata [69]. PLCS is an open Product Lifecycle Management (PLM) standard, which allows the connection of product models created at different

lifecycle stages. PLCS is part of a family of standards called Standard for the Exchange of Product model data (STEP), and it consists of a large information model to cover the entire product lifecycle. In addition to the STEP standardization, the Organization for the Advancement of Structured Information Standards (OASIS) works on the concept of Data EXchange Specification (DEX) to select a subset of this information model that covers a particular data exchange need [70]. In the context of archival systems, the submission agreement between producers and consumers can be defined as a DEX.

The key element of a DEX definition is a template. Whereas STEP defines an information model (static view) for PLCS, the OASIS focuses on the instantiation (dynamic view) of this information model using templates. A template corresponds to a particular instantiation of Application Protocol 239 – Product life cycle support (AP239) entities and relationships. Templates provide a convenient way to express what information is used, as the underlying model is abstracted. Only the name of the template, usually named after the intended behavior of the instantiation, and the necessary parameters are displayed. The logical models in this section are presented in terms of instantiated templates. Many parameters were ignored to improve the readability of the diagrams, but the most important parameters were kept to better explain what information is represented. The actual definition of the templates can be found on the DEXLib website[1].

An important note is that, in some cases, different metadata schema can represent the same kind of information. For example, the owner of a document may be represented both in PLCS and Dublin Core. There needs to be a common agreement on how to represent one kind of metadata. The strategy used here is to represent this information using the most generic metadata schema, so Dublin Core in this example.

In the following figures, the parameters *life_cycle_stage* and *application_domain* are generally assigned to the support stage and the product lifecycle support domain.

Figure 5.15 shows the templates used in the SIP. It is the logical model of the conceptual model presented in Figure 5.9. This information corresponds to a subset of the PDM information that gives *Reference*, *Provenance*, and *Context* information related to products. The presented templates originate from the ShipLTDR DEX, which was develop to support the LEAPS data [2].

The template *representing_part* is called to instantiate parts. The identifier and the version of the part are given. System and zonal breakdowns are respectively instantiated using the *representing_system_breakdown* and *representing_zone_breakdown* templates. These breakdowns have related breakdown elements, instantiated using the *representing_system_element* and *representing_zone_element* templates. Each element can be hierarchically connected to another of its type using the *representing_system_structure* and *representing_zone_structure* templates. The *representing_assembly_structure* template establishes hierarchical relationships among parts. Finally, the connections among the parts are represented as interface connections. A part may contains different interface definitions, instantiated using the *representing_interface_definition* template. A connection, created using the *representing_interface_connection* template, involves two definition occurrences, instantiated using the *representing_interface_occurrence*.

Figure 5.16 shows the metadata for 2D drawings. It is the logical model of the conceptual model presented in Figure 5.11. The *representing_digital_document* template is used

[1] http://www.plcs-resources.org/

[2] http://www.plcs-resources.org/plcs/dexlib/data/busconcept/US_Navy/ShipLTDR/sys/cover.htm

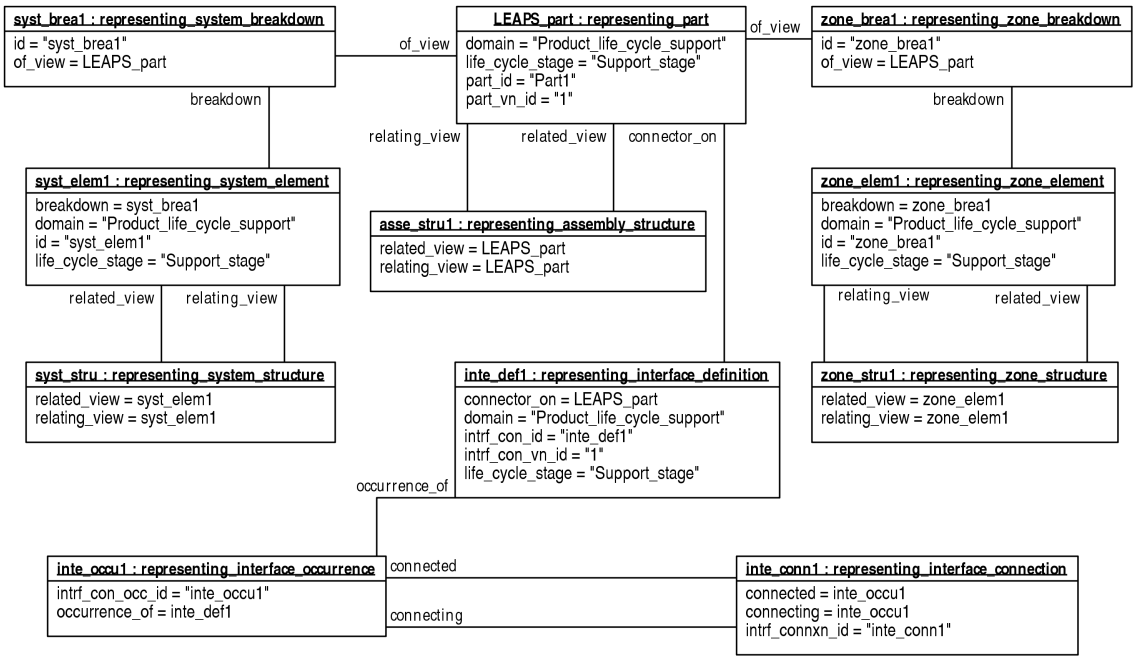

Figure 5.15: DIV-2 view of the PLCS templates used for LEAPS information

to represent the 2D drawing. The document is assigned to a particular zone element using the *assigning_document* template. The category of the document is given by the *assigning_reference_data* template.

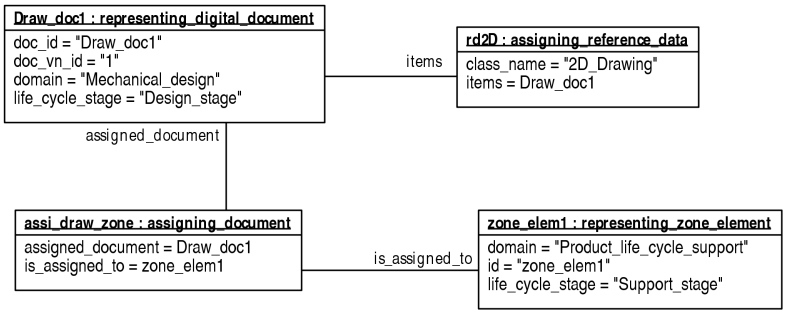

Figure 5.16: DIV-2 view of the PLCS templates for engineering drawings

Figure 5.17 shows the metadata for the CAD files. It is the logical model of the conceptual model presented in Figure 5.10. The CAD file – instantiated by the *representing_digital_document* template – and the part – instantiated using the *representing_part* template – are connected using the *assigning_document* template. Different persons are also associated with the part using the *assigning_person_in_organization* template. These persons have different roles, as defined in the conceptual model: creator, design owner, and design supplier. The category of the document is given by the *assigning_reference_data* template.

Figure 5.18 shows the metadata for the photos. It is the logical model of the conceptual model presented in Figure 5.13. The representation of the photo, its assignment, and its category are created using respectively the *representing_digital_document*, *assigning_*-

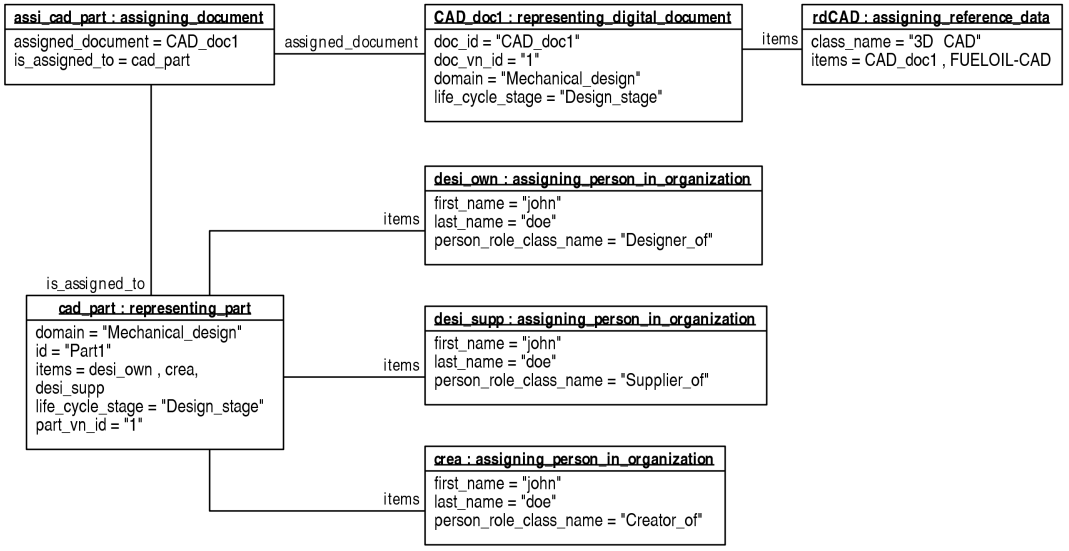

Figure 5.17: DIV-2 view of the PLCS templates used for CAD files

document, and *assigning_reference_data* template. The photo is assigned to a realized part, which is created using the *representing_product_as_realized* template. This template includes as parameter the designed part corresponding to the realized part.

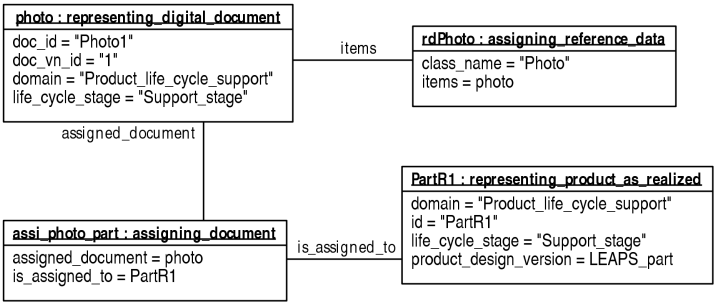

Figure 5.18: DIV-2 view of the PLCS templates for photos

Figure 5.19 shows the metadata for a specification document. It is the logical model of the conceptual model presented in Figure 5.12. The templates used are the same as for photos.

Figure 5.20 shows the metadata for manual information. It is the logical model of the conceptual model presented in Figure 5.14. The representation of the manual, its assignment, and its category are created using respectively the *representing_digital_document*, *assigning_document*, and *assigning_reference_data* template. The manual is then assigned to an activity, instantiated using the *representing_typical_activity* template. This activity represents the maintenance activity of a particular part, which is created using the *representing_part* template.

Then the Physical Data Model (DIV-3) can provide the actual implementation of the schema using a particular language. Note that it is unlikely that one single information model would support any kind of metadata for product model archival system. However, it is necessary to have these metadata schemas integrated so that a query can combine

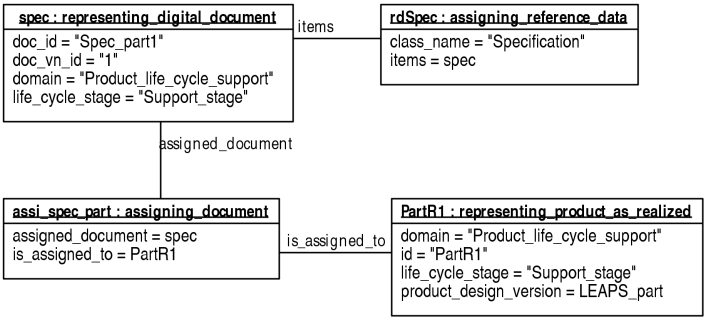

Figure 5.19: DIV-2 view of the PLCS templates for specifications

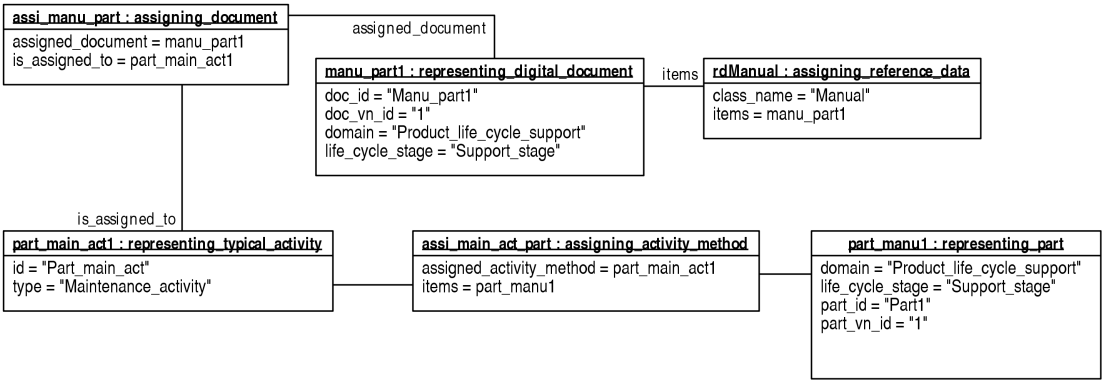

Figure 5.20: DIV-2 view of the PLCS templates for manuals

criteria from different schemas. This is one of the motivations for using the Semantic Web approach to define this metadata. This will be further detailed in the next chapter. In particular, Section 6.4 presents an approach to interconnect the pieces of information present in the product models. This approach is summarized in Figure 6.4

Formats used for the content Figure 5.21 shows the standard formats for each content. The used stereotypes are as defined in Figure 4.15. *Photos* need to conform to the Joint Photographic Experts Group (JPEG) format. *2D drawings* need to conform to the Tag Image File Format (TIFF) format. *User manuals* and *Specifications* need to conform to the Portable Document Format (PDF) format. *PDM Information* needs to conform to the STEP AP239, and *CAD files* need to conform to the STEP Application Protocol 214 – Core data for automotive mechanical design processes (AP214). Part of the verification of the SIP is to make sure the files present conform to the submission agreement. If a file does not conform to this standard, it will be rejected by the archival system. It is necessary to translate the files into accepted formats.

The choice of formats is crucial to ensure the preservation of information. The Library of Congress has proposed seven sustainability factors for digital formats. These factors should be considered for the selection of formats. These factors are disclosure, adoption, transparency, self-documentation, external dependencies, impact of patents, and technical protection mechanism. More details are available in Section 2.3.

As this chapter targets product model preservation and not any other content, the other contents use their native form: photos use the JPEG format, 2D drawings use the TIFF format, and users manuals and specifications use the PDF format.

For product models, the selection of format is in the scope.

CAD files were present in both proprietary and STEP AP214 files. Following the sustainability factors, the proprietary format generally offers low disclosure (specification not available), low transparency (binary files), low self-documentation (binary files), strong external dependencies (proprietary software), unknown impact of patents, and unknown technical protection mechanisms. The adoption is limited to the user of specific software. STEP files offer high disclosure (open formats), high transparency (ASCII or eXtended Markup Language (XML) files), high self-documentation (various metadata available), low external dependencies (supported by multiple software), no impact of patents, and no technical protection mechanisms. However, in practice, proprietary CAD formats are still more used than STEP formats CAD format[18].

From a long-term preservation perspective, it is better to choose STEP formats. But in addition to STEP formats, product models are also converted into Web Ontology Language (OWL) to be saved in addition to STEP. Compared to STEP files, OWL files cannot be directly interpreted by CAD software, but they allow a better self-documentation (semantic representation), and less external dependencies (more tools available for OWL than for STEP).

So, CAD files are stored in STEP AP214 format and OWL format. Similarly, PDM information (both LEAPS content and metadata) is stored in STEP AP239 format and OWL. The translation from STEP to OWL is described in Chapter 6.

Regarding ACTDR certification, these diagrams show explicitly the formats used for the content[31, Section 4.3.2].

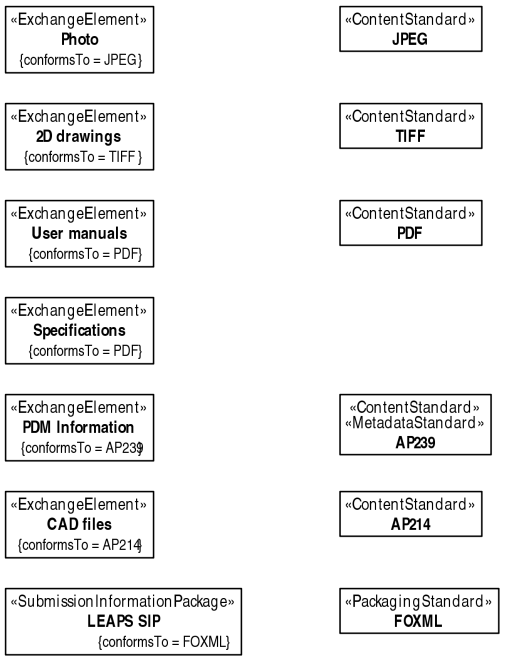

Figure 5.21: StdV-1 view of the formats used in the SIP

Ingest and access activities decomposition The ingest and access activities are decomposed into actions with activity diagrams. These diagrams do not detail how the actions are implemented, but they show what needs to be done. In the maintenance scenario depicted in Figure 5.22, the actions are:

- Issue service bulletin: getting the maintenance plan, and knowing which part has to be checked
- Perform check: getting part location, photo, and maintenance manual from the part information
- Report status
- Determine replacement: accessing part specifications and other catalogs,
- Order part
- Replace part: includes getting the replacement manual

The SIP creation from LEAPS, depicted in Figure 5.23, is not business specific. Indeed, the information had already been created and stored. The data in LEAPS is not well suited for preservation, so the first step is to translate the data into an archival-ready format. Then, the necessary metadata needs to be added. The SIP is created, and sent to the archival system.

Persons involved in the ingest and access activities The ingest and access activities involve different persons, at different posts. Each post requires a certain type of skills for the activity to be performed correctly. Figure 5.24 shows the posts, responsibilities, skills, and activities in focus. The stereotypes are as defined in Figure 4.7 and 4.13. In this case, we consider two posts, associated with two responsibilities. The *Designer*, with a *Producing* responsibility, is in charge of creating the SIP from the LEAPS data. The consumer, with a *Consuming* responsibility, consumes this information during the

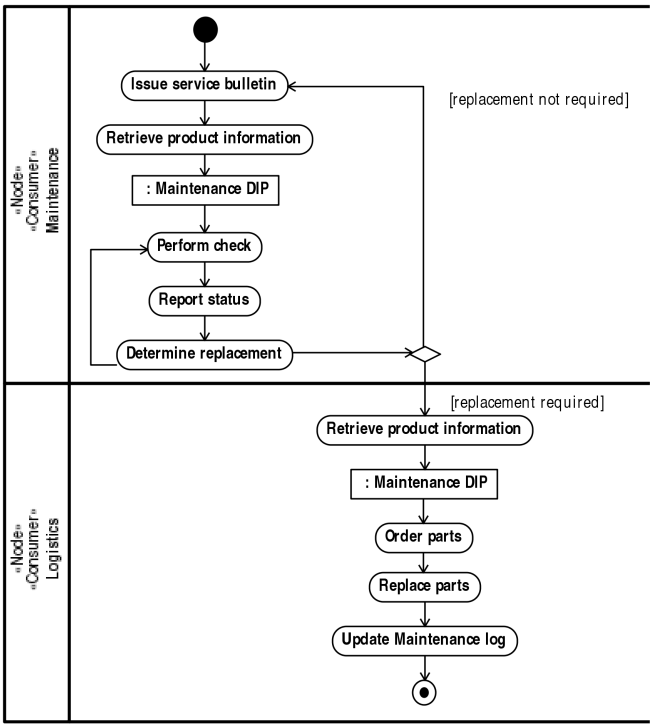

Figure 5.22: OV-5b view for the access activity

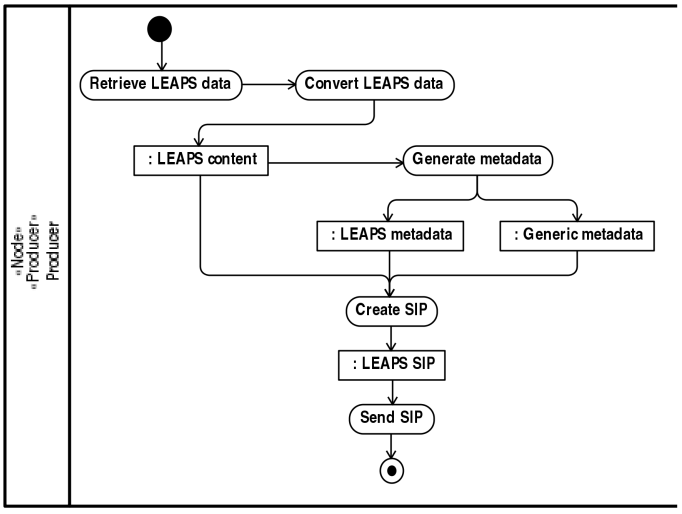

Figure 5.23: OV-5b view for the ingest activity

maintenance activity. These two activities require different competences.

The producer needs to know how to translate files into accepted formats, and needs to make sure that the required metadata is provided when creating a SIP. The producer need to know how to *Create product models* and *Ingest information*. The consumer needs to know how to *Access archive*, and also needs to *Understand product models*.

Regarding the ACTDR certification, this diagram serves to identify the designated community for which product models is intended [31, Section 3.3.1]. This diagram can also represent the knowledge base required for this community to interpret the product models.

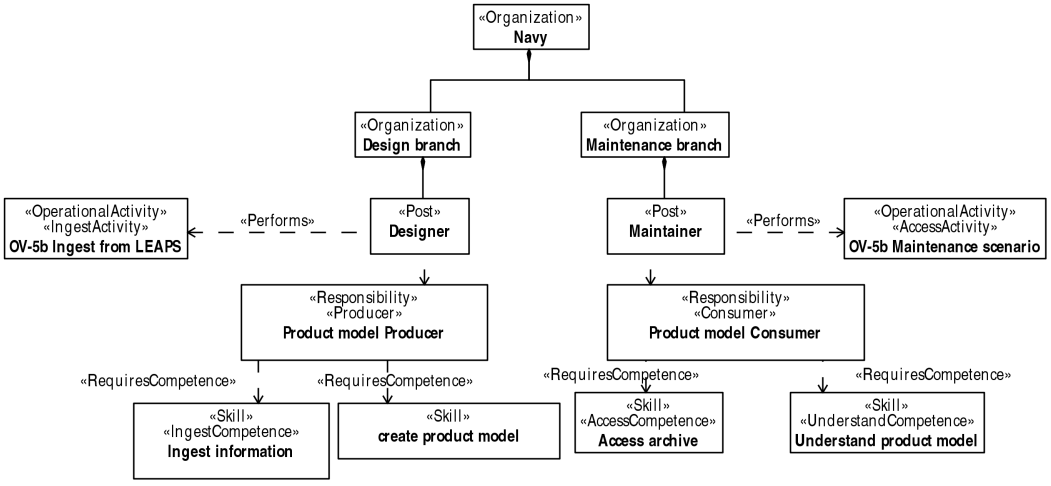

Figure 5.24: OV-4 view for the organization chart (ingest and access)

Various business rules may apply during the ingest and access activities. Figure 5.25 shows an example of rules that apply to SIP and the activities. The constraints are as defined in Figure 4.14. For the ingest activity, a rule states that only the LEAPS producer is allowed to perform the activity. For the consumption activity, similar rules apply for the consumers. A third rule states that the content must be modifiable by the archival system.

Regarding ACTDR certification, these diagrams show explicitly the descriptive metadata[31, Section 4.5.1], including Provenance[31, Section 4.1.4], Reference[31, Section 4.2.4], and Fixity[31, Section 4.4.1.2] information.

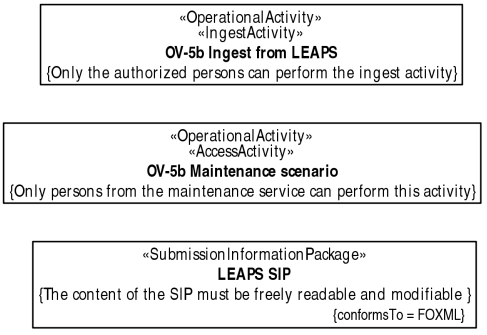

Figure 5.25: OV-6a view for the ingest and access rules

Exchanges among actors The ingest and access activities involve information exchanges among the different actors. These exchanges are depicted in respectively Figure 5.26 and Figure 5.27. The first Figure shows the interaction among the producer, the LEAPS system, and the archival system. The second one shows the interactions between the archival system and the different consumers during the maintenance activity.

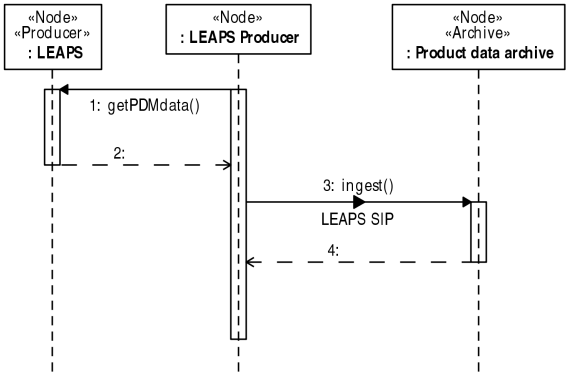

Figure 5.26: OV-6c view for the ingest scenario

5.2.3/ MANAGEMENT ACTIVITIES

Different activities are needed as part of the management of preserved content. Two particular activities are in focus: the migration of the content, which ensures the information stays accessible and understandable to the community, and the disposal activity.

From preservation capabilities to activities The preservation capabilities defined in the Overview part are implemented as specific activities. Figure 5.28 shows how the mapping is realized. The stereotypes used are as defined in Figure 4.9. The migration and disposal activities are shown.

Migration activity Migration is often necessary to preserve the information over time. The migration activity consists in retrieving the information from the archival system, modifying or adding the content or the metadata, and sending the newly created version to the archival system. It is described in Figure 5.29.

The migration activity is more detailed in Figure 5.30, by depicting the exchanges between the preservers and the archival system.

Disposal activity The disposal activity is performed when the preservation of a content is no longer needed. For example, it might be decided to discard product information when the product is no longer in service. Disposal activities need to be documented. A disposal activity consists in locating a particular content, and marking it ready to be disposed.

The disposal activity is detailed in Figure 5.31, by depicting the exchanges between the preservers and the archival system.

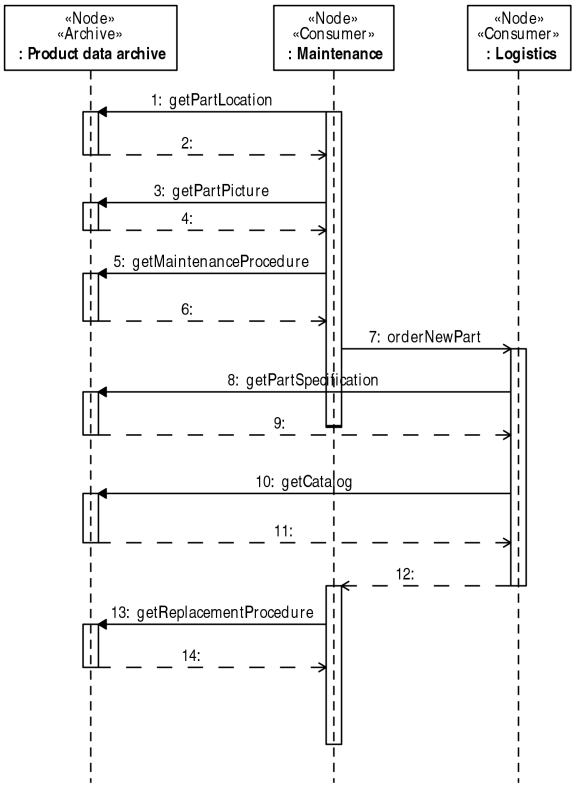

Figure 5.27: OV-6c view for the maintenance activity

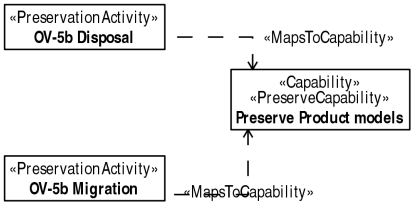

Figure 5.28: CV-6 view of the mapping between capabilities to product model preservation activities

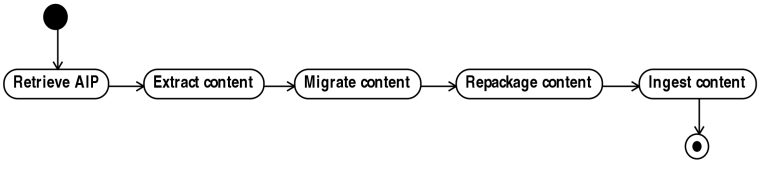

Figure 5.29: OV-5b view of the migration activity

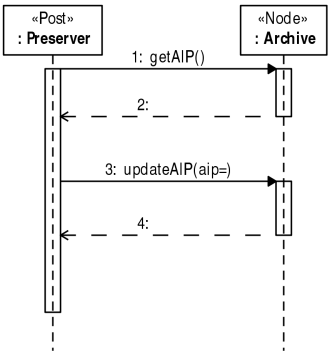

Figure 5.30: OV-6c view of the migration activity

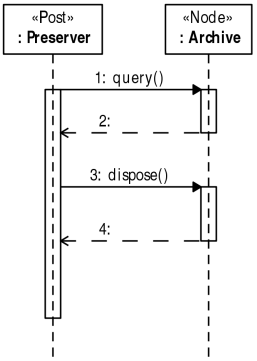

Figure 5.31: OV-6c view of the disposal activity

5.2.4/ PRODUCT MODEL ARCHIVAL SYSTEM AND SERVICES

System, as used in this section, refers to the resources that process automatically the information. Systems are exposed to their environment by services.

Capabilities to services Some capabilities of the archival system are implemented as services. These services are the interface the producers, consumers, and preservers use to ingest, access, and preserve the content. The mapping between capabilities and services is shown in Figure 5.32. Different types of services are proposed for the *Producers*, *Preservers*, and *Consumers*. The *Services* are stereotyped as shown in Figure 4.12.

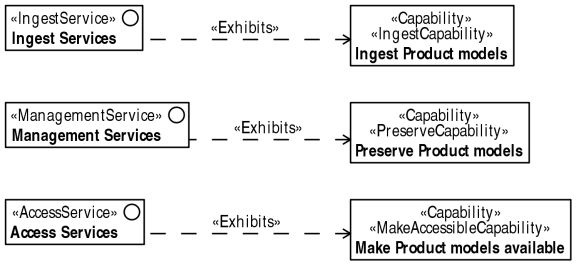

Figure 5.32: CV-7 view for the mapping between capabilities and services

Service definition The services provided by the archival system are shown in Figure 5.33. These are the *IngestService*, *ManagementService*, and *AccessService*. Different services could have been developed for different types of producer and consumers, but this example only represents one producer and one consumer.

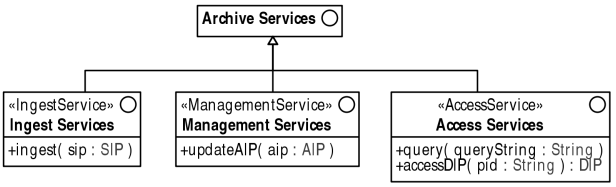

Figure 5.33: Svcv-1 view of the services of the product model archival system

Activities to services The different activities previously identified use the services provided by the archival system. Figure 5.34 shows the mapping between these activities and the services. In this simple example, the ingest activities uses the *IngestService*, the preservation activities use the *ManagementService*, and the access activities use the *AccessService*. The stereotypes used are as defined in Figure 4.12.

Service rules Figure 5.35 shows the rules that apply to the services. Each service is proposed to a certain type of community. The rules define what this community is: for the ingest service, only the LEAPS producer is authorized, for the access service, only maintainers are authorized, and for the management, only preservers are authorized.

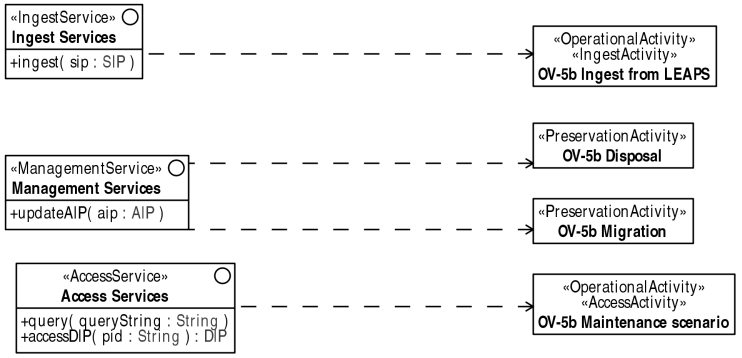

Figure 5.34: SvcV-5 view of the mapping between activities and services

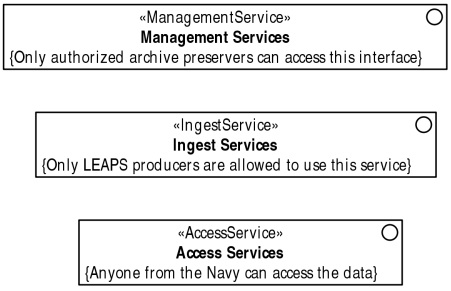

Figure 5.35: SvcV-10 view for the service rules

Regarding ACTDR certification, this diagram shows the access policy for the archival system[31, Section 4.6.1].

System composition The archival system is actually composed of different subsystems that have different tasks. The composition of the archival system is shows in Figure 5.36. It shows an archival software, which is in charge of managing the requests and processing the content, an archival storage software, which is dedicated to the physical preservation of the data, and the data management software, which manages the metadata. An example of archival software is Fedora[3]. Fedora makes it possible to use different archival storage solutions, from a simple file system to a dedicated software such as iRODS[4]. Fedora also proposes different ways to store the descriptive information, such as using a Resource Description Framework (RDF) triple store.

Services to Systems connection The services identified connect to the systems according to Figure 5.37. In this example, the services are connected exclusively to the archival software, which deals with all the requests.

Resource flows among systems The components of the archival system are connected and they exchange information, as seen in Figure 5.38. The SIP is exchanged

[3]http://www.fedora-commons.org/
[4]https://www.irods.org/

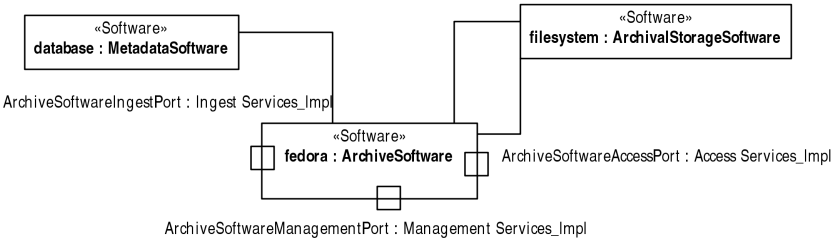

Figure 5.36: SV-1 view of the archival system

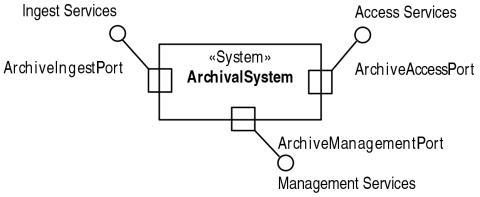

Figure 5.37: SvcV-3a view of the connection between services and systems

between the ingest service and the archival software, then the Archival Information Package (AIP) transits between the archival software and the archival storage software. The archival software sends descriptive information and queries to the data management software, which returns result sets. The archival software receives queries and orders from the access service, and sends back result sets and DIP.

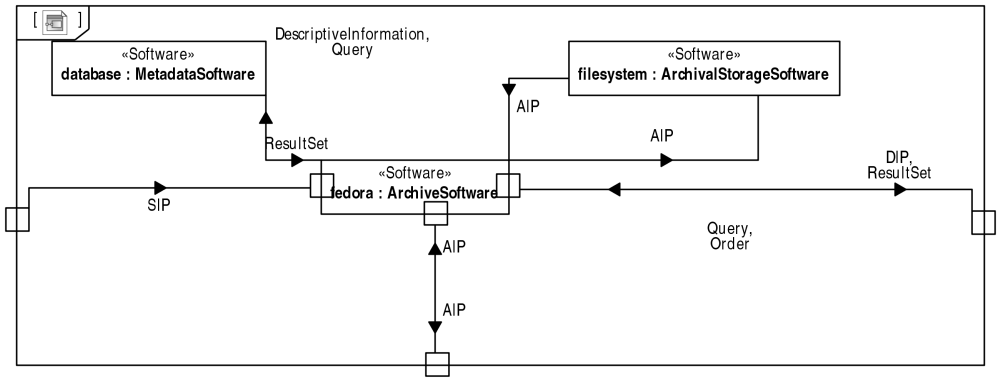

Figure 5.38: SV-2 view of the resource flows in the archival system

System functions Various functions are supported by the archival system. These functions are realized by the different components of the system. The stereotypes used for system functions are as defined in Figure 4.10.

Figure 5.39 shows the ingest function, as implemented by the archive management software. Upon receiving a SIP, the *ReceiveSubmission* function is called. The SIP is then sent to the *GenerateAIP* function, which gives an AIP. From this AIP, the *GenerateDescriptiveInformation* function is called, and the resulting descriptive information and the AIP are given to the *CoordinateUpdates* function.

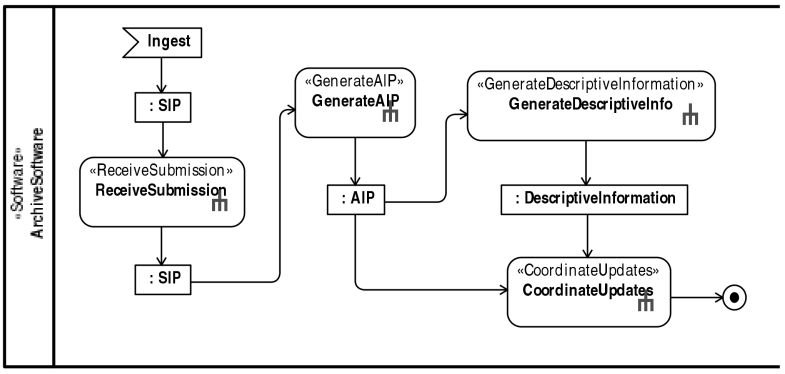

Figure 5.39: SV-4 view of the Ingest function

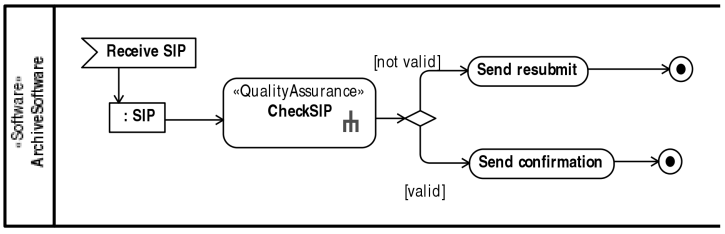

Figure 5.40: SV-4 view of the Receive Submission function

The *ReceiveSubmission* function (see Figure 5.40) gets the SIP, and performs the *checkSIP* function. If the SIP is valid, a confirmation is sent to the producer. If not, a resubmit request is sent.

The *checkSIP* function performs various checks on the SIP to validate it. In the case of the LEAPS SIP, these checks include verifying the SIP integrity, verifying that the content is authorized, verifying that the required metadata is present, and verifying the content checksum against what is declared in the SIP.

The *GenerateAIP* function (see Figure 5.41) adds the missing information to the SIP, converts the files, and repackage the content to create an AIP.

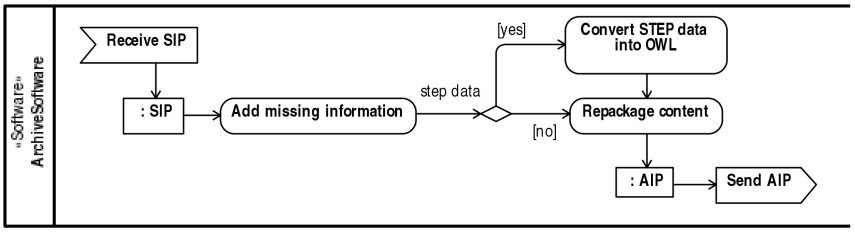

Figure 5.41: SV-4 view of the Generate AIP function

The *GenerateDescriptiveInformation* function (see Figure 5.42) extracts from the AIP the information that will serve as descriptive information. In the case of the LEAPS SIP, the descriptive information is made of Dublin Core metadata and PLCS metadata.

The *CoordinateUpdates* function (see Figure 5.43) receives the AIP and the descriptive information. It sends the AIP to the *ReceiveData* function performed by the *Archival-*

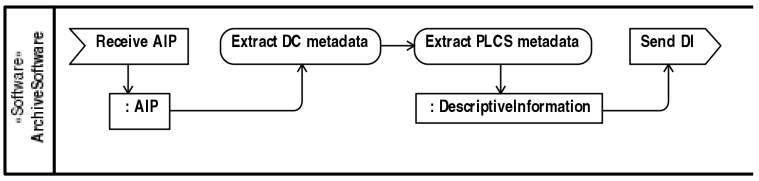

Figure 5.42: SV-4 view of the Generate Descriptive Information function

Storagesoftware. The location of the AIP is then returned, and added to the descriptive information. Descriptive information is then sent to the *MetadataSoftware* which performs the *ReceiveDatabaseUpdates* function.

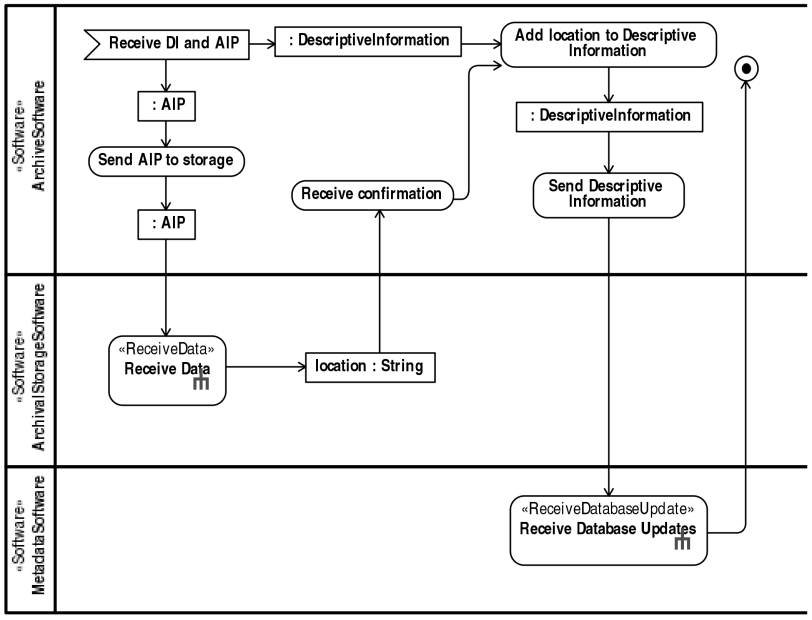

Figure 5.43: SV-4 view of the Co-ordinate updates function

The *ReceiveData* function (see Figure 5.44) receives the AIP, stores it to a location, and sends back this location.

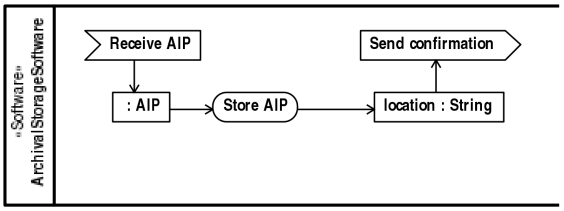

Figure 5.44: SV-4 view of the Receive Data function

The *ReceiveDatabaseUpdate* function (see Figure 5.45) receives the descriptive information, and update the database to include this information.

Figure 5.46 shows the *Access* function. Two types of requests may be received: a query, or an order. The *CoordinateAccessActivities* function manages these requests, and then

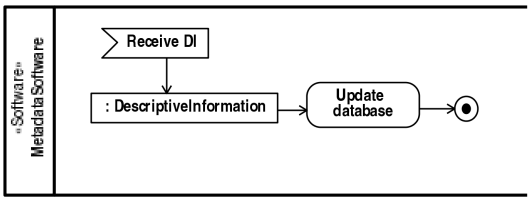

Figure 5.45: SV-4 view of the Receive Database Updates function

the *DeliverResponse* function sends the response back.

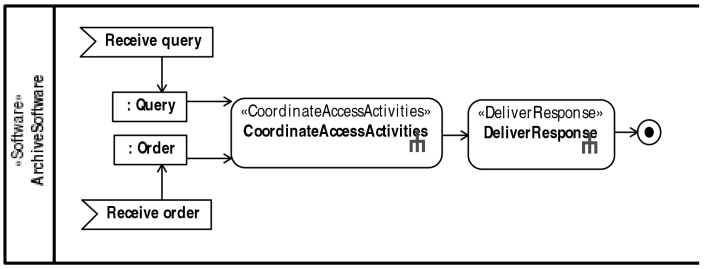

Figure 5.46: SV-4 view of the Access function

The *CoordinateAccessActivities* function (see Figure 5.47) has two different behaviors. When receiving an order, the *GenerateDIP* function is called to product a DIP that is sent back. In the case of a query, the *PerformQuery* function sends a result set back.

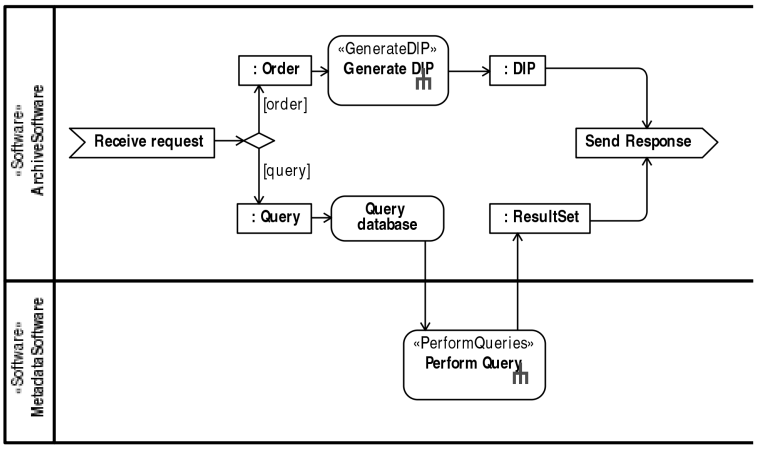

Figure 5.47: SV-4 view of the Co-ordinate Access Activities function

The *GenerateDIP* function (see Figure 5.48) receives an order, and retrieves the AIP through the *ProvideData* function. The wanted datastream is packaged as a DIP and sent back.

The *ProvideData* function (see Figure 5.49) receives the location of the AIP and gives the AIP back.

The *PerformQuery* function queries the metadata database (see Figure 5.50), and generates a result set that is sent back.

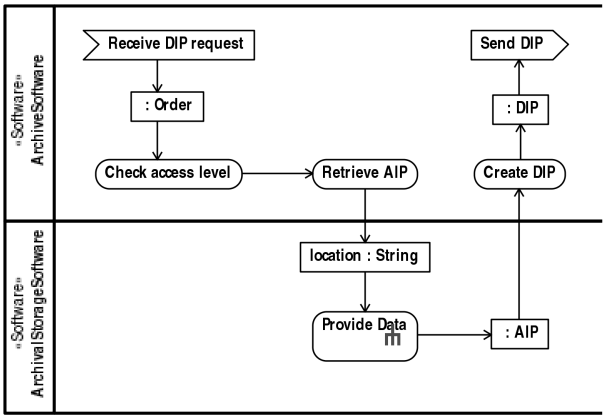

Figure 5.48: SV-4 view of the Generate DIP function

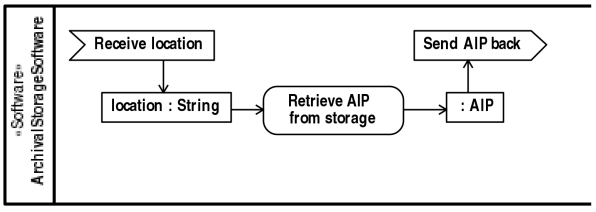

Figure 5.49: SV-4 view of the Provide Data function

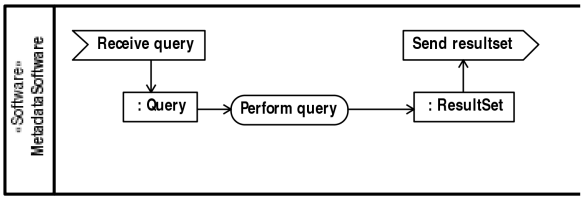

Figure 5.50: SV-4 view of the Perform Queries function

The *DeliverResponse* function (see Figure 5.51) sends either a result set or a DIP depending on the kind of query.

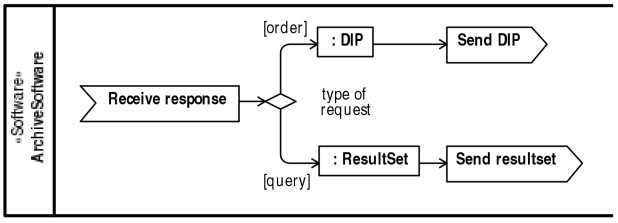

Figure 5.51: SV-4 view of the Deliver Response function

The physical exchanges among the systems are represented as sequence diagrams. Figures 5.52, 5.53, 5.54, and 5.55 show the interactions that occur during an ingest, a modification, a query, and an order respectively.

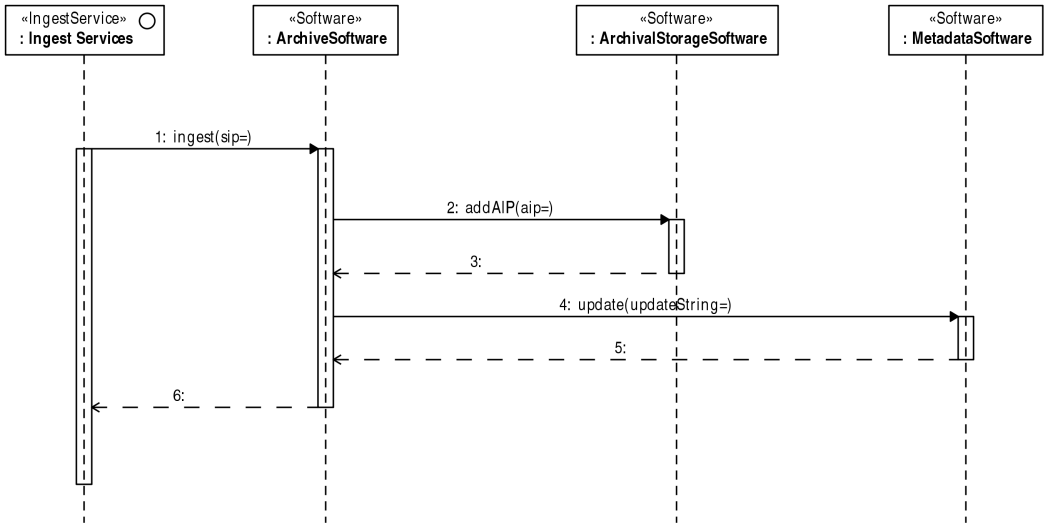

Figure 5.52: SV-10c view of the Ingest sequence

Regarding ACTDR certification, these diagrams intend to document various aspects of the archival system: SIP verification [31, Section 4.1.5], AIP generation [31, Section 4.2.2], AIP disposal [31, Section 4.2.3], Error checking [31, Sections 4.2.8, 4.2.9, 4.4.1.2], Receive data [31, Section 4.4.1], Generate DIP [31, Section 4.6.2].

Metadata schema The metadata schema is presented in Figures 5.56 and 5.57. The former represents generic metadata, and the latter represents product-specific metadata. One metadata schema is not sufficient to effectively support every kind of content. Specific types of content often require specific types of metadata. When the same kind of metadata is available in different schemas, the most generic should be used in priority. This is to provide a common way of representing one kind of metadata, and facilitate the future use. The first figure shows the Dublin Core, which contains generic metadata for digital resources. The second shows the PLM data, which serves as product metadata in this example. PLCS templates are used to simplify the view of the schema.

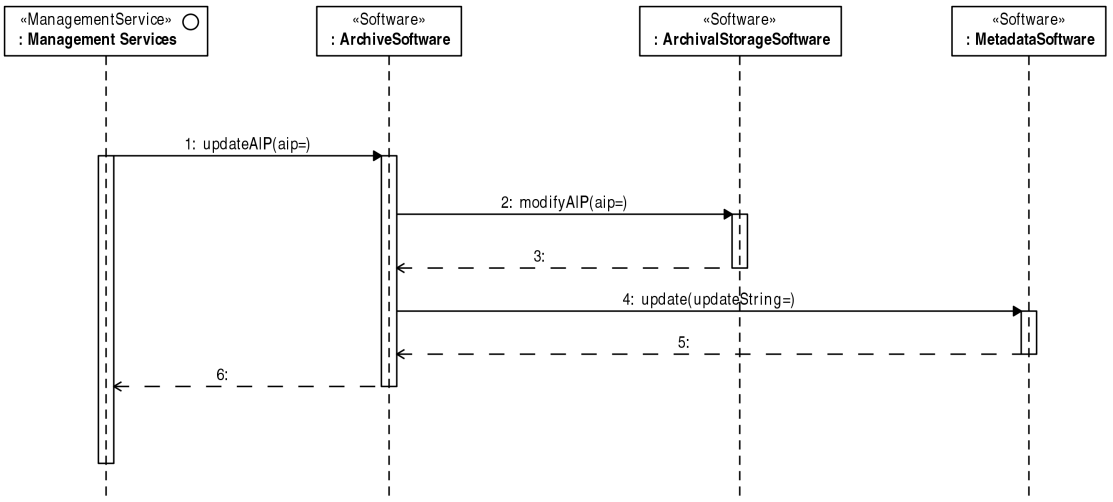

Figure 5.53: SV-10c view of the Modification sequence

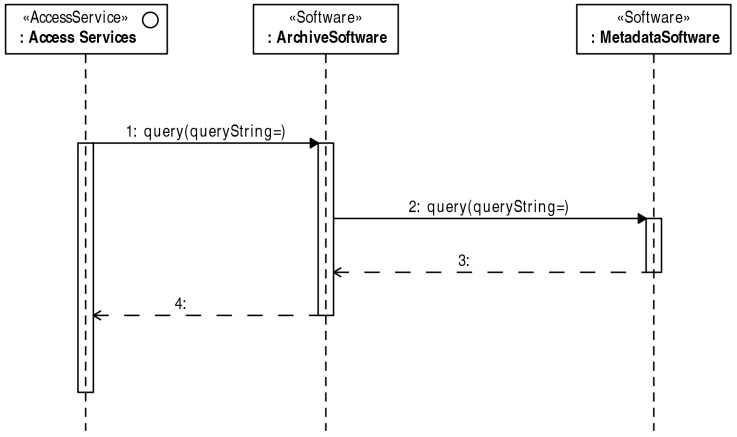

Figure 5.54: SV-10c view of the Query sequence

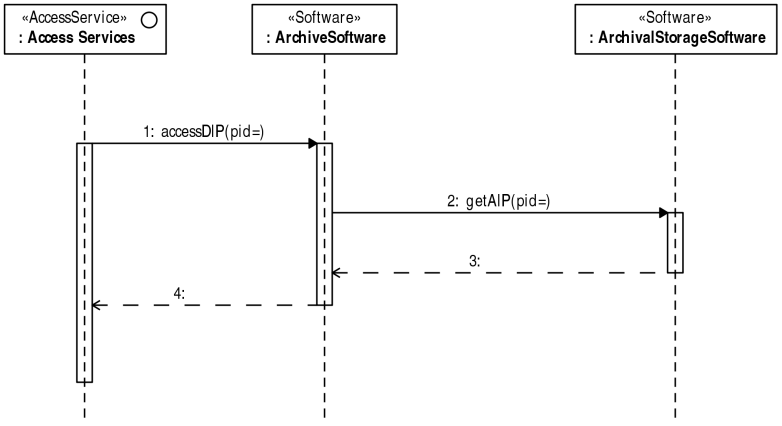

Figure 5.55: SV-10c view of the Order sequence

The product-specific metadata schema integrates the different metadata previously identified. Documents are represented using the *representing_digital_document* template, and assigned to other elements using the *assigning_document* template. In this schema, the documents may be assigned to designed parts – created using the *representing_part* template –, concrete parts – created using the *representing_product_as_realized* –, locations – created using the *representing_zone_element* –, and activities – created using the *representing_typical_activity* –. System elements and zone elements belong respectively to system and zonal breakdowns, instantiated using the *representing_system_breakdown* and *representing_zone_breakdown* templates. A hierarchy among these elements can be defined using the *representing_system_structure* and *representing_zone_structure* templates. The *representing_assembly_structure* template is used to create a hierarchy of parts. A part may have interface definitions, instantiated using the *representing_interface_definition* template. These interface definitions can be connected using the *representing_interface_connection* template, which relates two occurrences of interface definitions. These interface occurrences are created using the *representing_interface_occurrence* template.

Figure 5.56: DIV-2 view of the generic metadata schema

Information package structure The structure of the information packages is shown in Figure 5.58. The structure used is the Fedora Object Model. The central concept is the *DigitalObject*. A *DigitalObject* has for attributes the version of the model being used, a Persistent Identifier (PID) that identifies the digital object, and the acuri of the Fedora repository. A digital object may have various properties (name/value structures). Predefined properties are the state, the label, the creation date, the last modification date, and the Identifier (ID) of the owner of the object. A digital object contains various *Datastreams*. A *Datastream* represents files associated with the digital object: for example, it can be a representation of the digital object, or it can be metadata for this object. A datastream has an ID attribute, a control group attribute that indicates the type of datastream (metadata, content managed by the archival system, content managed by an external archival system, or redirection to an external archival system), the Uniform Resource Identifier (URI) of the Fedora repository, and a state attribute that indicate whether the object is being managed or not. Each datastream may have different versions, represented by a *DatastreamVersion*. A *DatastreamVersion* has most of the attributes usually associated with files: an ID, a label, a creation date, a Multipurpose Internet Mail Extensions (MIME) type, alternative IDs, a URI that indicate what format is used, and the size of the file. Finally, each *DatastreamVersion* has a content, which can be either embedded XML content, embedded binary content, or external content.

Regarding the ACTDR evidence, this diagram shows how the SIP can be parsed[31,

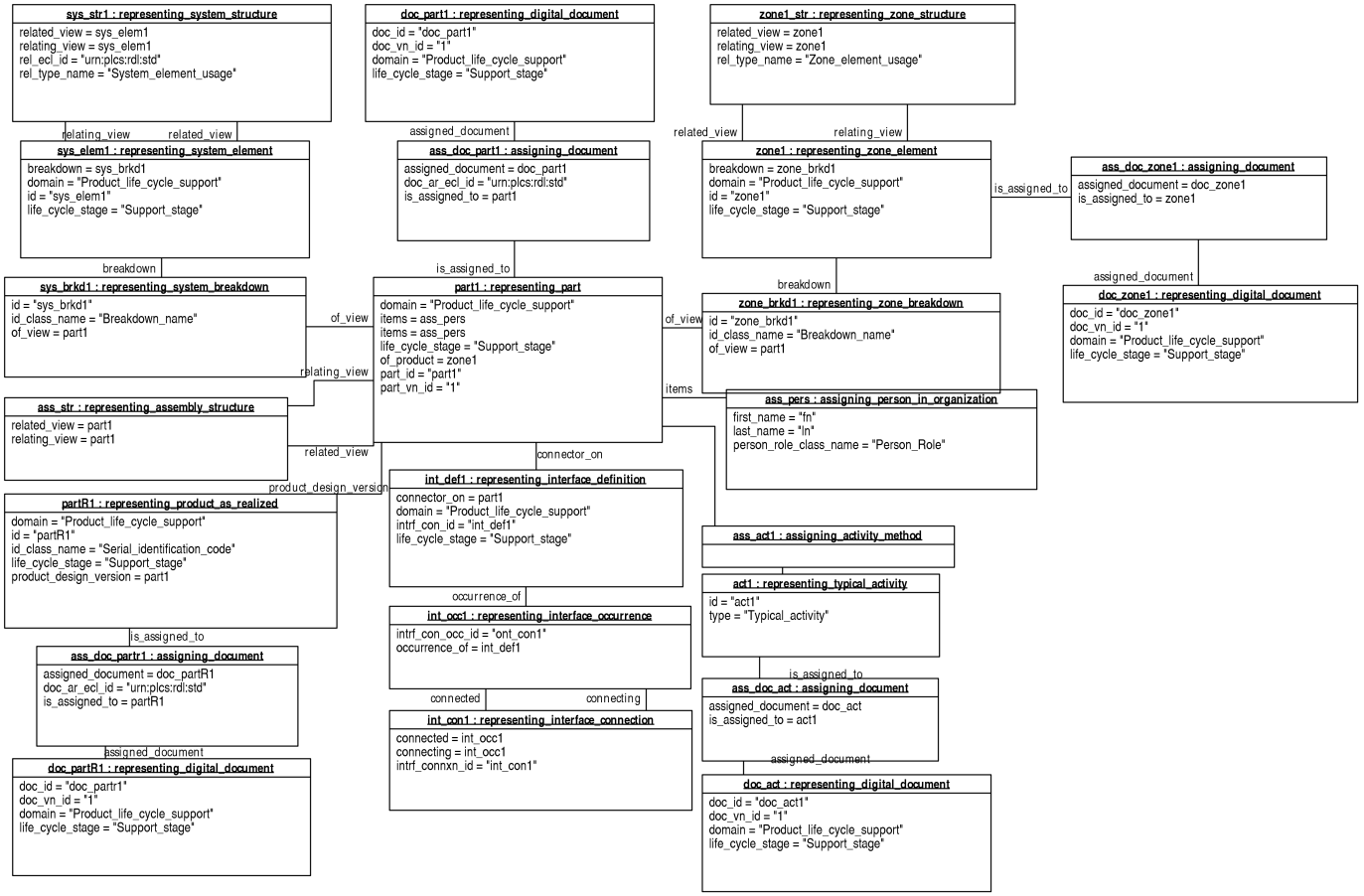

Figure 5.57: DIV-2 view of the product-specific metadata schema

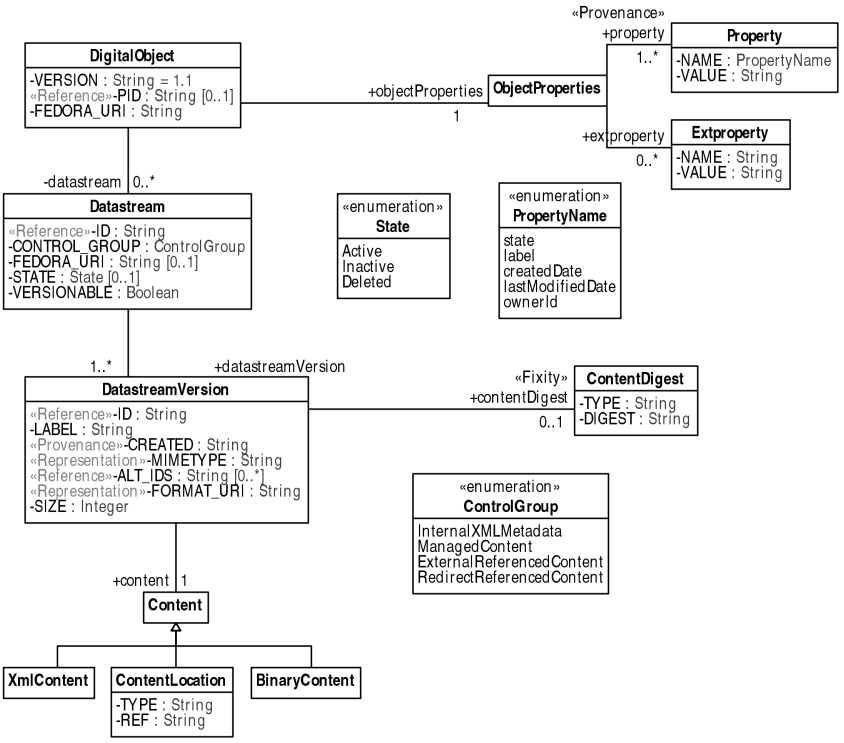

Figure 5.58: DIV-2 view of the information package structure

Section 4.1.3].

5.3/ CONCLUSIONS

This chapter presented an application of the Reference Architecture for Archival Systems described in Chapter 4 to describe an archival systems for product models. The resulting description used RAAS by applying all the proposed concepts and views to product models preservation. Moreover, parts of the formal description were implemented to demonstrate the feasibility of the overall approach. The objective of the application was to obtain a formal description that:

- uses the terminology defined in the OAIS RM, to provide a better understanding of the architecture from a preservation perspective
- includes views that demonstrate the ability of the described archival system to preserve information, according to the ACTDR
- formally describe the content, metadata, activities, system functions, and services involved in the product model preservation

Developing a complete software according to the architectural description is beyond the scope of this thesis. However, some parts of the description were implemented. Section A.5 provides additional information on how the ingest activity was implemented for a particular part: a gate valve.

The next section addresses another issue of digital preservation: the representation of preserved information. This section proposes a semantic representation for product mod-

els, to improve the search and the understanding of product information.

6

SEMANTIC REPRESENTATION OF PRODUCT MODELS

As explained in Chapter 1, the choice of a format for representing information is important. Such format should make it possible to describe the information so that it can be interpreted and understood over time. From a short-term perspective, having a format that has a good software support is important, because it allows user to directly consume the information. From a long-term perspective, different characteristics need to be considered. For example, formats that are well documented and have few external dependencies make it easier to develop software that translate the data into a new format, or simply render the data in a human-readable way.

For product models, international standards such as the Standard for the Exchange of Product model data (STEP) have given solutions to ensure that there will be a well-documented solution to represent the information. STEP defines information models and file formats that together provide a credible solution for long-term preservation. For this reason, STEP was selected in various sectors, such as in the automotive sector[34] and in the aerospace sector[71], as the format for the long-term preservation of product models. However, STEP still has some limitations regarding preservation and accessibility. The main one is the inability of the format to establish semantic relationships with external knowledge, such as other concepts or objects. Existing workaround are to use additional mechanisms, which are external dependencies. Referring to external concepts allows the use of a specific vocabulary that is not supported by the original generic information model, such as domain-specific terms. These two features are beneficial to the preservation for two reasons: humans will have more chance of understanding the data because more context is given, computers can leverage these features to discover preserved information more easily (data integration).

This chapter presents the benefits of the translation of STEP data models into Web Ontology Language (OWL). OWL is a knowledge representation language based on description logic [72], and it supports reasoning over the information. OWL was created in the context of semantic web, to give more semantic meaning to the information available on the Internet. In OWL, every entities is identified by a Uniform Resource Identifier (URI), so that it can be referenced by external entities. The translation from STEP to OWL is called OntoSTEP, and makes it possible to semantically enrich product models.

Product models can greatly benefit from semantically enriched representations. Many different product models may be used to represent a same complex product, and objects representing the same thing may be present across all these models. For example, a product model may represent the functional requirements of a part, another prod-

uct model may represent the geometry of this part, and yet another product model may represent manufacturing information regarding the same part. The relationships among product models are currently managed by specific systems, such as Product Data Management (PDM) systems, but these relationships are not available through the language. Including such relationships without relying on a dedicated system reduce the number of technology necessary to locate information.

As product information is integrated, it can be easier to discover the different type of metadata. For example, if metadata representing Representation, Provenance, or Context information is associated to a product in one product model, this information can be discovered in another product model thanks to the semantic connection.

Another major issue with product models is that the information models are usually too generic to support domain-specific terms. To import the knowledge associated with these terms, specific agreements and software have to be developed. With OWL, it is possible to directly classify any object using externally-defined ontologies. The associated knowledge present in the ontologies can be imported. This mechanism eases the preservation of information over time, as the knowledge is represented in a better way.

This chapter is organized as follows: Section 6.1 presents the connection between the semantic web technologies and preservation. Section 6.2 provides an evaluation of the semantic representation for product models. Section 6.3 presents OntoSTEP, the translation from STEP to OWL. Section 6.4 presents a first benefit of the semantic representation: product model integration. Section 6.5 presents a second benefit of the semantic representation: the use of external domain-specific vocabularies to enrich product models.

6.1/ On semantic web and preservation

Semantic web is defined as “an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation” [73]. Semantic web aims to overcome three issues: the lack of semantic definitions in individual systems, the lack of semantic integration among data sets, and the lack of semantic interoperability across disparate systems [74]. In practice, the semantic web does not aim at replacing the existing way of disseminating the information. However, this current way is more thought with human processing in mind. The semantic web aims to add the tools to allow computers to process the information.

The semantic web relies on the following languages, developed by the World Wide Web Consortium (W3C). eXtended Markup Language (XML) is a language that describes not only the data but also the structure, using “tags”. Resource Description Framework (RDF) has been created as a mean to link or associate information. RDF relies on triples composed of a subject, an association, and a target. Finally, OWL provides the ability to reason over distributed data.

Used together, these technologies allows the interconnection of dispersed information and the discovery of new knowledge through reasoning. These connections are directly established as part of the content. Semantic web does not mean that the entire content should be represented in OWL, but just the “useful” part of it, such as metadata for instance.

The Web and archives share many similarities, at different scales.

The information is generally represented for human consumption. In the product data world, scanned engineering drawings is a good example. Computers process them as images, and not as what they represent for humans. A clear drawback is the inability to leverage computers to process this information, and doing search for instance. A way to overcome this problem is to attach a formal representation that computer can process.

A key to enable fast discovery is not only to have a formal representation, but also to be able to interconnect this information. To do so, the language used to represent this information has to allow the establishment of these interconnections.

This ability to connect information allows to define this information in a semantically stronger way. Section 6.4 shows how disaggregated product information is integrated, thus enabling an easy navigation through the different occurrence of a same thing. Section 6.5 showed how to use the semantic representation to define product data more precisely by using concepts defined in a shared ontology, and hence enabling semantic queries.

The archive benefits from a semantic representation, as it makes the information better defined, and more accessible. So, a semantic representation offers more guarantees that the content will be accessible and understandable over time.

6.2/ EVALUATION OF THE SEMANTIC REPRESENTATION FOR PRESERVATION

The evaluation of OWL for preservation is based on seven sustainability factors defined by the Library of Congress for digital formats[75]. The OWL representation is evaluated against STEP standards and proprietary formats. These factors are disclosure, adoption, transparency, self-documentation, external dependencies, impact of patents, and technical protection mechanism.

Proprietary formats are usually characterized by:

- low disclosure: information about the format are not always available,
- low transparency: use of binary formats,
- low self-documentation: the file itself offers few clue on how to interpret the data,
- strong external dependencies: only specific proprietary software on specific platform can interpret the data,
- unknown impact of patents,
- unknown technical protection mechanisms.

All these aspects are barrier to the long-term preservation of product models, because they don't give preservers enough documentation and the independence to manipulate the data.

On the other side, STEP formats are characterized by:

- high disclosure: the formats are accessible to anyone,
- high transparency: the data is stored in ASCII or XML, and it is understandable by humans,
- high self-documentation: various metadata are available,

- low external dependencies: formats are supported by multiple software,
- no impact of patents,
- no technical protection mechanisms.

The only drawback of STEP formats are that new features take more time to be standardized, so they sometimes lag behind proprietary formats.

The OWL representation of product models offers almost the same characteristics as STEP representation, with only a few differences. On one side, the OWL representation cannot be directly interpreted by existing software. This is a drawback from a short-term perspective. On the other side, the OWL representation directly provides features (such as data integration or use of a controlled vocabulary) that would otherwise require external mechanisms. As a result, the OWL representation provides a better self-documentation by allowing the semantic enrichment, and less external dependencies to understand the data.

In conclusion, the OWL representation can constitute a good complement to the STEP representation.

6.3/ OntoSTEP: translating STEP data models into OWL

This section describes how STEP data is translated into OWL, to provide the benefits presented in Chapter 6. STEP data has to conform to an information model described in the EXPRESS language. The information model gives semantics of the data and presents how this data should be structured. In the STEP standard, information models are published as Application Protocol (AP). For example, the Application Protocol 203 – Configuration controlled 3D design of mechanical parts and assemblies (AP203) defines the information model to represent 3D geometry. The STEP data associated with the AP203 would be a CAD file containing the 3D representation of a product.

Both information model and data have to be translated into OWL. This section presents the mapping from EXPRESS information models — also called schemas — to OWL. The resulting concepts and relationships constitute the Terminology Box (TBox). Then, it presents the mapping from STEP data to OWL. The resulting data constitutes the Assertional Box (ABox).

The *Courier New* font denotes EXPRESS code, and *Arial Narrow* font denotes OWL code.

6.3.1/ Mapping the information model

The following example, extracted from AP203, illustrates our translation of the main EXPRESS concepts.

```
ENTITY product_category;
name        : label;
description : OPTIONAL text;
END_ENTITY; -- product_category

ENTITY product_related_product_category
```

```
SUBTYPE OF (product_category);
products : SET [1:?] OF product;
END_ENTITY; -- product_related_product_category

ENTITY product;
id                  : identifier;
name                : label;
description         : text;

END_ENTITY; -- product
```

In this example three entities are described: *product*, *product_category*, and *product_related_product_category*. A product has an *identifier*, a *name* and a *description*. A *product_category* contains a *name* and may have a *description*. A *product_related_product_category* is a *product_category* that identifies the products that satisfy the type identified by the category. To simplify this example, some parts of the actual AP203 entity definitions have been removed.

The concept of entity in EXPRESS is similar to the concept of a class in object-oriented modeling: entities can be seen as abstractions of real-world objects (instances) and can be organized in hierarchies. These hierarchies conform to the following inheritance principle: sub-entities (*product_related_product_category* in our example) inherit the attributes of their super-entities (*product_category* in our example) and the instances of the former are also instances of the latter. Attributes specify relationships between entities or between entities and data types. An attribute consists of a name and a type: in our example, the first attribute of the entity product is called *id*, and its type is *identifier*. An attribute may be optional, as in the case of *description* in *product_category*, and its type can be a collection of data, as in the case of products in *product_related_product_category*.

In our translation, EXPRESS entities and instances map respectively to OWL classes and individuals. Attributes correspond to OWL properties — *ObjectProperties* link classes together, while *DataProperties* link classes to data types. The domain of a property defines which classes can have this property. Without restrictions, properties in OWL are aggregations, so an individual can be linked to several individuals by using the same property. To define the usage of a property, it is possible to restrict its cardinality through the *ObjectExactCardinality* construct and its values through the *ObjectAllValuesFrom* construct. In the case of an optional attribute, the *ObjectAllValuesFrom* construct is used to link the entity to the union of the attribute type and the class *owl:Nothing*. This solution is adopted to explicitly express the semantics of the *OPTIONAL* keyword: a value is not required for this attribute.

An ontology may contain statements related to both classes (TBox – terminological box) and individuals (ABox – assertional box). In our approach, an EXPRESS schema is translated into an ontology that contains mainly classes and property definitions [76]. Figure 6.1 summarizes the proposed OWL mapping of the basic concepts of EXPRESS.

The naming conventions for the properties were redefined. Consider, as an example, the entities *product* and *product_category*, both having the name attribute. In EXPRESS attributes are defined to be within the scope of the entity. In OWL properties have a global scope, so the property name would be the same for the *product* and the *product_category*. Attribute names are prefixed with the entity names to differentiate attributes. So, the entities *product* and *product_category* will contain, respectively, the attributes *product_*-

EXPRESS	OWL
Schema	Ontology
Entity	Class
Subtype of	Subclass of
Attribute with an entity type	*ObjectProperty*. The domain of the property is the class that corresponds to the entity that contains the attribute. This class is restricted to have *ObjectExactCardinality* equal to 1 and *ObjectAllValuesFrom* equal to the entity type for that property.
Attribute with a simple data type	*DataProperty*. The domain of the property is the class that corresponds to the entity that contains the attribute. This class is restricted to have *ObjectExactCardinality* equal to 1 and *ObjectAllValuesFrom* equal to the data type for that property.

Table 6.1: Mapping of the basic concepts from EXPRESS to OWL

has_name and *product_category_has_name*. The following OWL statements, expressed in functional syntax, are the translation of the previous entity definitions:

```
SubClassOf(product ObjectAllValuesFrom(
  product_has_description text))
SubClassOf(product ObjectExactCardinality(1
  product_has_description))

SubClassOf(product ObjectAllValuesFrom(
  product_has_name label))
SubClassOf(product ObjectExactCardinality(1
  product_has_name))

SubClassOf(product ObjectAllValuesFrom(
  product_has_id identifier))
SubClassOf(product ObjectExactCardinality(1
  product_has_id))

SubClassOf(product_related_product_category
ObjectAllValuesFrom(
  product_related_product_category_has_products
  product))

SubClassOf(product_related_product_category
  product_category)
SubClassOf(product_related_product_category
  ObjectMinCardinality(1
  product_related_product_category_has_products
  ))
```

```
SubClassOf(product_category ObjectAllValuesFrom(
  product_category_has_name label))
SubClassOf(product_category
  ObjectExactCardinality( 1
  product_category_has_name))

SubClassOf(product_category ObjectAllValuesFrom(
  product_category_has_description
  ObjectUnionOf(owl:Nothing text)))
SubClassOf(product_category
  ObjectExactCardinality(1
  product_category_has_description))
```

Some EXPRESS constructs, such as functions, cannot be translated without some additional efforts: these constructs usually define entity constraints and attributes computation, and may rely on complex algorithms. OWL, as it is based on Description Logic, does not contain any procedural aspects. The next paragraphs focus on the EXPRESS language constructs (e.g. data types, bags, select, enumerations, abstract entities and inheritance) that can be automatically translated into OWL.

Data types EXPRESS includes all the data types (such as *integer*, *boolean*, *string*) required to capture product information. OWL inherits the data types defined in the XML Schema Definition (XSD) language. Some EXPRESS types, e.g., *Boolean* and *String*, have an exact equivalent in OWL, while other types, e.g., *Number* and *Real*, are represented in a slightly different way in OWL. For example, the EXPRESS *Real* type corresponds to *double* in OWL, even though the precision of those two data types is different.

EXPRESS allows the derivation of data types from simple types. A type hierarchy is constructed in OWL to deal with these derived types, we build a type hierarchy and apply the concept of data wrapping.

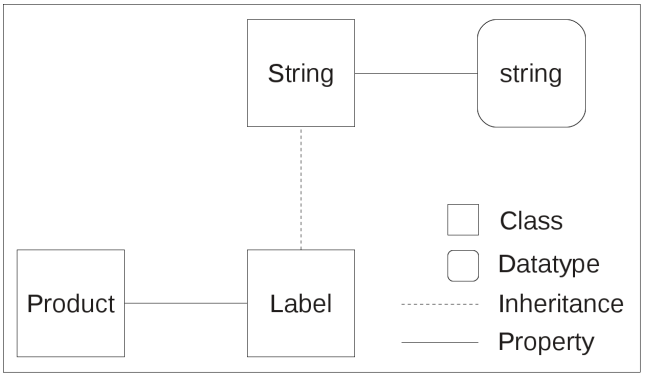

Figure 6.1: Representation of EXPRESS attributes in OWL

In the example in Figure 6.1, we define a class *String* that has a *DataProperty* relation to the string data type. It is then sufficient to subclass the class *String* to translate all the user-defined data types (*Label* in this case) derived from *String*. Because of the possible use of functions, we cannot guarantee an automatic translation of data type restrictions.

Using a manual case-by-case translation, most of the types defined in AP203 can be translated.

Aggregations EXPRESS provides four different kinds of aggregations: *set*, *bag*, *list*, and *array*. Each of these aggregations has order and duplication policies. When an actual aggregation is used in a schema, the type of its content and the number of elements it shall have are defined. The detailed mapping of aggregations is explained in [77]. Here, as an example, we provide the detailed mapping of bag.

Bags are unsorted collections of elements. The only difference between sets and bags is the duplication policy: the same element can be repeated several times in a bag. As object properties in OWL do not allow duplications, we create the structure shown in Figure 6.2 to correctly map EXPRESS bags to OWL. Consider a bag called *Container*, which contains items of type *Content*. To correctly map this bag to OWL, we create two classes, (*Container* and *Bag*), and two properties (*hasBag* and *hasContent*).

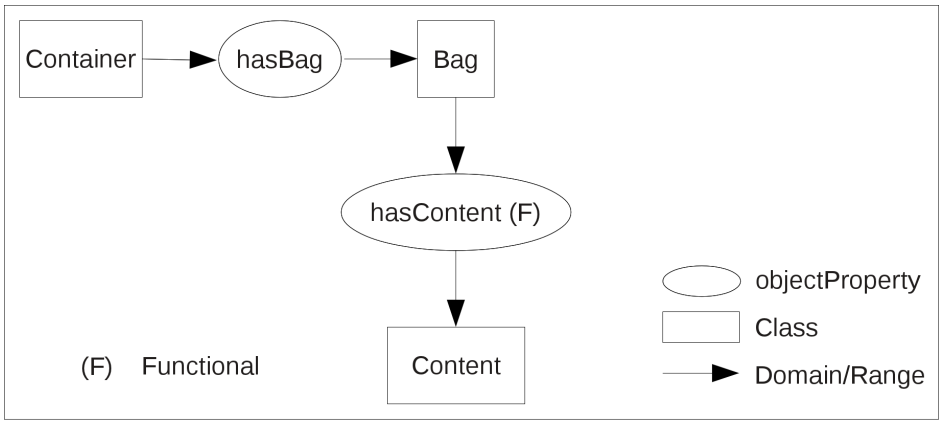

Figure 6.2: Representation of a Bag in OWL (TBox)

The class *Bag* is used to represent an occurrence of *Content* using the property *hasContent*. The property *hasContent* is declared as *Functional*, which means that there can be only one instance of *Content* linked to an instance of *Bag* by the property *hasContent*. Figure 6.3 represents the instantiation of the schema presented in Figure 6.2, showing how an EXPRESS bag containing a duplicated element is converted to OWL. An instance of *Container* (cont) is linked to two different instances of the *Bag* class (*b1* and *b2*). Each of these two instances is then linked to the same instance of the *Content* class (*elem1)*. Since *b1* and *b2* are different, the ontology contains the fact that *elem1* is present twice in the aggregation.

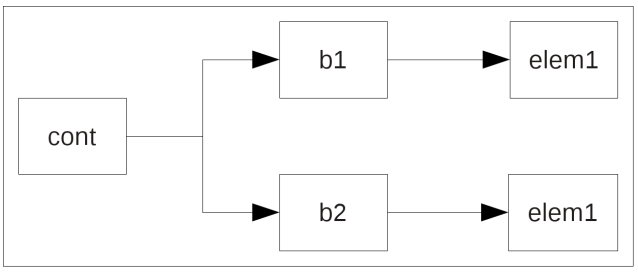

Figure 6.3: Representation of a Bag in OWL (ABox)

Select *SELECT* is a keyword used in EXPRESS to choose from a set of different types. A new OWL class is created to represent the *SELECT* construct. This class is defined as the union of all the types included in the *SELECT*.

Enumeration Enumerations are defined in EXPRESS as a finite set of values. In OWL, a new class representing the enumeration is created, and the values are represented by individuals. The OWL *OneOf* construct is used to give the individuals of the enumeration.

Abstract entity and class An entity in EXPRESS may be declared as abstract. The meaning is the same as in object-oriented programming: an abstract entity cannot be directly instantiated, but it may be subclassed. Consider, as an example, the following two entities extracted from AP203:

```
ENTITY document_reference
ABSTRACT SUPERTYPE;
assigned_document : document;
source            : label;
END_ENTITY; -- document_reference
ENTITY cc_design_specification_reference
SUBTYPE OF (document_reference);
items : SET [1:?] OF specified_item;
END_ENTITY; -- cc_design_specification_reference
```

The entity *document_reference* is defined as abstract, and the entity *cc_design_specification_reference* is defined as its subtype. This means that the entity *document_reference* cannot be directly instantiated, but *cc_design_specification_reference* can be instantiated.

OWL does not provide any feature to translate the *ABSTRACT* keyword, i.e. an OWL class cannot be declared as abstract. Using an OWL class to represent an abstract entity causes a problem: because of the Open World Assumption, we cannot assume that the OWL class will never be instantiated. To overcome this problem, we can declare the subtype classes as partitions of the supertype. A partition forces the instances of the supertype to belong to at least one subtype. This is achieved by declaring that the set of instances of the supertype is equivalent to the set of instances of all its subtypes. In that case, if an individual is declared as an instance of *document_reference* and not an instance of *cc_design_specification_reference*, the reasoner would detect an inconsistency. However, this solution works only when the supertype and all the subtypes are declared within the same schema. Because of these reasons, we choose to ignore the *ABSTRACT* keyword. Consequently instantiating an entity defined as abstract in the EXPRESS schema will not raise an error.

Inheritance EXPRESS provides three keywords to specify the allowed combination of subtypes for an entity: *ONEOF*, *ANDOR*, and *AND*. Along with the ABSTRACT keyword, they restrict the usage of the instantiation mechanism.

ONEOF: The *ONEOF* keyword takes a list of entities as its parameter, and it specifies that only one of these entities can be instantiated. An equivalent behavior in OWL is obtained by defining the subclasses as disjoint: an inconsistency is detected when an individual is an instance of two of these subclasses. We mark the set of classes contained in a

ONEOF list as disjoint. Another solution could be to use the logical definition of XOR. We could also use the OWL intersection, union and complement operations, to translate *AND*, *OR*, and *NOT*. However, this increases the complexity of the ontology, as the length of the formula increases dramatically with the number of elements involved. For this reason, we choose the first solution.

ANDOR: When no specific constraints are defined, the default keyword for the instantiation is *ANDOR*. This means that the instance can belong to more than one subclass. A set of entities joined by an *ANDOR* is translated to the union of the corresponding classes in OWL. We first represent the union of the subclasses by using the *ObjectUnionOf* construct and then declare this union to be equivalent to the parent class.

AND: The *AND* operator imposes that the object is an instance of all the subclasses. To represent this constraint in OWL, we use the *ObjectIntersectionOf* construct to link the subclasses.

More details regarding the translation of other EXPRESS concepts, such as *UNIQUE*, *LIST*, *ARRAY*, and *SET* can be found in [77].

6.3.2/ MAPPING THE PRODUCT MODELS

EXPRESS schema can be instantiated in two different ways. The first way is to create a file as defined in "Clear Text Encoding of the Exchange Structure -10303-21," or Part 21 [78]. The second way is to create an XML file as defined in Part 28[79]. Computer-Aided Design (CAD) software can export data in STEP format that complies with the AP203 schema and Part 28. This section addresses the translation of Part 21 files.

The translation to OWL is performed by performing a syntactic analysis (a process called parsing) to recognize all the instances declared within the Part 21 file, and then creating individuals and property assertions. In STEP, the schema and the instances are declared in different files: the related schema is specified in the Part 21 file in the *FILE_SCHEMA* section. In a similar fashion, we generate two different ontologies during the translation: a schema ontology for the EXPRESS schema and an instance ontology for the Part 21 file. OWL provides a mechanism to import statements declared in an external OWL ontology. We use this feature in the instance ontology to import the schema ontology, so that we keep the schema ontology separated from the instance ontology while accessing both simultaneously. By having the final ontology containing both the TBox and the ABox, we are able to check the consistency of the instances against the schema.

All the EXPRESS instances contained in a Part 21 file are distinct, which means that any two EXPRESS instances represent two different real world objects. Conversely, OWL individuals are not inherently distinct. Unless explicitly declared as distinct, two OWL individuals may represent the same real world object. To translate the EXPRESS instances correctly, it is then necessary to declare explicitly all the OWL individuals contained in the same file to be distinct using the *DifferentIndividuals* OWL construct.

The processing of an unknown fact is another major difference between EXPRESS and OWL. In EXPRESS, any unknown fact is supposed to be false. For example, let us consider two EXPRESS entities called *product* and *product_category*. If an instance of *product* is not declared as an instance of *product_category*, then the system assumes it is not. This behavior is called the Closed World Assumption (CWA), because it supposes that the world is limited to what is stated. OWL uses the Open World Assumption (OWA):

unless a reasoner proves a fact is false, that fact is unknown. Hence, the translation sometimes requires additional information to capture the semantics of EXPRESS in OWL. The difference between CWA and OWA causes a translation problem when an instance is constrained to have one attribute. The attribute ID of the entity product is not declared as optional, so it must be instantiated for all the instances of product. In EXPRESS, the lack of data will raise an error. In OWL, even if the id is not specified for an instance of product, the reasoner will not detect an inconsistency: the instance is still considered to have an unknown id. To allow the reasoner to detect an inconsistency if an id is missing, it would be required to declare explicitly that that instance of product has no id. However, research is being conducted to work with CWA in OWL [80]. In the context of optional attributes, the use of CWA in OWL would allow correct representations of the constraints defined in the EXPRESS schema. An error would occur if a mandatory attribute was missing.

The translation of some additional concepts, such as derived data types, is also required before completing the translations of STEP APs.

6.3.3/ BENEFITS OF THE NEW REPRESENTATION

OWL [81] is based on Description Logics (DL), a family of knowledge representation languages. These languages can be used to define domain concepts according to a predefined and well-understood formalism. Concepts are used to represent the domain of the objects, while roles are used to represent relationships between these concepts. The OWL representation provides benefits subject to the level of OWL expressivity. The expressivity of OWL is denoted using different characters, such as ($\mathcal{D}$). The explanation of this expressivity is given in [82]. In [76] the authors provide examples of DL semantic axioms in product modeling and highlight that DL semantics can be implemented in a reasoner engine to:

- Check the consistency of the ontology.
- Perform inference on the class hierarchy.
- Perform inference on the membership of the individuals to the classes.
- Query and search the model.

The performance of the reasoner engine depends on the expressivity of the ontological representation. In the next paragraphs, first, the tradeoffs between performance and expressivity for ontology reasoning are explained. Then, we introduce three benefits given by the reasoning: consistency checking, inference procedures, and queries.

Performance of the reasoning and expressivity of ontological representation The choice of OWL may lead to performance issues when dealing with large ontologies. Because the use of certain OWL constructs have severe repercussions on the computational time and memory space, it is possible to improve the performance by altering parts of the mapping. The second version of OWL, OWL 2, defines sublanguages that trade some expressive power for the efficiency of reasoning. An option for improving the performance is to use only the constructs defined in the sublanguage called OWL 2 Rule Language (RL). The use of OWL 2 RL guarantees in the worst case a polynomial complexity instead of an exponential complexity for OWL 2. Some restrictions defined in the EXPRESS schemas will not be correctly expressed into OWL 2 RL, but the performance will be drastically bet-

ter than using OWL. The consequence of limiting the expressivity to OWL RL is that the following restrictions defined in EXPRESS schemas will not be expressed: the minimum cardinalities on aggregations do not appear and the ENUMERATION type and SELECT type are considered extensible. The benefit of this approach is that the reasoning time is improved, especially for large ontologies. It is important to note that the choice of using OWL 2 RL to represent the ontology does not affect the translation of the data. Only some restrictions declared in the schema are not translated, which means only the validation of the data is affected.

Consistency checking of the ontology The consistency checking procedure can be applied at the schema level and the instance level. At the schema level, it determines whether an instantiation of a concept should create an inconsistency in the model. At the instance level, it checks whether an instance of a class satisfies the definition of this class.

Currently, software libraries are available to check the consistency of EXPRESS schemas and Part 21 files. With OntoSTEP, a DL reasoner performs consistency checking at the schema as well as at the instance levels. Checking the logical consistency of the OWL classes and individuals resulting from a translation is a necessary condition to use an inference procedure.

Knowledge inference from the ontology An inference procedure uses the data evidences in a context and draws conclusions using certain problem solving strategies [83]. To perform inference procedures, reasoners use a knowledge base as a source of data, such as concepts, roles, and axioms, to reach a conclusion. The expressivity of the axioms and concept definitions is dependent on the used logical language.

Once the reasoner has applied all the inference procedures on our ontology, new knowledge and data are made available both at the schema and at the instance level. These dynamic modifications cannot be done in EXPRESS. One can then use a querying mechanism to query the new data, which represents an enriched version of the original ontology.

Querying the ontology Queries are performed to retrieve specific data from a large amount of information. This mechanism does not readily exist as part of STEP, although some mechanisms have been developed [84]. In our case, we perform queries to retrieve some specific product information from an OntoSTEP file. The information contained in a CAD file is first translated into OWL representation, then checked for consistency, inferred upon, and finally queried.

Two approaches are mainly used to perform queries on OWL ontologies. The first approach uses a language called SPARQL Protocol and RDF Query Language (SPARQL) [85], and the second approach uses the Semantic Query-Enhanced Web Rule Language (SQWRL) [86]. OntoSTEP gives users the freedom to choose any query language.

SPARQL was specifically developed for RDF models, so we would need to translate our OWL ontology to RDF before performing SPARQL queries. SPARQL has two major drawbacks. First, the translation from OWL to RDF increases the computational time. Second, the OWL 2 reasoner we used, Pellet [87], does not support several SPARQL built-in func-

tions, such as DESCRIBE, OPTIONAL, or FILTER. While SPARQL was developed for RDF, SQWRL was specifically developed for OWL. It is based on Semantic Web Rule Language (SWRL) [88] and does not need any bridge[24]. In addition to the large number of its built-in functions, SQWRL also provides some classical aggregation functions like maximum, minimum, sum, or average, which are missing in SPARQL. While being more appropriate to OWL, we found two issues with SQWRL. First, it is based on the proprietary engine Jess (a rule engine and scripting environment) [89], which is the only option currently available to process SQWRL queries. Second, because it is based on [88], [86] does not allow combining functions together, e.g., it is not possible to query the maximum of averages.

6.3.4/ Implementation of the translation

In this section we present and discuss aspects related to the implementation of these rules and mappings. The main goal here is to create tools to generate ontologies from STEP data. These tools translate both schema files and instances files (Part 21).

We use the Protégé [90] editor to implement OntoSTEP as a plug-in to Protégé. Protégé is a free, open source ontology editor and knowledge base framework. It is one of the most widely used tools to edit and manage knowledge bases. The Protégé architecture allows third party developers to write their own extensions in Java.

The implementation involves the following steps: generation of the OWL Schema from the EXPRESS Schema (schema translation for creating TBox), generation of the OWL individuals from the Part 21 file (schema instantiation for creating ABox), and development of a plug-in to integrate the TBox and ABox within the Protégé environment.

Translating EXPRESS schemas The translation of an EXPRESS schema is carried out in two stages. First, we retrieve the syntax tree, i.e. structured representation, of the schema. This is done by using a tool to generate a parser from the EXPRESS grammar. This parser is then able to create a syntax tree from a file that conforms to the grammar. Second, we scan the syntax tree and apply the rules described in Section 6.3. During this scan, specific actions are taken depending on the encountered element. For instance, in our implementation, the detection of the keyword ENTITY leads to the creation of a class in [81].

To obtain the syntax tree of the EXPRESS schema, we use a modified version of the open source EXPRESS Parser [91], which contains the EXPRESS grammar. This parser is implemented using ANTLR (ANother Tool for Language Recognition) [92] to generate the EXPRESS parser from the EXPRESS grammar. We used the open source OWL Application Programming Interface (API) [93] for OWL 2 to create the ontology. This API is used by Protégé 4 [90].

Another approach could have been to use a metamodeling approach. As a Meta-Object Facilities (MOF)[94] representation has been developed for both OWL[95] and EXPRESS[96], it could be possible to specify the mapping in a formal way using a language such as Query/View/Transformation (QVT)[97].

Translating instances The translation process for Part 21 files is similar to the translation process for the EXPRESS schema. First, we retrieve the syntax tree of a Part 21 file using the Part 21 grammar and a parser generator. Second, we scan the syntax tree and create the OWL constructs using an API. For each instance encountered in the Part 21 file, an individual is created, and its attributes are obtained and translated.

The result is a file containing all the individuals and the assertions on these individuals (ABox). The TBox translation of the EXPRESS schema is also imported as it contains the definition of the OWL classes. The same technologies as for the TBox are used: ANTLR is used to generate a Part 21 parser, and OWL API is used to generate the OWL constructs.

6.4/ PRODUCT MODELS INTEGRATION

This section provides an example of data integration by the combination of product models defined in OWL. Whereas STEP formats do not support integrating different STEP product models, OWL supports the integration of different ontologies. This feature makes it possible to semantically relate object located in different files.

Motivation for integrating product models The relationships among product models need to be captured to organize the product models. This organization has two major benefits: it allows the search of a particular piece of information, and it facilitates the understanding of the product model by giving a context.

In STEP, the relationships among product models are not defined at a low level of granularity. Considering that a product model is composed of objects, within a product model, an object can refer to another product model, but not to a contained object. On the other side, OWL makes it possible to establish semantic relationships between objects present in different files. The integration mechanism is part of the OWL language, so no other integration mechanism is required.

Example of integration: AP214 and AP239 PDM is a domain that can benefit from product model integration. The idea behind PDM is to define a core product model on which any product data can be attached. PDM defines bill of materials, system breakdowns, or activity definitions that help organizing product data. However, due to the limitation of STEP, the externally-defined product data can only be files, and not objects defined within these files. Figure 6.4 shows different levels of granularity in the reference to external knowledge. On the left side, references target files, while on the right side, references target specific objects.

Connecting to external objects may be achieved using additional mechanisms, which are not part of STEP. With OWL, these mechanisms are directly available through the language used to express the information. This is made possible because the same language is used to express information, and because this language supports references to any objects, using the system of URI.

The following example shows how to connect a product defined in an Application Protocol 214 – Core data for automotive mechanical design processes (AP214) CAD file, and a

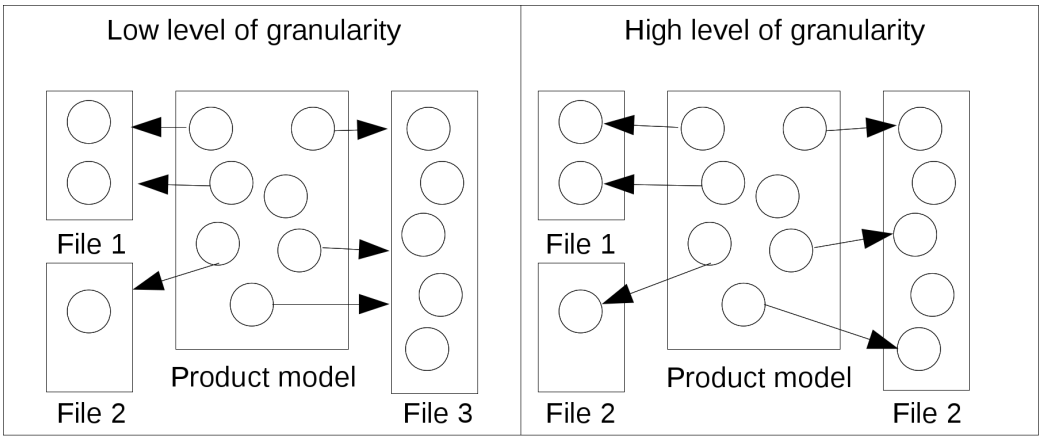

Figure 6.4: Granularity of the connections

product defined in an Application Protocol 239 – Product life cycle support (AP239) PDM file. Both files are translated into OWL using OntoSTEP, so that it is possible to establish semantic relationship between parts of these two product models.

Figure 6.5 shows in Unified Modeling Language (UML) the EXPRESS entities to define parts. The three main entities are *product*, *product_version*, and *product_definition*. *Product* represents a particular product. It has an identifier, a name, and a description. *Product_version* represent a particular version of a product. It has an identifier, a description, and the related product. *Product_view_definition* represents a particular view of a product version. It has an identifier, a name, a characterization, one of several contexts, and the related product version. Each of these entities specializes respectively in *Part*, *Part_version*, and *Part_view_definition*. A *Product_category*, with an identifier, a name, and a description, can be defined and assigned to a *product* using the *Product_category_assignment* entity. Finally, a *Identification_assignment* can be assigned to a *Product* or a *Product_version*. It has a identifier, a role, a description, and the related identified items. The other entities introduced in this example are not essential to understand this example.

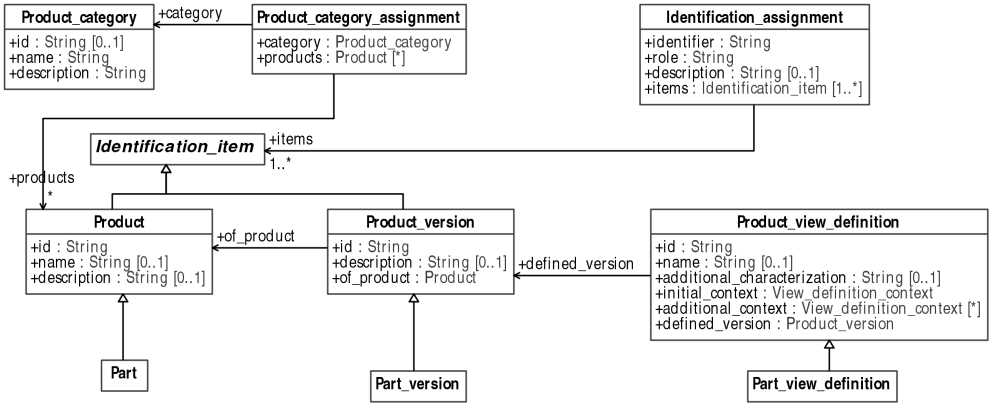

Figure 6.5: Samples from the AP239 (in UML)

The following sample is extracted from the AP239 file introduced in the previous chapter. The *Part* declared is the gate valve presented in Section A.5. The identifiers of the *Part* and of the *Part_version* are not defined as attributes, but using the *Identification_-*

assignment entity. A *Product_category* “part” is assigned to the gate valve.

```
...
#14=PRODUCT_CATEGORY('/IGNORE','part','/IGNORE');
...
#55589=PART('/IGNORE','/IGNORE','/IGNORE');
#55590=IDENTIFICATION_ASSIGNMENT('2110-FUELOIL-V816',
   '/IGNORE','/IGNORE',(#55589));
#55594=PRODUCT_CATEGORY_ASSIGNMENT(#14,(#55589));
#55595=PART_VERSION('/IGNORE','/IGNORE',#55589);
#55596=IDENTIFICATION_ASSIGNMENT('None','/IGNORE',
   '/IGNORE',(#55595));
#55600=PART_VIEW_DEFINITION('/IGNORE','/IGNORE',
   '/IGNORE',#22,$,#55595);
...
```

As for the CAD file, Figure 6.6 shows the EXPRESS entities to define parts in AP214. A *product* has an identifier, a name, a description, and a frame of reference. A *product_definition_formation* represents a product version. It has an identifier, a description, and the product it relates to. A specialization of this entity is *product_definition_formation_with_specified_source*, and can indicate whether the version is bought or manufactured. A *product_definition* represents a particular view of the product. It has an identifier, a description, the related product version, and the frame of reference of the view. A category can be represented using *product_category*, which has a name and a label. A specialization called *product_related_product_category* can refer to the related product.

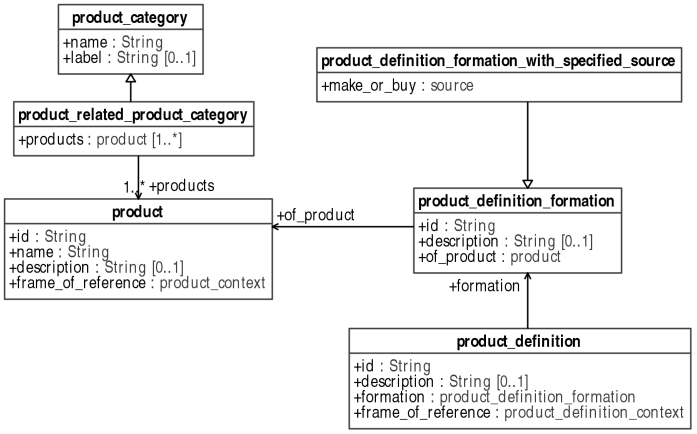

Figure 6.6: Samples from the AP214 (in UML)

The following sample is extracted from the AP214 CAD file named 2110-FUELOIL-V816, and described in Section A.5. The product is identified and named simply as “Product”, and it has a “part” category.

```
...
#32=PRODUCT('Product','Product','',(#31));
#33=PRODUCT_RELATED_PRODUCT_CATEGORY('part',$,(#32));
...
#40=PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE(
'None','',#32,.BOUGHT.);
```

```
...
#52=PRODUCT_DEFINITION('None','',#40,#51);
...
```

Note that the gate valve has a different name, which makes it hard to develop tools to automatically establish the relationships.

Benefits By translating these two STEP files into OWL, it is possible to directly establish the relationship between the part in the CAD file and the part in the PDM file. It is actually possible to navigate through the PDM model, and see all the occurrences of a part, using simply the mechanisms offered by the OWL language. And this concept can be extended to any object defined in STEP. Figure 6.7 shows the resulting ontology after reasoning. The two parts previously introduced share the same properties.

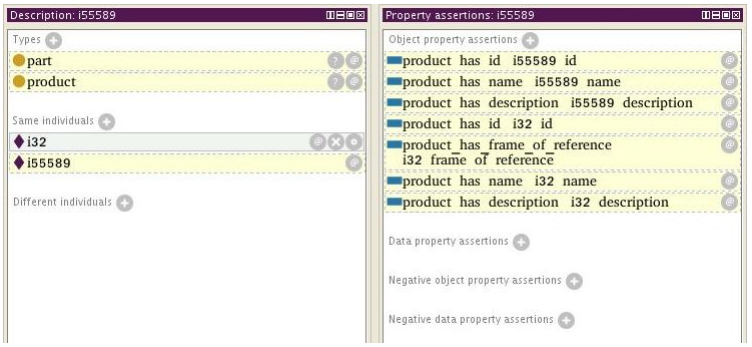

Figure 6.7: Integrated parts in Protégé

6.5/ USE OF EXTERNAL CONTROLLED-VOCABULARIES

Product model schemas are built for generic purposes, so they do not always capture business specific concepts. However product modelers sometimes need to refer to more specific concepts to fully capture fine details. Shared ontologies offer a controlled vocabulary and can be easily used by various product modelers to precisely define specific concepts. People referring to the same concept use the same definition of this concept. Shared ontologies do not always provide sufficient specialization to precisely express the designer's idea, so these ontologies can often be extended to capture more specific concepts. An example of shared ontology in the product modeling domain is ISO 13584: Part Library (PLib) [40]. PLib aims at offering a model and exchange format for digital libraries of technical components.

To use a shared ontology, it is necessary to establish the connection between a product model and the external concepts it refers to. This connection will likely be explicitly stated within the product model, or defined elsewhere if the product model schema does not allow such extension.

As an example of shared ontology usage, if a modeler creates a product model that represents a piping system, all the parts will be indicated as "part". The modeler may want to assert these parts are actually pipes, elbows or valves, but the original schema

does not contain these concepts. The modeler can then connect the part to a concept that is defined in PLib.

The use of ontologies enables semantic searches to be performed instead of text searches. For instance, let's suppose a part is declared as a "gate valve". If someone is looking for a "valve", a simple text search will be able to retrieve this part. However, if someone is looking for "piping equipment", a text search will fail. In a semantic context, "gate_valve" is not just a text, but a concept that is connected to other concepts. So assuming the "gate valve" is well defined as piping equipment, the semantic query will succeed.

Presentation of the STEP approach to refer to external concepts STEP provides a mechanism that enables customization of product models with domain-specific concepts. Modelers can classify STEP instances with concepts defined in an external controlled vocabulary. This is called external classification, as this controlled vocabularies formally define the semantic of domain-specific concepts that specialize generic terms from the EXPRESS schemas.

In the following example (Figure 6.8), we show an OWL ontology that defines a product taxonomy. This taxonomy can be used to specialize the generic concept of product defined in APs such as AP203, AP214 or AP239. The use of such taxonomy helps capturing more accurately the intent of the product modeler.

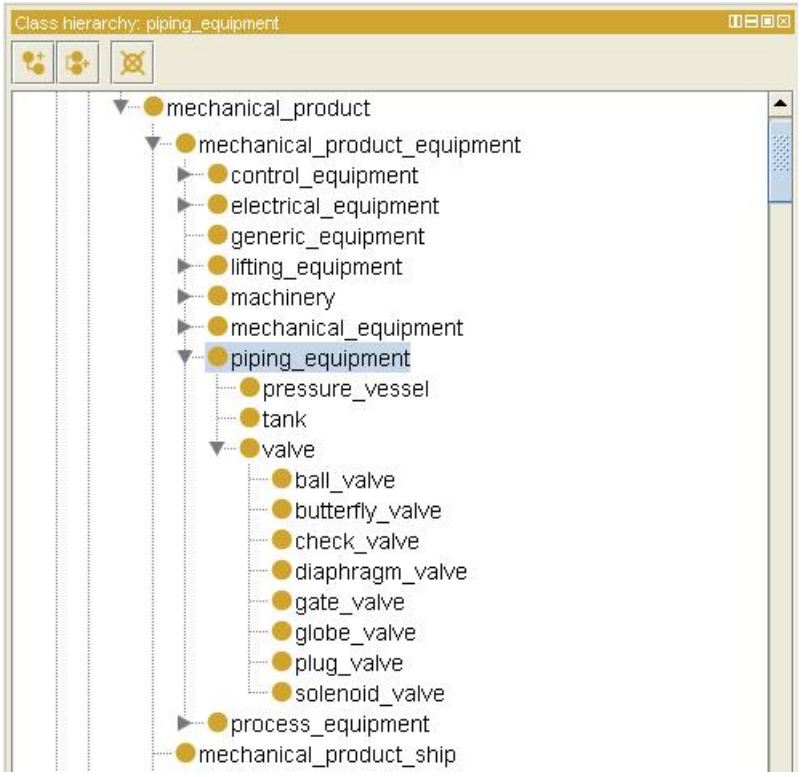

Figure 6.8: OWL ontology used for classification

The most recent STEP APs, such as AP239 or AP203 second edition, support the external classification. Instances are classified by providing an external library identifier and a class name. Unfortunately no specific guidance or limitations are given in the standard

regarding how the classification mechanism should be implemented. Additional and ad hoc tools are needed to check the correctness and to merge the external knowledge with the product model. Implementation guidance can be defined as part of an agreement. For example, for the AP239, the Organization for the Advancement of Structured Information Standards (OASIS) recommends that OWL be used for representing controlled vocabularies that specialize the AP239. In a particular data exchange agreement using Product Lifecycle Support (PLCS), an identifier is given to the external library. This identifier is then used in the STEP instance file. As a result, knowledge of the agreement is necessary to implement the classification mechanism.

Presentation and benefits of the OntoSTEP approach to refer to external concepts
OntoSTEP provides a mechanism for translating product models from STEP to OWL. One of the goal of OWL is to provide a way for different data source to commit to the same ontology for shared meaning [98]. As part of the Semantic Web, OWL ontologies can be easily published on the Internet. One of the benefits of having a representation of STEP product models into OWL is the possibility to directly classify individuals defined in the product model as concepts defined in external ontologies. The knowledge associated with these concepts is automatically imported and merged with the product model knowledge. Because this mechanism is part of the OWL language, no additional tool is necessary to achieve the goal of classifying STEP instances.

Figure 6.9 shows a query over a product model translated into OWL in Protégé. The individuals #55403, #55456 and #55483 were respectively classified as gate valve, check valve and globe valve according to the taxonomy shown in Figure 6.8. This query asks for the piping equipment in the product model, and thanks to the knowledge imported from the taxonomy, all three products were identified as piping equipment.

6.6/ CONCLUSIONS

This chapter presented the benefits of a semantic representation for product models with regards to preservation. The OWL representation provides a better expression of the information by allowing knowledge interconnection and reasoning. This makes the information more understandable and accessible than if STEP was used. The OWL representation should not be seen as a replacement for STEP, but rather as a complementary solution.

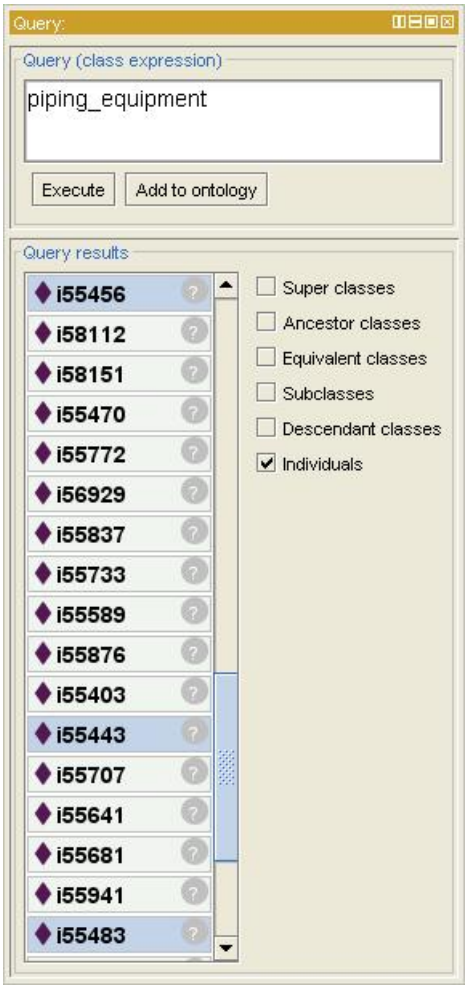

Figure 6.9: Individuals classified as piping_equipment

7

CONCLUSION AND FUTURE WORKS

This thesis presented issues related to digital preservation. The thesis looked at the problems from a computer science point of view, with a particular focus on product models. This thesis proposed an approach for the formal description of archival systems and for a semantic representation of product models.

7.1/ SUMMARY

Information preservation is a complex field: many issues need to be addressed to enable long-term preservation and accessibility. The emergence of digital representations as a main way to exchange information created new problems regarding preservation. A first problems concerns the necessity to have information systems that store and manage digital content. Such systems need to be designed in a way that demonstrates their ability to support preservation and accessibility of the content. The second problem concerns the digital contents themselves, and the characteristics of the formats. These formats should, among other things, allow a precise definition of the information, and provide evidence that the data can be discovered and understood by both computers and humans. This can be achieved by choosing a format that enables semantic relationships.

These generic challenges are addressed in this thesis with a particular type of information in mind: product models. Product models are formal representation of products. The preservation of product models is complex for various reasons. A first reason is that product model preservation is usually one among many business functions. As such, the archival systems may interact with many different actors that either submit information to the archive for preservation, or access preserved information to support business activities. The design of the archival system has to represent this complex environment to demonstrate the ability to preserve information. A second reason is that the product models representation relies on complex information models and different formats that have different characteristics. These characteristics are not necessarily beneficial from a long-term preservation perspective.

Because digital preservation, especially product model preservation, is not trivial, various efforts have attempted to address some of the main problems. Regarding the description of archival systems, one of the main efforts has been to clearly define a terminology for preservation, so that it is possible to describe and compare preservation solutions using a common vocabulary. This effort is the Reference Model for an Open Archival Information System (OAIS RM). Another effort has defined a set of criteria that should be enough to demonstrate the trustworthiness of the archive: the Audit and Certification of

Trustworthy Digital Repositories (ACTDR). Because these efforts were not proposing or recommending a particular way to describe actual archives, it was necessary to find a way to describe not only the archival system, but also the potentially complex environment of product models preservation. The description of systems within an enterprise, called enterprise architecture, was selected for this task. Regarding the representation of product models, the most commonly agreed standard for product model preservation, Standard for the Exchange of Product model data (STEP), did not facilitate a precise representation of the information, especially to enrich product models with domain-specific knowledge, and establish semantic relationships among product models. As a result, the translation of STEP information into Web Ontology Language (OWL) was proposed. This translation provides the benefits previously stated, and hence gives more guarantees that product models can be understood over time by both human and computers.

The description of archival systems was addressed using enterprise architecture. The proposed approach was to see how a generic enterprise architecture framework could be used for archival systems in particular. This approach, called Reference Architecture for Archival Systems (RAAS), provides an archival terminology, partly based on the OAIS RM, that can be used along with enterprise architecture terminology to describe archival systems. RAAS uses Department of Defense Architecture Framework (DoDAF) to provide the enterprise architecture terminology, such as systems, activities, information, or services. As a result, the approach makes it possible to easily describe and compare archival system description, in the continuation of the goals of the OAIS RM. In addition to defining the elements of an architecture, RAAS includes a selection of views, based on some of the ACTDR criteria, to demonstrate that the archival system can be trusted. RAAS provides a clear and formal description of accepted contents, what additional information is needed for the preservation and accessibility, what are the activities involving producers, consumers, and managers, and describes how the archival system fulfills its preservation objectives. RAAS makes it possible to describe and compare archival systems using a formal terminology, based on enterprise architecture terminology. This approach also facilitates the integration of the archival system within an enterprise by showing how to interact with the archival system. Another benefits of relying on Unified Modeling Language (UML) is the possibility to use code generation in a context of Model-Driven Architecture (MDA)[99].

To keep the scope manageable and coherent with the computer science focus of this thesis, the implementation of certain preservation activities was not covered. Activities that does not involve interactions with the archival system are not considered (e.g. monitoring activities).

A high level description of a product model archival system, developed in accordance with RAAS, was then presented. The objective was to demonstrate the ability of RAAS to describe the main aspects of a product model archival system, so that the system can successfully support the preservation of these product models. This description showed the ingest of Product Data Management (PDM) information and managed product data. A maintenance use case was introduced to show how the preserved data can be accessed. The overall description showed different aspects of the preservation: what information was preserved, what additional information was needed to ensure long-term preservation and access of product models, what activities were supported by the archival system, and how the users interact with the archival system. The description put the archival system within the context of the enterprise, by describing how the information is transferred from its original source to the archive, and then how this information is accessed by other

business functions. This description showed that RAAS could be used to describe an archival system in accordance with the OAIS RM, to provide views that demonstrate how the archival system supports the preservation activities, and to show how the archival system integrates within an enterprise.

The representation of product model was also addressed in this thesis. The representation affects the ability to understand and access the information over time. It also affects the technology that will be used to implement the archival system. A translation of STEP data into OWL was proposed to improve this representation with regards to long-term preservation. STEP is a standard that proposes information models and file formats for the representation of product models. However, even if STEP is widely accepted as the format for product model preservation, it offers a limited description of the information. Product data cannot be directly related to knowledge outside of the product model. Such limitation makes more complex the ability to understand and access this product data over time. By translating STEP product models into OWL, it becomes possible to natively – without using additional tools – connect product data to externally-defined product data, or to use controlled vocabulary to further specialize product data.

7.2/ USAGE SCENARIO FOR THE PROPOSED ARCHIVAL APPROACH

The reference architecture and the semantic representation of product models can be put together into a global preservation scenario.

1. a designer creates various product models for a product
   (a) he creates a CPM model to conceptually represent the form, function, and behavior of the product
   (b) he creates CAD files for the assembly and the parts
   (c) he manages the different models using a PDM system
2. the designer wants to make sure the product models he created are preserved over time
   (a) he looks at the product model archival system description to know which formats are accepted
   (b) he translates the CAD files into STEP AP203 or AP214
   (c) he translates the PDM information into STEP, according to the Data EXchange Specification (DEX) defined in the archival system description.
   (d) he then translate these CAD files and PDM information into OWL
   (e) the CPM model is already in OWL
   (f) he establishes the semantic relationships among the product models, by marking as equivalent the different representation of the same product in different product models.
   (g) he enriches the product models by connecting some entities to more specialized concepts. For example, all the parts and assembly he designed are classified according to a part library
   (h) he sends the files to the archive, according to the protocol defined in the architecture (Submission Information Package, activities, services)
3. later on, someone else wants to look for product models for a specific type of part
   (a) he can access the repository according to the protocol defined in the architecture
   (b) he searches for the specific type of part in the archival system. The archival

system description tells him how to access the repository, and what information models are used. Thanks to the reasoning feature of OWL, he is able to locate the parts he is interested in, and the product models that contain them.
   (c) once the product models are identified, he is able to retrieve them thanks to the archival system description
4. in case of obsolescence of the formats or of the archival system
   (a) the content is converted to new formats. The public availability of STEP information models and formats and of OWL makes it possible to create specific translators if none are available
   (b) the availability of the archival system architecture facilitates the transition to the new information models and formats
   (c) the availability of the archival system architecture facilitates the transition to the new systems

7.3/ CONCLUSIONS AND FUTURE WORKS

The development of RAAS is the result of a conciliation between the OAIS RM and an Enterprise Architecture Framework (EAF). This work is beneficial to both sides: it shows how parts of the OAIS RM can be concretely represented, and it provides a standard terminology to describe archival systems using enterprise architecture.

RAAS provides a strong base for a more comprehensive description of archival systems. It can be used to see how the archival system is integrated within an organization, and how it communicates with the other systems or people. Future works regarding RAAS could include incorporating ideas from the domains that were not addressed in this thesis. Actually implementing an archival system would indeed require considering aspects, such as security, risk management, data quality, as identified in [60]. As this thesis focused on the interactions and functions of the archival system, one direction is to enrich RAAS with concepts from these domains. Another direction is to consider the organizational unit that is in charge of the preservation, and describe the preservation activities that drive the interactions with the archive. RAAS could also be further specialized to address more specific contexts, such as a specific organization, or a specific target of preservation. Finally, another complementary work would be to combine already existing archive development method with RAAS to provide a comprehensive development method for the archival system.

To follow-up on the use case, a next step would be to go further in the application, and implement the entire archival system from the architectural description, in a production environment. Transitioning from enterprise architecture to software architecture may include different types of programs.

First, the archival system itself can be detailed to an implementation level. RAAS already provides a high level description of the archival system, including the services it provides, the functions it performs, or the information it manages. In a practical case, the entire archival system is likely to be composed of several components working together and having different roles: the programs that provide the archival services, the programs in charge of storing the data, and the programs in charge of managing this data may not be part of the same solution.

For example, Fedora [67] is a digital repository that proposes services. Fedora has a

modular architecture. Fedora can delegate the physical file storage to specialized software such as iRODS[100] or SRB[101]. The data management can be performed by RDBMS such as MySQL[102] or PostGreSQL[103]. Metadata management can be performed by Resource Description Framework (RDF) datastore[104]. But Fedora will have to be tailored to manage specific types of content, so specific models may have to be developed. The architectural description can describe how the module is integrated within Fedora.

Another direction from the use case could be to address other types of product models, with data coming from multiple lifecycle stages, and address more ingest and access activities.

The semantic representation of product models enhances the way information is represented in digital form, by making it possible to establish semantic relationships among various sources of knowledge. Future works for the semantic representation of product models could include leveraging the reasoning capability of OWL to infer more knowledge from the product model ontology (e.g. assembly feature recognition). It could also be possible to study how to extract information from various product data (text document, drawings, etc.) to automatically or semi-automatically extract product information in the form of models. Finally, the idea of using semantic representation for complex information can be extended to other types of information. For example, the principles could be applied to health-care, for the preservation of medical records.

A

COMPLEMENTARY INFORMATION

A.1/ OAIS FUNCTIONS

This section provides details on the functions defined in the Reference Model for an Open Archival Information System (OAIS RM)[1]. Figure A.1 provides an over view of these functions. Six main functions are defined, each of them being decomposed into subfunctions: **Ingest**, **Archival Storage**, **Data Management**, **Access**, **Preservation Planning**, and **Administration**.

A.1.1/ INGEST

This goal of the **Ingest** function is to accept Submission Information Packages (SIPs) from producers, and prepare the content for preservation. The **Ingest** function includes receiving SIPs, performing quality assurance on SIPs, generating an Archival Information Package (AIP) which complies with the archive's standards, extracting descriptive information from the AIPs, and coordinating updates to **Archival Storage** and **Data Management**.

The **Receive Submission** is in charge of receiving the SIPs from producers. This function also returns confirmation of receipt of a SIP to the producers, so that they can resubmit a SIP is an error occurred during the ingest.

The **Quality Assurance** function validates the SIP. Basic validations may include Cyclic Redundancy Checks (CRCs) or checksums associated with each file. Other types of validation can include whether the file follows the format (e.g. well-formed XML file), or the presence of mandatory data or metadata within the file or the SIP.

The **Generate AIP** function transforms one or more SIPs into one or more AIPs. This may involve file format conversions, data representation conversions or reorganization of the content information in the SIPs. This function may send SIPs or AIPs for audit to the **Audit Submission** function.

The **Generate Descriptive Information** function generate the Descriptive Information for an AIP, for example by extracting content from the AIP. Descriptive information consists of metadata used to search and retrieve AIPs.

The **Coordinate Updates** function sends the AIPs to **Archival Storage** and the descriptive information to **Data Management**. In general, the AIP is stored, and then he location of this AIP is added to the descriptive information before it is sent to **Data Management**.

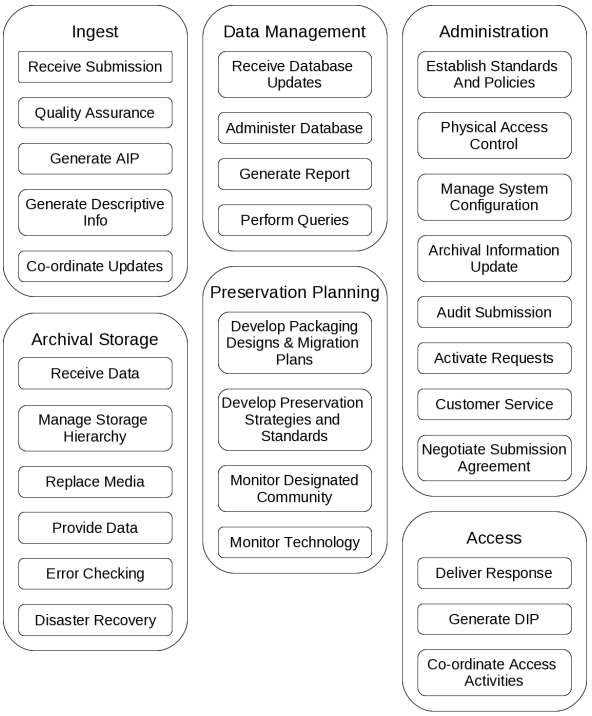

Figure A.1: Functions defined in the OAIS Reference Model

A.1.2/ ARCHIVAL STORAGE

The **Archival Storage** function is in charge of the storage, maintenance and retrieval of AIPs. The **Archival Storage** functions include receiving AIPs from **Ingest**, and adding them to permanent storage, managing the storage hierarchy, refreshing the media on which archive holdings are stored, performing routine and special error checking, providing disaster recovery capabilities, and providing AIPs to **Access** to fulfill orders.

The **Receive Data** function receives an AIP from **Ingest** function, and physically stores this AIP. This function also provides the location of the stored AIP.

The **Manage Storage Hierarchy** function is in charge of moving the stored AIP, and provide various statistics regarding the storage.

The **Replace Media** function can reproduce the AIPs over time. This function can modify the packaging of the AIP, but it cannot change the preserved data.

The **Error Checking** function aims to make sure the AIP is not corrupted. Fixity information, such as CRCs, can be used to check the data.

The **Disaster Recovery** function aims to prevent any potential problem by duplicating the preserved content and transferring it to a different physical location.

The **Provide Data** function aims to retrieve AIPs whenever they need to be accessed.

A.1.3/ DATA MANAGEMENT

The **Data Management** function is in charge of managing the descriptive information. The **Data Management** functions include administering the archive database, performing

database updates, performing queries on the data management data to generate result sets, and producing reports from these result sets.

The **Administer Database** function is in charge of maintaining the integrity of the Data Management database. The **Administer Database** function is responsible for creating the database, making the descriptive information available, and performing validation of the preserved content.

The **Perform Queries** function receives a query from the **Access** function, executes it, and returns the result set to the requester.

The **Generate Report** function is in charge of generating reports needed by the **Ingest**, **Access**, or **Administration** functions. These reports may include summaries of the preserved content or usage statistics. It may also return the descriptive information associated to a specific AIP.

The **Receive Database Updates** function adds, modifies or deletes information stored in the database. Such updates may be provided by the **Ingest** or the **Administration** function.

A.1.4/ ADMINISTRATION

The **Administration** function is in charge of determining how the archive should operate. The **Administration** functions include negotiating submission agreements, auditing submissions to ensure that they meet the agreed standards, and maintaining configuration management of system hardware and software. It is also in charge of monitoring and improving the archive operations, and to migrate the preserved content.

The **Negotiate Submission Agreement** function solicits desirable archival information and negotiates submission agreements with the producers. This function can also negotiate a schedule the the data submissions. The submission include what formats and what procedure are used.

The **Manage System Configuration** function is in charge of monitoring the archival system, by verifying the integrity and auditing the system operations, performances and usage.

The **Archival Information Update** function is in charge of updating the preserved content. To do so, this function requests a Dissemination Information Packages (DIPs), update the content, and resubmits it as SIPs.

The **Physical Access Control** function provides mechanisms to restrict or allow physical access to the archival system.

The **Establish Standards and Policies** function is in charge of establishing and maintaining the overall standards and policies for the archival system . It deals with budget information and different policies provided by the management.

The **Audit Submission** function is in charge of checking that the SIP and AIP conform to the submission agreements. In particular, it verifies that there is enough information for the consumers to understand the AIP.

The **Activate Requests** function is in charge of doing periodic requests to the archival system.

The **Customer Service** is in charge of managing the consumer accounts.

A.1.5/ PRESERVATION PLANNING

The **Preservation Planning** function is in charge of monitoring the environment of the archival system, and providing recommendations to ensure that the preserved information remains accessible over time. The **Preservation Planning** functions include evaluating the contents of the archive, recommending archival information updates, developing recommendations for archive standards and policies, and monitoring changes in the technology environment and in the designated community.

The **Monitor Designated Community** function is in charge of tracking the requirements of the producers and consumers, as well as the technologies associated with the content. Such technology include data formats, media, software packages, new computing platforms, and communication with the archival systems.

The **Monitor Technology** function is in charge of monitoring the technologies related to the archival system, such as hardware and software.

The **Develop Preservation Strategies and Standards** function is in charge of developing and recommending future strategies and standards for the archival system.

The **Develop Packaging Designs and Migration Plans** function develops new AIP or SIP designs, as well as migration plans and prototypes.

A.1.6/ ACCESS

The **Access** function is in charge of locating and disseminating a desired content to the consumers. The **Access** functions include receiving requests from the consumers, limiting the access to protected content, executing consumer requests, generating responses, and delivering these responses to Consumers.

The **Coordinate Access Activities** function is in charge of receiving all the requests from the consumers: queries, specific orders, or reports.

The **Generate DIP** function is in charge of retrieving the AIP from the **Archival Storage** function, and generating a DIP.

The **Deliver Response** function is in charge of returning what the consumer requested: the result os a query, a DIP, or a report.

A.2/ ACTDR CRITERIA

This section presents the Audit and Certification of Trustworthy Digital Repositories (ACTDR) criteria[31] addressed in this thesis. The criteria are described exactly as they appear in the ACTDR specification. Each criteria will be described with its ”supporting text“, and with ”Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement“, ac defined in ACTDR. In addition to this information, the Department of Defense Architecture Framework (DoDAF) view(s) used to fulfill this criteria are given.

3.1.1 The repository shall have a mission statement that reflects a commitment to the preservation of, long-term retention of, management of, and access to digital information.

Supporting Text This is necessary in order to ensure commitment to preservation, retention, management and access at the repository's highest administrative level.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Mission statement or charter of the repository or its parent organization that specifically addresses or implicitly calls for the preservation of information and/or other resources under its purview; a legal, statutory, or government regulatory mandate applicable to the repository that specifically addresses or implicitly requires the preservation, retention, management and access to information and/or other resources under its purview.

DoDAF views Overview and Summary Information (AV-1), High-Level Operational Concept Graphic (OV-1), Vision (CV-1)

3.1.2 The repository shall have a Preservation Strategic Plan that defines the approach the repository will take in the long-term support of its mission.

Supporting Text This is necessary in order to help the repository make administrative decisions, shape policies, and allocate resources in order to successfully preserve its holdings.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Preservation Strategic Plan; meeting minutes; documentation of administrative decisions which have been made.

DoDAF views Overview and Summary Information (AV-1), High-Level Operational Concept Graphic (OV-1), Vision (CV-1)

3.1.3 The repository shall have a Collection Policy or other document that specifies the type of information it will preserve, retain, manage, and provide access to.

Supporting Text This is necessary in order that the repository has guidance on acquisition of digital content it will preserve, retain, manage and provide access to.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Collection policy and supporting documents; Preservation Policy, mission, goals and vision of the repository.

DoDAF views Conceptual Data Model (DIV-1)

3.3.1 The repository shall have defined its Designated Community and associated knowledge base(s) and shall have these definitions appropriately accessible.

Supporting Text This is necessary in order that it is possible to test that the repository meets the needs of its Designated Community.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement A written definition of the Designated Community.

DoDAF views Organizational Relationships Chart (OV-4)

3.5.2 The repository shall track and manage intellectual property rights and restrictions on use of repository content as required by deposit agreement, contract, or license.

Supporting Text This is necessary in order to allow the repository to track, act on, and verify rights and restrictions related to the use of the digital objects within the repository.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement A Preservation Policy statement that defines and specifies the repository's requirements and process for managing intellectual property rights; depositor agreements; samples of agreements and other documents that specify and address intellectual property rights; documentation of monitoring by repository over time of changes in status and ownership of intellectual property in digital content held by the repository; results from monitoring, metadata that captures rights information.

DoDAF views Operational Rules Model (OV-6a)

4.1.1 The repository shall identify the Content Information and the Information Properties that the repository will preserve.

Supporting Text This is necessary in order to make it clear to funders, depositors, and users what responsibilities the repository is taking on and what aspects are excluded. It is also a necessary step in defining the information which is needed from the information producers or depositors.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Mission statement; submission agreements/deposit agreements/deeds of gift; workflow and Preservation Policy documents, including written definition of properties as agreed in the deposit agreement/deed of gift; written processing procedures; documentation of properties to be preserved.

DoDAF views Conceptual Data Model (DIV-1), Operational Activity Model (OV-5b)

4.1.2 The repository shall clearly specify the information that needs to be associated with specific Content Information at the time of its deposit.

Supporting Text This is necessary in order that there is a clear understanding of what needs to be acquired from the Producer.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Transfer requirements; producer-archive agreements; workflow plans to produce the AIP.

DoDAF views Conceptual Data Model (DIV-1), Logical Data Model (DIV-2), Physical Data Model (DIV-3)

4.1.3 The repository shall have adequate specifications enabling recognition and parsing of the SIPs.

Supporting Text This is necessary in order to be sure that the repository is able to extract information from the SIPs.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Packaging Information for the SIP; Representation Information for the SIP Content Data, including documented file format specifications; published data standards; documentation of valid object construction.

DoDAF views Conceptual Data Model (DIV-1), Standards Profile (StdV-1)

4.1.5 The repository shall have an ingest process which verifies each SIP for completeness and correctness.

Supporting Text This is necessary in order to detect and correct errors in the SIP when created and potential transmission errors between the depositor and the repository.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Appropriate Preservation Policy and Preservation Implementation Plan documents and system log files from system(s) performing ingest procedure(s); logs or registers of files received during the transfer and ingest process; documentation of standard operating procedures, detailed procedures, and/or workflows; format registries; definitions of completeness and correctness.

DoDAF views Systems Functionality Description (SV-4)

4.1.6 The repository shall obtain sufficient control over the Digital Objects to preserve them.

Supporting Text This is necessary in order to ensure that the preservation can be accomplished, with physical control, and is authorized, with legal control.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Documents showing the level of physical control the repository actually has. A separate database/metadata catalog listing all of the digital objects in the repository and metadata sufficient to validate the integrity of those objects (file size, checksum, hash, location, number of copies, etc.)

DoDAF views Services Context Description (SvcV-1)

4.1.8 The repository shall have contemporaneous records of actions and administration processes that are relevant to content acquisition.

Supporting Text This is necessary to ensure that such documentation, which may be needed in an audit, is captured and is accurate and authentic.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Written documentation of decisions and/or action taken; preservation metadata logged, stored, and linked to pertinent digital objects, confirmation receipts sent back to providers.

DoDAF views Services Context Description (SvcV-1)

4.2.2 The repository shall have a description of how AIPs are constructed from SIPs.

Supporting Text This is necessary in order to ensure that the AIPs adequately represents the information in the SIPs.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Process description documents; documentation of the SIP-AIP relationship; clear documentation of how AIPs are derived from SIPs.

DoDAF views Systems Functionality Description (SV-4)

4.2.3 The repository shall document the final disposition of all SIPs. In particular the following aspect must be checked.

Supporting Text This is necessary in order to ensure that the SIPs received have been dealt with appropriately, and in particular have not been accidentally lost.

Examples of Ways the Repository can Demonstrate it is Meeting these Requirements System processing files; disposal records; donor or depositor agreements/deeds of gift; provenance tracking system; system log files; process description documents; documentation of SIP relationship to AIP; clear documentation of how AIPs are derived from SIPs; documentation of standard/process against which normalization occurs; documentation of normalization outcome and how the resulting AIP is different from the SIP(s).

DoDAF views Systems Functionality Description (SV-4)

4.2.4 The repository shall have and use a convention that generates persistent, unique identifiers for all AIPs.

Supporting Text This is necessary in order to ensure that each AIP can be unambiguously found in the future. This is also necessary to ensure that each AIP can be distinguished from all other AIPs in the repository.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Documentation describing naming convention and physical evidence of its application (e.g., logs).

DoDAF views Conceptual Data Model (DIV-1), Logical Data Model (DIV-2), Physical Data Model (DIV-3)

4.2.5 The repository shall have access to necessary tools and resources to provide authoritative Representation Information for all of the digital objects it contains.

Supporting Text This is necessary in order to ensure that the repository's digital objects are understandable to the Designated Community.

Examples of Ways the Repository can Demonstrate it is Meeting these Requirements Subscription or access to registries of Representation Information (including format registries); viewable records in local registries (with persistent links to digital objects); database records that include Representation Information and a persistent link to relevant digital objects.

DoDAF views Conceptual Data Model (DIV-1), Logical Data Model (DIV-2), Physical Data Model (DIV-3)

4.2.6 The repository shall have documented processes for acquiring Preservation Description Information (PDI) for its associated Content Information and acquire PDI in accordance with the documented processes.

Supporting Text This is necessary in order to ensure that an auditable trail to support claims of authenticity is available, that unauthorized changes to the digital holdings can be detected, and that the digital objects can be identified and placed in their appropriate context.

Examples of Ways the Repository can Demonstrate it is Meeting these Requirements Standard operating procedures; manuals describing ingest procedures; viewable documentation on how the repository acquires and manages Preservation Description Information (PDI); creation of checksums or digests, consulting with Designated Community about Context.

DoDAF views Conceptual Data Model (DIV-1), Logical Data Model (DIV-2), Physical Data Model (DIV-3)

4.2.8 The repository shall verify each AIP for completeness and correctness at the point it is created.

Supporting Text This is necessary in order to ensure that what is maintained over the long term is as it should be and can be traced to the information provided by the Producers.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Description of the procedure that verifies completeness and correctness of the AIPs; logs of the procedure.

DoDAF views Systems Functionality Description (SV-4)

4.2.9 The repository shall provide an independent mechanism for verifying the integrity of the repository collection/content.

Supporting Text This is necessary to enable the audit of the integrity of the collection as a whole.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Documentation provided for 4.2.1 through 4.2.4; documented agreements negotiated between the producer and the repository (see 4.1.1-4.1.8); logs of material received and associated action (receipt, action, etc.) dates; logs of periodic checks.

DoDAF views Systems Functionality Description (SV-4)

4.3.2 The repository shall have mechanisms in place for monitoring its preservation environment.

Supporting Text This is necessary so that the repository can react to changes and thereby ensure that the preserved information remains understandable and usable by the Designated Community.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Surveys of the Designated Community of the repository.

DoDAF views Organizational Relationships Chart (OV-4)

4.4.1 The repository shall have specifications for how the AIPs are stored down to the bit level.

Supporting Text This is necessary in order to ensure that the information can be extracted from the AIP over the long term.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Documentation of the format of AIPs; EAST and Data Entity Dictionary Specification Language (DEDSL) descriptions of the data components.

DoDAF views Systems Functionality Description (SV-4), Physical Data Model (DIV-3)

4.4.1.2 The repository shall actively monitor the integrity of AIPs.

Supporting Text This is necessary in order to protect the integrity of the archival objects over time.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Fixity information (e.g., checksums) for each ingested digital object/AIP; logs of fixity checks; documentation of how AIPs and Fixity information are kept separate; documentation of how AIPs and accession registers are kept separate.

DoDAF views Conceptual Data Model (DIV-1), Logical Data Model (DIV-2), Physical Data Model (DIV-3)

4.5.1 The repository shall specify minimum information requirements to enable the Designated Community to discover and identify material of interest.

Supporting Text This is necessary in order to enable discovery of the repository's holdings.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Retrieval and descriptive information, discovery metadata, such as Dublin Core, and other documentation describing the object.

DoDAF views Conceptual Data Model (DIV-1), Logical Data Model (DIV-2), Physical Data Model (DIV-3)

4.6.1 The repository shall comply with Access Policies.

Supporting Text This is necessary in order to ensure the repository has fully addressed all aspects of usage which might affect the trustworthiness of the repository, particularly with reference to support of the user community.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement Statements of policies that are available to the user communities; information about user capabilities (authentication matrices); logs and audit trails of access requests; explicit tests of some types of access.

DoDAF views Services Rules Model (SvcV-10a)

4.6.2 The repository shall follow policies and procedures that enable the dissemination of digital objects that are traceable to the originals, with evidence supporting their authenticity.

Supporting Text This is necessary to establish an auditable chain of authenticity from the AIP to disseminated digital objects.

Examples of Ways the Repository Can Demonstrate It Is Meeting This Requirement System design documents; work instructions (if DIPs involve manual processing); process walkthroughs; production of a sample copy with evidence of authenticity; documentation of community requirements for evidence of authenticity.

DoDAF views Systems Functionality Description (SV-4)

A.3/ UML NOTATIONS

This section provides a brief summary of the Unified Modeling Language (UML) notation used in the two major UML diagrams: the class diagram, and the activity diagram.

Figure A.3 presents a self-describing class diagram: all the elements of the diagram are explained as comments.

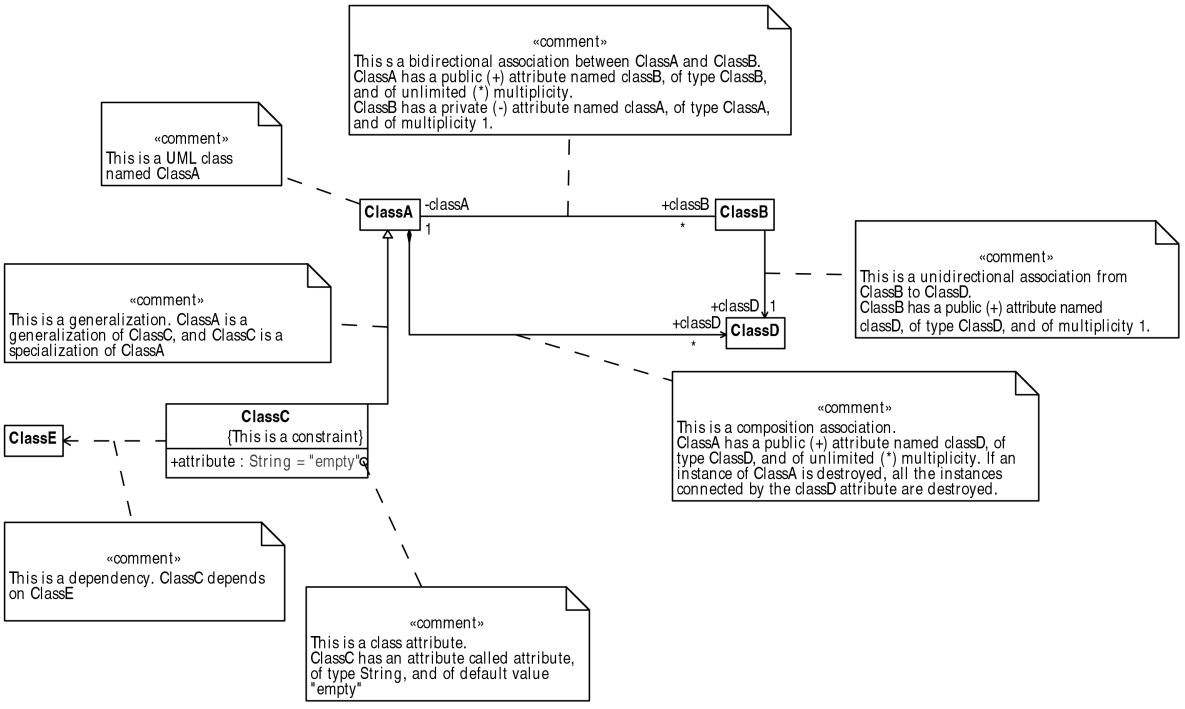

Figure A.2: UML Class diagram

Figure A.3 presents a self-describing activity diagram. The common elements of activity diagrams are also explained as comments.

A.4/ UML METAMODEL

This section presents the subset of UML necessary to understand Reference Architecture for Archival Systems (RAAS). This subset is depicted in Figure A.4. The diagram

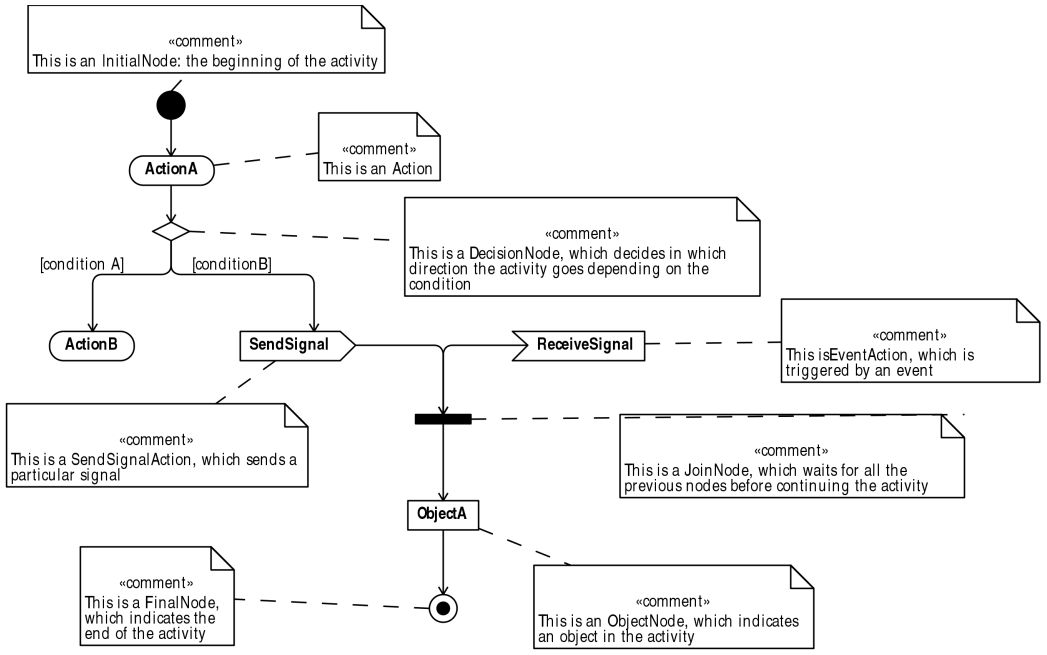

Figure A.3: UML Activity diagram

only shows the metaclasses, properties, and specializations necessary to understand and make consistent the models presented in Section 4.2. A complete definition of the metamodel is available in the UML 2 specification[51].

The root element of UML is the metaclass *Element*. *Elements* may have various *Constraints* defined.

NamedElement is a specialization of *Element*. *Dependencies* have as client and supplier exactly one *NamedElement*.

RedefinableElement, *PackageableElement*, and *TypedElement* are three specializations of *NamedElement*.

StructuralFeature is a specialization of *TypedElement*, and *Property* is a specialization of *StructuralFeature* and of *ConnectableElement*.

Type is a specialization of *PackageableElement*, and a *TypedElement* may have a *Type*.

Classifier is a specialization of *PackageableElement*, and a *Classifier* may have multiple *Generalizations*, which refer to a general *Classifier*.

InformationFlow is a specialization of *PackageableElement*, and an *InformationFlow* can convey a *Classifier*.

StructuredClassifier, *DataType*, and *Interface* are specializations of *Classifier*.

A *StructuredClassifier* may have multiple *Connector*, which have multiple *ConnectorEnds*. A *ConnectorEnd* has a *StructuralFeature* as role.

EncapsulatedClassifier is a specialization of *StructuredClassifier*. An *EncapsulatedClassifier* may have multiple *ports*, which are specializations of *Properties*.

Class is a specialization of *EncapsulatedClassifier*. *Classes*, *DataTypes*, and *Interfaces*

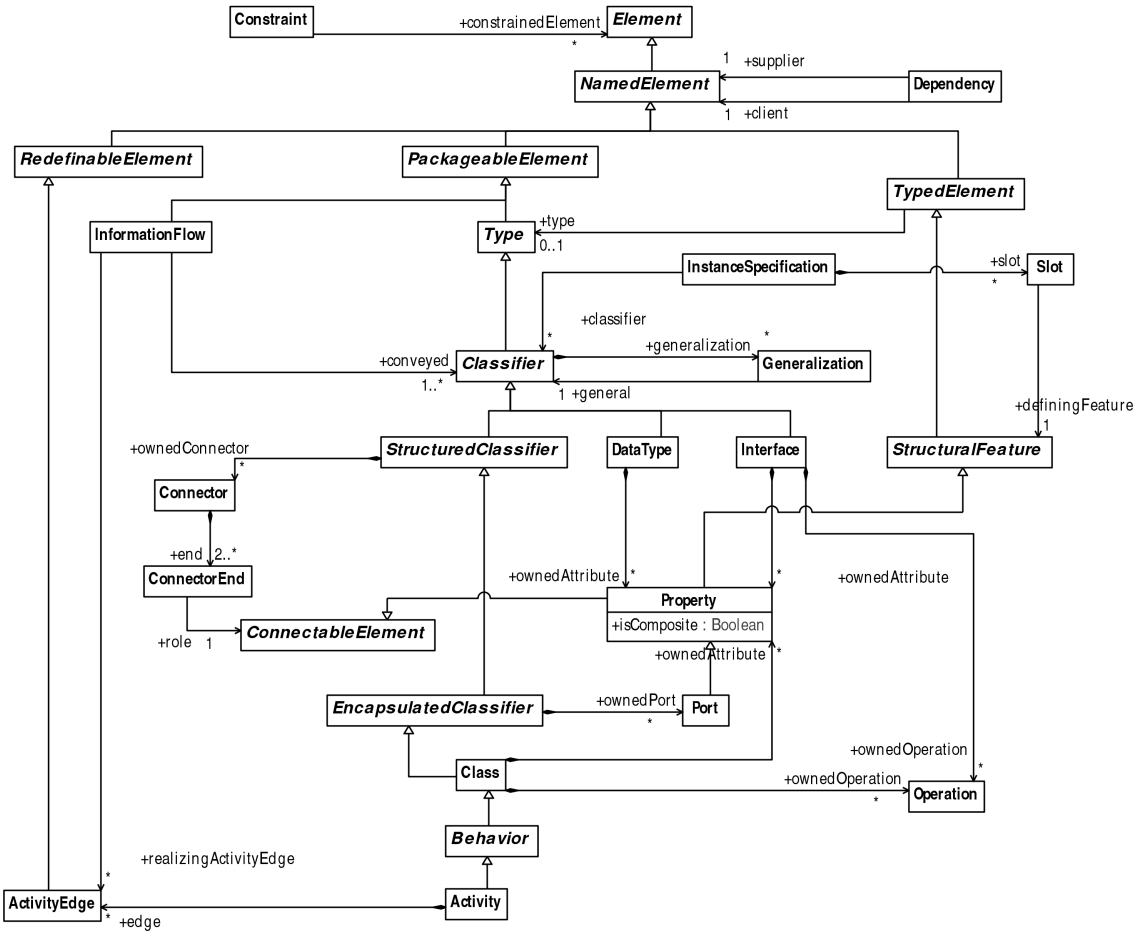

Figure A.4: Subset of UML used in this thesis

may have multiple *Property* as attributes. Moreover, *Classes* and *Interfaces* may have multiple *Operations*.

Behavior is a specialization as *Class*, and *Activity* is a specialization of *Class*. An *Activity* may have multiple *ActivityEdges*, which are *RedefinableElements*. An *ActivityEdge* may be the realization of an *InformationFlow*.

Finally, an *InstanceSpecification* may have multiple *Classifiers*, and may own multiple *Slots*. A *Slot* defines a *StructuralFeature*.

A.5/ PREPARATION OF SHIP PRODUCT MODELS FOR PRESERVATION

Chapter 5 presented the architectural description of a product model archival system. This section shows how some parts of this architectural description were actually executed. This section shows in detail how the data related to a gate valve is prepared for preservation. This preparation corresponds to the ingest activity, described in Figure 5.23. This section is split into four parts: the presentation of the target data, the presentation of the conversion process to obtain this data, the description of the SIP, and explanations on how to access this data for a maintenance use case.

Description of the LEAPS files The starting point is the selection of the data from the Torpedo Weapon Retriever (TWR) dataset. The data selected is a small and coherent subset, which relates to a particular gate valve. The following paragraphs will describe each file and will show what actual metadata is expected for this particular example.

In the figures, the documents represented refer to digital objects, which are the target of the preservation.

LEAPS data The Leading Edge Architecture for Prototyping Systems (LEAPS) data, which contains Product Data Management (PDM) information, is available as several Excel files. One of these files, named “Result_STRUCT_vAll_Struct_Parts.xlsx”, lists various systems that compose the ship, and the parts that compose each system. The gate valve used in this example is part of the fuel oil system. The gate valve is referred to as V15, and it has the following description: ”VALVE GATE 2 1/2IN FLGD150LB DI W/BRNZ TRIM ANSI B16.10 FF”. This description stands for a 2.5in gate valve flanged, able to support a pressure of 150lb, with bronze trim, and conform to ANSI B16.10. Three occurrences of this gate valve in the fuel oil system are present: 2110-FUELOIL-V816, 2110-FUELOIL-V821, 2110-FUELOIL-V839.

To be preserved, the information present in the Excel file is converted into a standard format: Application Protocol 239 – Product life cycle support (AP239). To do so, a software was created to parse the Excel file, and to instantiate Product Lifecycle Support (PLCS) templates according to the data available in the Excel file. This part could be automated.

Figure A.5 shows the data model for the PDM information concerning the gate valve. This is a particular instantiation of the generic model shown in Figure 5.15. The system containing the gate valve is defined as being a breakdown of the ship, and the valve is assigned as a realization of a element of this system.

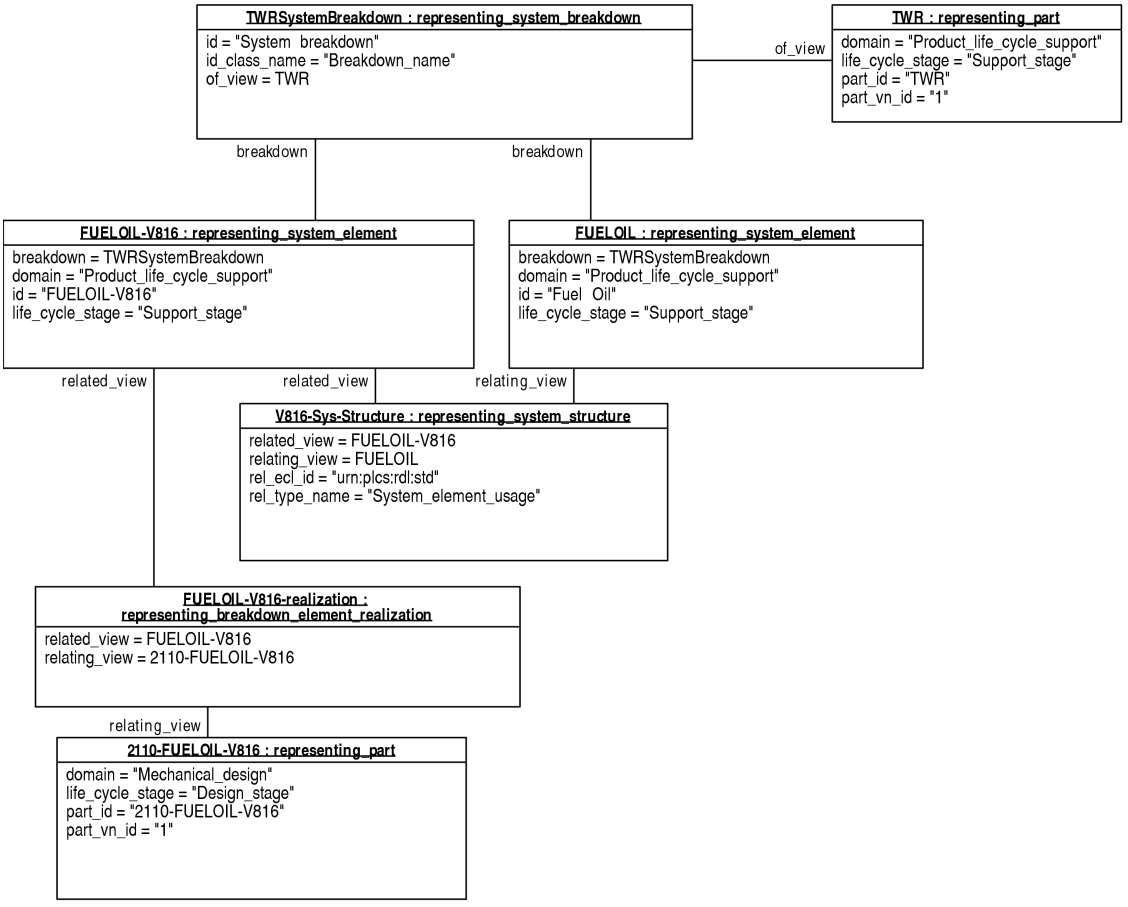

Figure A.5: PDM information

CAD files There are various Computer-Aided Design (CAD) files in the dataset. Two models are related to the gate valve: one that describes the entire fuel oil system, and one that describes the gate valve only. Each file is stored in both proprietary format and in Application Protocol 214 – Core data for automotive mechanical design processes (AP214). The file representing the system is named “fuelOilPipeAndEqpt_1.stp”, and the file representing the valve is called “2110-FUELOIL-V816.stp”.

Figure A.6 shows the metadata for the fuel oil system CAD representation. This figure is a particular instantiation of the model shown in Figure 5.17. It shows the reference to the existing fuel oil system, and the assignment of a document — the CAD file — to the system.

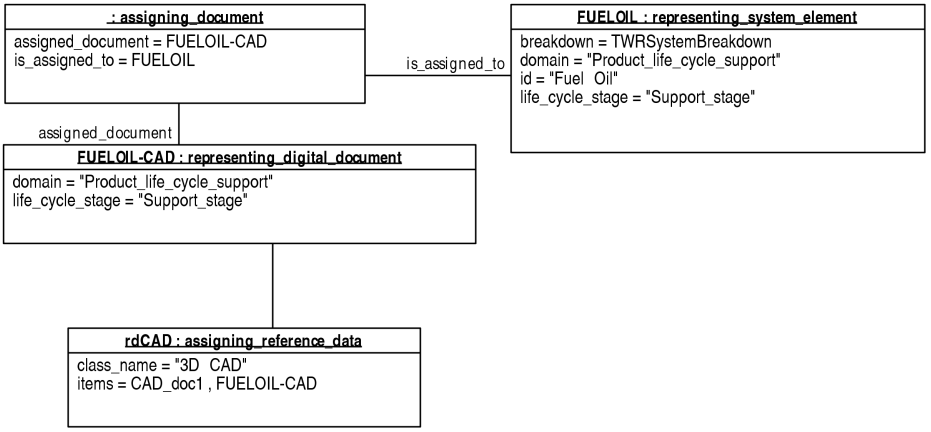

Figure A.6: Metadata for the Fuel Oil System CAD file

Figure A.7 shows the metadata for the gate valve CAD representation. This follows the generic instantiation shown in Figure 5.17. This figure shows the reference to the existing part, and the assignment of the CAD file to the part.

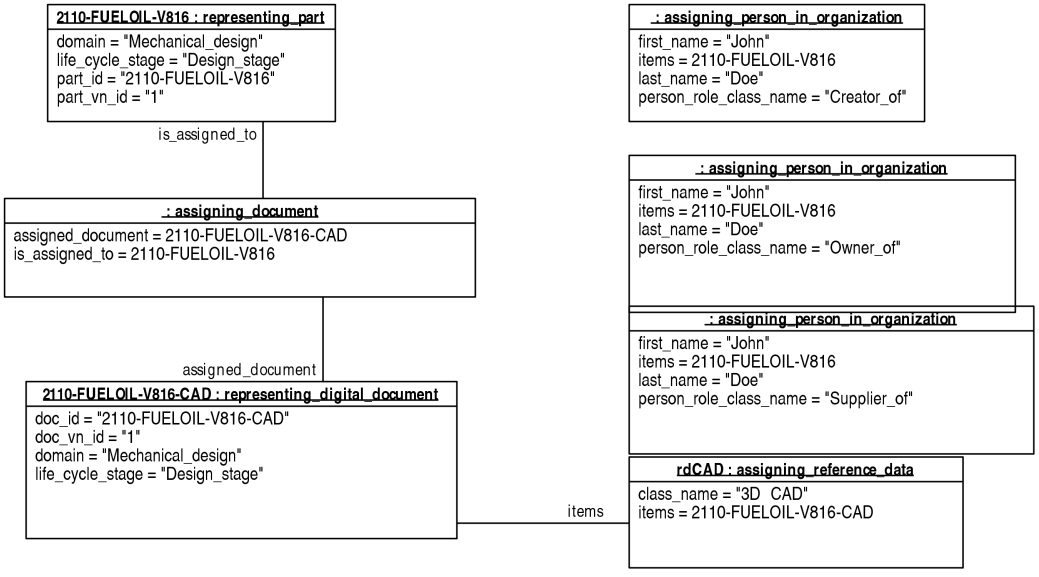

Figure A.7: Metadata for the valve CAD file

Photo The dataset contains photos of the parts installed in the ship. One of them show the installed gate valve and its environment. The file is named “fos.png”.

Figure A.8 shows the metadata for the part picture.This is a particular instantiation of the model shown in Figure 5.18. The picture is simple assigned to the part it depicts.

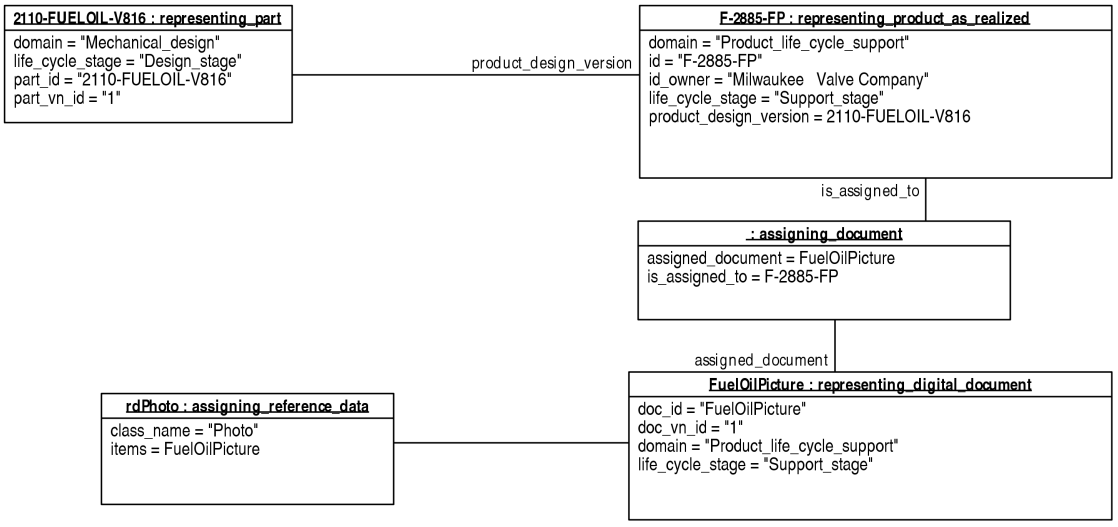

Figure A.8: Metadata for the picture of the valve

Maintenance procedure One of the main file in the dataset is the boat information booklet, which defines various descriptions and procedures to operate and maintain the ship. This file is called “TWR_120_BoatInformationBook.pdf”. It contains, among other things, the procedure for installing, maintaining, replacing, and turning on and off different systems.

Figure A.9 shows the metadata for the system maintenance procedure. This is a particular instantiation of the data model shown in Figure 5.20. The maintenance manual is assigned to the maintenance activity of the part.

Specifications of the part The last file contains the specification of a commercial gate valve. The file is named “F_2885_FP_Rev_1.pdf”.

Figure A.10 shows the metadata for the specification. This is a particular instantiation of the data model shown in Figure 5.19. The specification is assigned to the realized part, which relates to the gate valve as a designed part.

PLCS implementation and conversion of the LEAPS data Because this chapter targets product models, the conversion of LEAPS data concerns only the CAD files, the PDM data, which includes the metadata for photos, specifications, and manuals. Only the CAD files are in Standard for the Exchange of Product model data (STEP) format. The PDM data needs to be either generated or converted into STEP.

As explained earlier, PLCS is used as data model to represent the PDM information. PLCS is a large information model that is too large to be fully implemented in a data

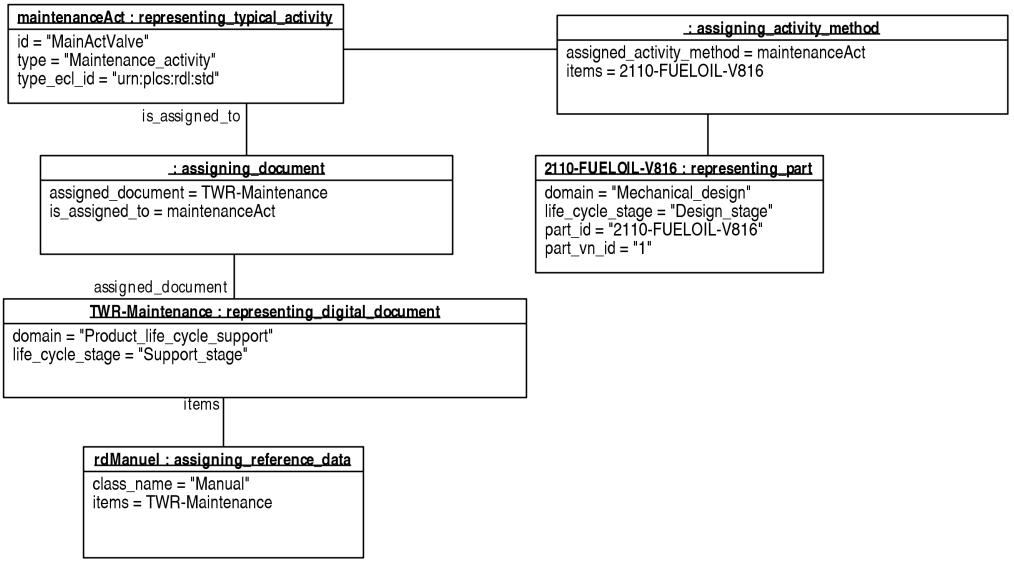

Figure A.9: Metadata for the maintenance manual

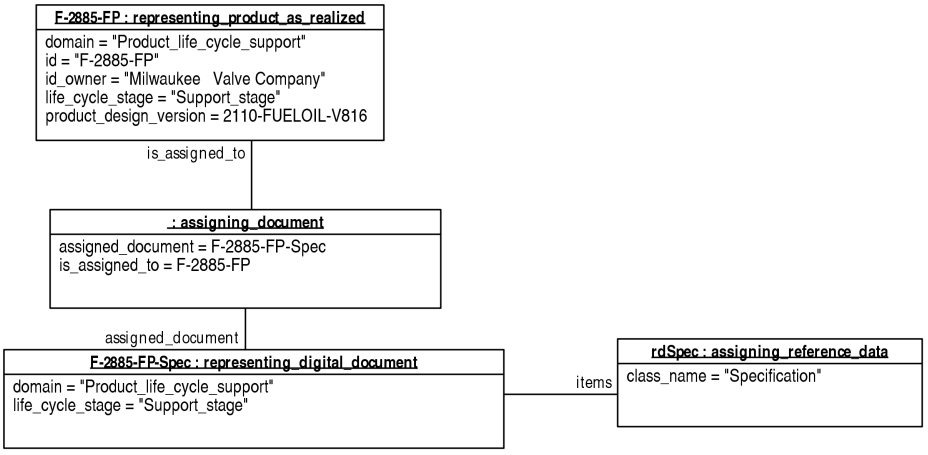

Figure A.10: Metadata for the valve specification

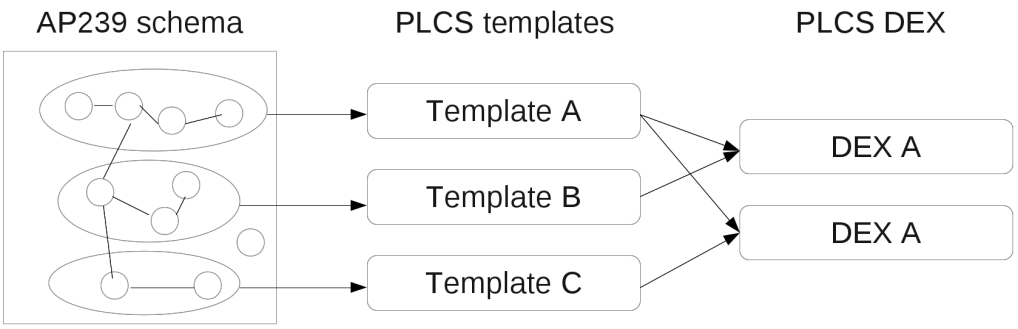

Figure A.11: PLCS architecture

exchange scenario, so conformance to the entire PLCS is of little use. An actual data exchange would cover only a subset of the information model, called a Data EXchange Specification (DEX). With the notion of DEX, it is possible to define conformance to subsets of PLCS. The development of PLCS is split into two entities: the International Organization for Standardization (ISO) is in charge of managing the whole AP239 schema, and the Organization for the Advancement of Structured Information Standards (OASIS) is in charge of providing an architecture to define DEXs.

The first architecture delivered by the OASIS for the creation of DEXs is called DEXlib. DEXlib introduces building blocks called templates. A template defines a instantiation of entities and attributes. A DEX can be seen as a set of templates, and templates can be reused in multiple DEXs (see Figure A.11). Templates provide an abstract view of the information model, which facilitates the understanding of the model and also the data generation.

A template is composed of different things:

- Input parameters are values given in input for the instantiation of entities
- The instantiation path uses the input parameters to instantiate entities or other templates
- Reference parameters give reference to the entities created by the template. These references can then serve an input parameters for other templates
- Unique rules makes it possible to reuse entities already created, and hence avoid creating entities that represent the same thing

A DEX also has a Reference Data Library (RDL), which defines concepts that STEP files can refer to. This is a way to specialize the entities defined in the AP239 schema.

In DEXlib, PLCS templates are formally defined in XML files. Because templates are formally defined, it is possible to automatically process them to accomplish different purpose. In this use case, templates were processed to accomplish two different purpose:

- Generate an Application Programming Interface (API) that implement the architecture in Java
- Generate UML models

The generation of UML models facilitates the visualization of the information, and these models were actually used in the architectural description.

The creation of an API facilitates the generation of PLCS data, as templates abstract a particular instantiation patterns. The creation of an API in Java from templates is defined

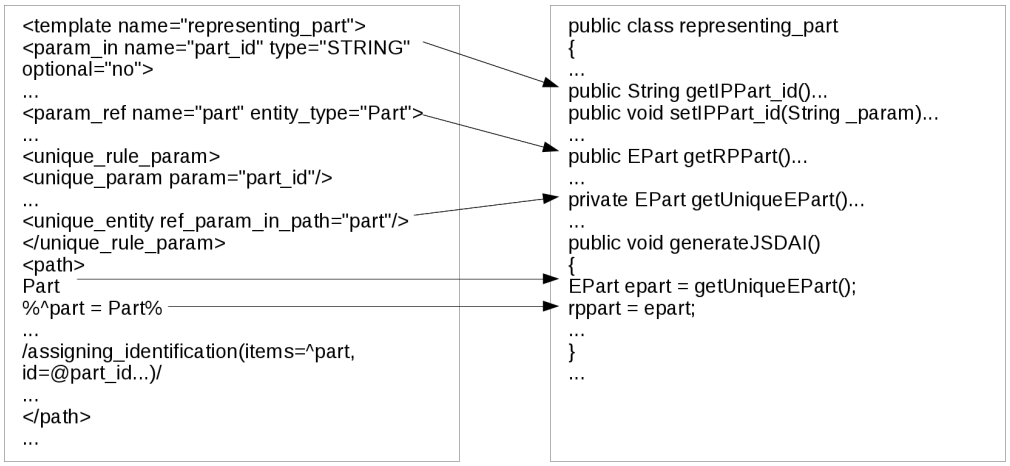

Figure A.12: Mapping of PLCS template in Java

as follows (see Figure A.12):

- A Java package is created for a DEX
- A template is converted into a Java class
- Input parameters were converted into attributes, which can be set
- The instantiation paths are parsed and converted into the equivalent instantiation in Java, using the JSDAI library[1].
- The created entities can be saved into an AP239 instance file

The resulting API was used to generate PLCS data, based on the templates defined in DEXlib. The LEAPS data, available as a spreadsheet, was parsed to automatically generate AP239 files using the API. Similarly, the metadata for the various files was manually generated using this API.

Some of the templates used in this chapter (e.g. systems and interface definitions) are taken from a DEX called ShipLTDR[2], created by the Navy to support the long-term preservation of ship product model data. The rest of the templates (activities, realized product) are added to define more precisely to which PDM concept the product data was referring to.

Finally, once PDM information and the product models serving as metadata are created, the STEP data was translated into Web Ontology Language (OWL) to enable semantic enrichment. The process and the benefits related to this translation are available in Chapter 6.

Creation of the LEAPS SIP Once the data and metadata are ready, the SIPs can be created and sent to the archive. As explained earlier, these SIPs follow the Fedora Object XML (FOXML) format. Due to the way FOXML works, there are almost no differences between the SIP and the AIP. A SIP corresponds to one AIP. If a new version of the digital object was added to an existing version, the content of the SIP would complement the existing AIP.

Figure A.13 shows a representation of a SIP generated from this example. The class

[1] http://www.jsdai.net

[2] http://www.plcs-resources.org/plcs/dexlib/data/busconcept/US_Navy/ShipLTDR/sys/cover.htm

diagram corresponding to this example is shown in Figure 5.58This SIP presents a digital object containing the LEAPS information, with three different datastreams. The first corresponds to the raw data, the second corresponds to Dublin Core information, and the third corresponds to the metadata that is extracted from data. This metadata can then be referred to by other types digital files.

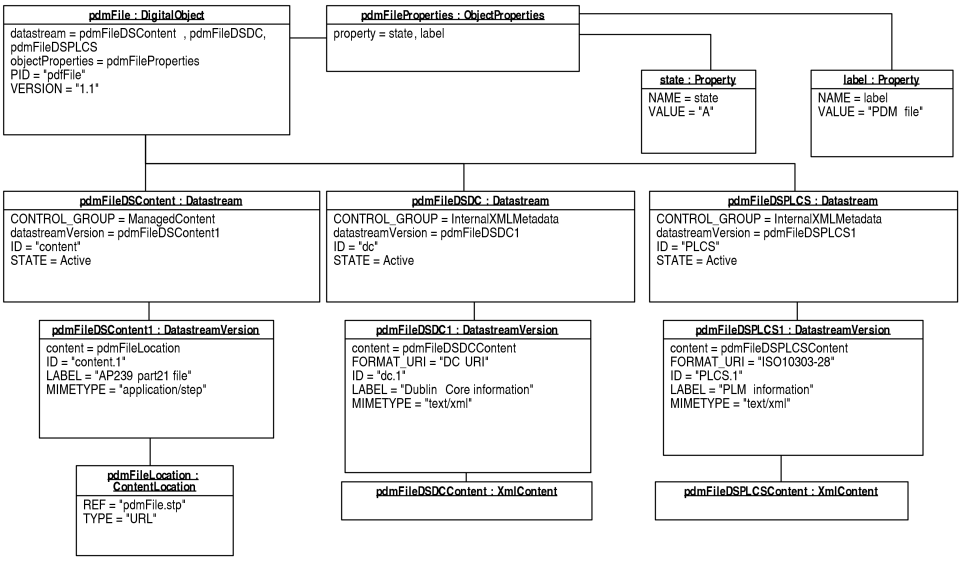

Figure A.13: Composition of a SIP for the TWR data

Future access in a context of maintenance Different types of product information can be available by accessing the archive. The following examples show how to use the product models defined in this use case to support various steps in the maintenance of the gate valve.

The first step is to identify the part marked for maintenance. The software in charge of managing the maintenance activities maintains the list of the parts that are actually installed on the ship. A product can be seen as two different things: the product as designed, and the product as realized, which represent the part actually maintained.

The metadata defined in Figure A.8 contains a document, which is a photo assigned to a particular realized part. The connection between the realized part and the designed part is established, which indicated that this realized part can fulfill the role of the designed part. So, from the part identification, it is possible to access a photo to see how the part look like.

Similarly, Figure A.7 defines the metadata that connects the part to its CAD representation. Maintainers can use it to see how the part look like.

If a more global view is need to locate the part, it is possible to have the CAD representation of the system containing the part. The metadata shown in Figure A.5 connects the part to the system it belongs to, and the metadata shown in Figure A.6 associate the CAD file to the system.

The metadata defined in Figure A.9 presents a document that corresponds to the maintenance manual, and that relates to an activity associated with the part. So, from the part

identification, it is possible to access the corresponding maintenance activity, and retrieve the document associated with this activity.

If the part needs to be replaced, its specifications can be retrieved using the metadata shown in Figure A.10, so that similar parts can be ordered. Or, if the parts has to be manufactured, the CAD representation can be retrieved and sent to the manufacturer.

BIBLIOGRAPHY

[1] International Organization for Standardization. ISO 14721:2012, Space data and information transfer systems – Open archival information system (OAIS) – Reference model, 2012. URL http://public.ccsds.org/publications/archive/650x0m2.pdf.

[2] E.A.W. Budge. *The Rosetta Stone*. Dover Pubns, 1989.

[3] T. Kuny. The digital dark ages? challenges in the preservation of electronic information. *International Preservation News*, pages 8–13, 1998.

[4] International Organization for Standardization. ISO 10303-1: 1994, Industrial automation systems and integration – Product data representation and exchange – Part 1: Overview and fundamental principles, 1994.

[5] Federal Aviation Administration. Order 8110.4c: Type certification. URL http://www.faa.gov/regulations_policies/orders_notices/index.cfm/go/document.information/documentID/15172.

[6] Aerospace Industries Association (AIA) and the Aerospace and Defense Industries Association of Europe for Standardization (ASD-STAN). pren/nas 9300-001. URL http://www.lotar-international.org/lotar-standard.html.

[7] S. Rachuri, E. Subrahmanian, A. Bouras, S.J. Fenves, S. Foufou, and R.D. Sriram. Information sharing and exchange in the context of product lifecycle management: Role of standards. *Computer-Aided Design*, 40(7):789–800, 2008.

[8] Joshua Lubell, Sudarsan Rachuri, and Mahesh Mani. Sustaining engineering informatics: Toward methods and metrics for digital curation. *The International Journal of Digital Curation*, 3(2):59–73, July 2008. URL http://www.ijdc.net/index.php/ijdc/article/view/87/58.

[9] Vijay Srinivasan. Step in the context of product data management. In Xun Xu and Andrew Y. C. Nee, editors, *Advanced Design and Manufacturing Based on STEP*, Springer Series in Advanced Manufacturing, pages 353–381. Springer London, 2009. ISBN 978-1-84882-739-4. URL http://dx.doi.org/10.1007/978-1-84882-739-4_16.

[10] Xenia Fiorentini and Sudarsan Rachuri. STEP-OAGIS Harmonization Joint Working Group, PDM Subgroup Interim Report. Technical report, NIST, 2009. URL http://www.oagi.org/oagi/downloads/ResourceDownloads/2009_OAGIS_STEP_Final.pdf.

[11] C.R. Snyder, C.A. Snyder, and C.S. Sankar. Use of information technologies in the process of building the boeing 777. *Journal of Information Technology Management*, 9:31–42, 1998.

[12] S.J. Fenves, S. Foufou, C. Bock, and R.D. Sriram. CPM2: a core model for product data. *Journal of Computing and Information Science in Engineering*, 8(1): 014501/1–014501/6, 2008.

[13] M. M. Baysal, U. Roy, R. Sudarsan, R. D. Sriram, and K. W. Lyons. The open assembly model for the exchange of assembly and tolerance information: overview and example. In *Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, 2004.

[14] John S Gero and Udo Kannengiesser. The situated function–behaviour–structure framework. *Design studies*, 25(4):373–391, 2004.

[15] M Labrousse and A Bernard. *FBS-PPRE, an enterprise knowledge lifecycle model*, pages 285–305. Springer, 2008.

[16] Jeffrey van der Hoeven, Bram Lohman, and Remco Verdegem. Emulation for digital preservation in practice: The results. *International Journal of Digital Curation*, 2(2): 123–132, december 2007. URL http://www.ijdc.net/index.php/ijdc/article/view/50/35.

[17] M.J. Pratt. Introduction to iso 10303 — the step standard for product data exchange. *Journal of Computing and Information Science in Engineering*, 1:102, 2001.

[18] LongView Advisors. Collaboration & interoperability market report 2008, 2008.

[19] Ben Kassel and Patrick David. Long term retention of product model data. *Journal of Ship Production*, 23(2):118–124, May 2007.

[20] Ben Kassel and Ted Briggs. An alternate approach to the exchange of ship product model data. *Journal of Ship Production*, 24(2):92–98, May 2008. URL http://www.isetools.org/yabbfiles_ISE/Attachments/P18.pdf.

[21] Ted L. Briggs, Burt Gischner, Pete Lazo, Mike Olson, Jim Vicedomine, and Ken Wolsey. Enabling interoperability through the ship life cycle. *Journal of Ship Production*, 25(1):33–44, February 2009.

[22] Integrated Shipbuilding Environment Consortium (ISEC). Ise-6 final report. Technical report, Integrated Shipbuilding Environment Consortium (ISEC), 2009. URL http://www.nsrp.org/Project_Information/major_projects/deliverables/ase_0608001.pdf.

[23] Michal Ondrejcek, Jason Kastner, Rob Kooper, and Peter Bajcsy. Information extraction from scanned engineering drawings. Technical report, National Center for Supercomputing Applications, 2009. URL http://isda.ncsa.uiuc.edu/peter/publications/techreports/2009/NCSA-ISDA-2009-001.pdf.

[24] O. Lassila and R.R. Swick. Resource description framework (RDF) model and syntax specification, 1999.

[25] Manjula Patel, Alexander Ball, and Lian Ding. Strategies for the curation of cad engineering models. *International Journal of Digital Curation*, 4(1):84–97, 2009. ISSN 1746-8256. URL http://www.ijdc.net/index.php/ijdc/article/view/104/.

[26] William C. Regli, Michael Grauer, and Joseph B. Kopena. A framework for preservable geometry-centric artifacts. In *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, pages 67–78, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-711-0. doi: http://doi.acm.org/10.1145/1629255.1629265. URL http://gicl.cs.drexel.edu/people/tjkopena/publications/regli-spm2009.pdf.

[27] C. Schlenoff, M. Gruninger, F. Tissot, J. Valois, J. Lubell, and J. Lee. The Process Specification Language (PSL) Overview and Version 1.0 Specification.

[28] X. Fiorentini, I. Gambino, V. Liang, S. Foufou, R. Sudarsan, M. Mani, and C. E. Bock. An ontology for assembly representation. Technical report, National Institute of Standards and Technology, 2007. NISTIR 7436.

[29] L. Klein, G. Liutkus, V. Nargelas, P. Sileikis, T. Baltramaitis, B. Schowe-von der Brelie, A. Alfter, and C. Wesbuer. Ontologies derived from step data models. Technical report, S-TEN, 2008. URL http://www.s-ten.net/.

[30] W. Zhao and J. K. Liu. Owl/swrl representation methodology for express-driven product information model. *Computers in Industry*, 59:580–600, 2008.

[31] Consultative Committee for Space Data Systems. Audit and Certification of Trustworthy Digital Repositories, October 2009. URL http://public.ccsds.org/sites/cwe/rids/Lists/CCSDS6520R1/Attachments/652x0r1.pdf.

[32] A.J. Gilliland-Swetland and P.B. Eppard. Preserving the authenticity of contingent digital objects: the interpares project. 2000.

[33] L. Duranti. The interpares 2 project (2002-2007): An overview. *Archivaria*, 64(0), 2008.

[34] Verband der Automobilindustrie e. V. VDA 4958: Long-Term Archiving of digital Product Data, which are not based on technical drawings. Technical report, Verband der Automobilindustrie e. V., 2005. URL http://vda.de/en/publikationen/publikationen_downloads/index.html.

[35] Joshua Lubell, Sudarsan Rachuri, Eswaran Subrahmanian, and Willian Regli. Long term knowledge retention workshop summary. Technical report, NIST, 2006. URL http://www.ukoln.ac.uk/events/ltkr-2007/ltkr-workshop-summary2006.pdf.

[36] A. Ball and L. Ding, editors. *Proceedings of the Atlantic Workshop on Long-Term Knowledge Retention*, 2007. University of Bath. URL http://www.ukoln.ac.uk/events/ltkr-2007/proceedings/awltkr2007-proceedings.html.

[37] Joshua Lubell. Metadata for long term preservation of product data. In *Proceedings of the International Symposium on XML for the Long Haul: Issues in the Long-term Preservation of XML*, volume 6 of *Balisage Series on Markup Technologies*, 2010. doi: 10.4242/BalisageVol6.Lubell01. URL http://www.balisage.net/Proceedings/vol6/html/Lubell01/BalisageVol6-Lubell01.html.

[38] Joshua Lubell, Ben Kassel, and Sudarsan Rachuri. Descriptive metadata requirements for long-term archival of digital product models. In *Proceedings of the Indo-US Workshop on International Trends in Digital Preservation*, 2009.

[39] Jörg Brunsmann and Wolfgang Wilkes. Enabling product design reuse by long-term preservation of engineering knowledge. *The International Journal of Digital Curation*, 3(4):17–28, december 2009. URL http://www.ijdc.net/index.php/ijdc/article/view/131/149.

[40] G. Pierra, E. Sardet, J. C. Potier, G. Battier, J. C. Derouet, Willmann N., and Mahir A. Exchange of component data: the plib (iso 13584) model, standard and tools. In *Proceedings of the CALS EUROPE'98 Conference*, pages 160–176, 1998.

[41] Wolfgang Wilkes, Jörg Brunsmann, Dominic Heutelbeck, Andreas Hundsdörfer, Matthias Hemmje, and Hans-Ulrich Heidbrink. Towards support for long-term digital preservation in product lifecycle management. *The International Journal of Digital Curation*, 6(1):282–296, March 2011. URL http://www.cdlib.org/services/uc3/iPres/presentations/Wilkes.pdf.

[42] S.A. Bernard. *An introduction to enterprise architecture*. AuthorHouse, 2005.

[43] *ISO/IEC 42010 Systems and software engineering — Architecture description*. International Organization for Standardization, 2007.

[44] Roger Sessions. A comparison of the top four enterprise architecture methodologies, May 2007. URL http://www.objectwatch.com/whitepapers/4EAComparison.pdf.

[45] J. Schekkerman. *How to survive in the jungle of enterprise architecture frameworks: Creating or choosing an enterprise architecture framework*. Trafford, 2006.

[46] J. A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 26(3):276 –292, 1987. ISSN 0018-8670. doi: 10.1147/sj.263.0276.

[47] The Open Group, editor. *TOGAF Version 9.1, Enterprise Edition*. Van Haren Publishing, 2011. URL http://pubs.opengroup.org/architecture/togaf9-doc/arch/.

[48] Department of Defense Architecture Framework (DoDAF) 2.0. URL http://dodcio.defense.gov/sites/dodaf20/.

[49] Ministry of Defence Architecture Framework (MODAF) 1.2. URL http://www.mod.uk/DefenceInternet/AboutDefence/WhatWeDo/InformationManagement/MODAF/.

[50] Unified Profile for the Department of Defense Architecture Framework (DoDAF) and the Ministry of Defence Architecture Framework (MODAF). URL http://www.updm.com/index.htm.

[51] Unified Modeling Language (UML) 2.0. URL http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML.

[52] Systems Modeling Language (SysML) 1.2. URL http://www.omg.org/technology/documents/modeling_spec_catalog.htm.

[53] IFIP-IFAC Task Force. IFIP-IFAC task force on architectures for integrating manufacturing activities and enterprises. In *IFIP newsletter/IFAC newsletter*. 1993.

[54] P. Bernus, L. Nemes, and G. Schmidt. *Handbook of Enterprise Architecture*, pages 40–82. Springer Verlag, 2003. IFIP-IFAC Task Force, GERAM: The Generalized Enterprise Reference Architecture and Methodology.

[55] TJ Williams, P. Bernus, J. Brosvic, D. Chen, G. Doumeingts, L. Nemes, JL Nevins, B. Vallespir, J. Vlietstra, and D. Zoetekouw. Architectures for integrating manufacturing activities and enterprises. *Computers in Industry*, 24(2-3):111–139, 1994.

[56] International Organization for Standardization. ISO/IS 15704:2000, industrial automation systems requirements for enterprise reference architectures and methodologies, 2000.

[57] T.J. Williams. The Purdue enterprise reference architecture. *Computers in industry*, 24(2-3):141–158, 1994.

[58] A CIMOSA presentation of an integrated product design review framework. *Int. J. Computer Integrated Manufacturing*, 84(4):260–278, 2005.

[59] G. Doumeingts, D. Chen, B. Vallespir, P. Fenie, and F. Marcotte. GIM (GRAI Integrated Methodology) and its evolutions-A methodology to design and specify Advanced Manufacturing Systems. In *Proceedings of the JSPE/IFIP TC5/WG5. 3 Workshop on the Design of Information Infrastructure Systems for Manufacturing*, pages 101–120. North-Holland Publishing Co., 1993.

[60] C. Becker, G. Antunes, J. Barateiro, R. Vieira, and J. Borbinha. Modeling digital preservation capabilities in enterprise architecture. In *In 12th Annual International Conference on Digital Government Research (dg. o 2011), June 12-15, College Park, MD, USA*, 2011.

[61] *NATO Architecture Framefork 3.0*. NATO, 2007.

[62] Office of the DoD CIO. Reference architecture description. Technical report, Department of Defense, 2010.

[63] OASIS. Reference architecture for service oriented architecture version 1.0, public review draft 1, 2008.

[64] M. Gogolla and B. Henderson-Sellers. Analysis of UML Stereotypes within the UML Metamodel. *UML 2002—The Unified Modeling Language*, pages 63–81, 2002.

[65] Dublin Core Metadata Initiative. Dublin core metadata element set, version 1.1, 2012.

[66] Daniel Davis. Fedora Object XML (FOXML), 2011. URL http://fedora-commons.org/download/2.0/userdocs/digitalobjects/introFOXML.html.

[67] C. Lagoze, S. Payette, E. Shin, and C. Wilper. Fedora: an architecture for complex objects and their relationships. *International Journal on Digital Libraries*, 6(2):124–138, 2006.

[68] M.K. Smith, M. Barton, M. Bass, M. Branschofsky, G. McClellan, D. Stuve, R. Tansley, and J.H. Walker. DSpace: An open source dynamic digital repository. 2003.

[69] International Organization for Standardization. ISO 10303-239:2005. Industrial automation systems and integration – Product data representation and exchange – Part 239: Application protocol: Product life cycle support, 2005.

[70] Organization for the Advancement of Structured Information Standards. Oasis product life cycle support (plcs) technical committee, 2003. URL https://www.oasis-open.org/committees/plcs/charter.php.

[71] ASD-STAN. LOTAR - LOng Term Archiving and Retrieval of digital technical product documentation such as 3D, CAD and PDM data. URL http://www.lotar-international.org.

[72] F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. *Mechanizing Mathematical Reasoning*, pages 228–248, 2005.

[73] T.B. Lee, J. Hendler, and O. Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 5(1), 2001.

[74] S. Jolly and J. Sredevi. The semantic web: An overview. 2006.

[75] C. Arms and C. Fleischhauer. Digital formats: Factors for sustainability, functionality, and quality. In *IS& T Archiving Conference, Society for Imaging Science and Technology, Washington, DC*, pages 26–29, 2005.

[76] X. Fiorentini, S. Rachuri, M. Mahesh, S. Fenves, and R.D. Sriram. Description logic for product information models. In *Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, 2008.

[77] S. Krima, R. Barbau, X. Fiorentini, S. Rachuri, and R.D. Sriram. OntoSTEP: OWL-DL Ontology for STEP. Technical report, National Institute of Standards and Technology, 2009. NISTIR 7561.

[78] International Organization for Standardization. Iso 10303-21: 2002, industrial automation systems and integration – product data representation and exchange – part 21: Implementation methods: Clear text encoding of the exchange structure, 2002.

[79] International Organization for Standardization. Iso 10303-28:2007 industrial automation systems and integration – product data representation and exchange – part 28: Implementation methods: Xml representations of express schemas and data, using xml schemas, 2007.

[80] J. Tao, E. Sirin, J. Bao, and D.L. McGuinness. Integrity constraints in owl. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, 2010.

[81] World Wide Web Consortium. OWL 2 Web Ontology Language, 2012. URL http://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

[82] P. Hitzler, M. Krotzch, and S. Rudolph. *Foundations of Semantic Web Technologies*. CRCPress, 2009.

[83] R. D. Sriram. *Intelligent Systems for Engineering: A Knowledge-Based Approach*. Springer, 1997.

[84] D. Koonce, L. Huang, and R. Judd. EQL an EXPRESS Query Language. *Computers & Industrial Engineering*, 35(1–2):271–274, 1998.

[85] SPARQL Query Language for RDF, 2008. URL http://www.w3.org/TR/rdf-sparql-query/.

[86] SQWRL: a query language for OWL, 2009. URL http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL.

[87] Pellet, a reasoner for OWL, 2008. URL http://clarkparsia.com/pellet/.

[88] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, 2004. URL http://www.w3.org/Submission/SWRL/. W3C Member Submission.

[89] Sandia National Laboratories. Jess engine, 2008. URL http://herzberg.ca.sandia.gov/.

[90] The Protégé OWL Editor, 2008. URL http://protege.stanford.edu.

[91] J. Lubell and S. Lardet. Open source EXPRESS parser, 2001. URL http://sourceforge.net/projects/osexpress/.

[92] Antlr parser generator, 2008. URL http://www.antlr.org/.

[93] University of Manchester. Owl api, 2008. URL http://owlapi.sourceforge.net.

[94] Object Management Group. Meta Object Facility Core Specification Version 2.4.1, 2011.

[95] Object Management Group. Ontology Definition Metamodel, 2008.

[96] Object Management Group. Reference metamodel for the EXPRESS information modeling language RFC, 2008.

[97] Object Management Group. Meta Object Facility 2.0 Query/View/Transformation Specification, 2011.

[98] W3C. Shared ontologies. URL http://www.w3.org/TR/webont-req/#goal-shared-ontologies.

[99] A.G. Kleppe, J.B. Warmer, and W. Bast. *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional, 2003.

[100] A. Rajasekar, R. Moore, C.Y. Hou, C.A. Lee, R. Marciano, A. de Torcy, M. Wan, W. Schroeder, S.Y. Chen, L. Gilbert, et al. iRODS primer: integrated rule-oriented data system. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2(1):1–143, 2010.

[101] C. Baru, R. Moore, A. Rajasekar, and M. Wan. The SDSC storage resource broker. In *Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative research*, page 5. IBM Press, 1998.

[102] D. Axmark and M. Widenius. Mysql introduction. *Linux Journal*, 1999(67es):5, 1999.

[103] B. Momjian. *PostgreSQL: introduction and concepts*, volume 192. Addison-Wesley, 2001.

[104] F. Stegmaier, U. Gröbner, M. Döller, H. Kosch, and G. Baese. Evaluation of Current RDF Database Solutions. In *Proceedings of the 10th International Workshop on Semantic Multimedia Database Technologies (SeMuDaTe), 4th International Conference on Semantics And Digital Media Technologies (SAMT)*, pages 39–55, 2009.

Résumé :

Nowadays, a major part of the information is in digital form. Digital preservation is essential to allow people to access information over time. From a computer science perspective, two major objectives have to be met to enable digital preservation: developing archival systems to manage the preserved digital information, and select information representations that will facilitate the preservation. For complex information such as product models, these two objective are particularly hard to meet. Archival systems have to operate in a complex environment, interact with many different systems, and support may different business functions. Product model representations do not use all the possibilities of computer interpretation.

Regarding the development of archival systems, the key is to determine what has to be described to prove that the archival system can effectively support the digital preservation. The Reference Model for an Open Archival Information System (OAIS) proposes a terminology to describe and compare archives. The Audit and Certification of Trustworthy Digital Repository (ACTDR) provides criteria for the certification of archives. One issue with these efforts is that there is not guidance on how to use them within archival system descriptions.

This thesis proposes a method called Reference Architecture for Archival Systems (RAAS) to describe archival systems implementations. RAAS relies on the DoD Architecture Framework to describe the various aspects of the archival systems. Moreover, RAAS provides an archival-specific terminology inspired by the OAIS Reference Model. RAAS also explains how the archival system description can help for the ACTDR certification.

RAAS is applied to a product model preservation case, to describe the various aspects of the archival system. This description includes the interactions involving the archival systems, the archival system functions, the definition of the preserved content, and the definition of the metadata. This description formally refers to the OAIS terminology, and provides ACTDR certification evidence.

This thesis also address the representation of product models by proposing the translation of product models from STEP to OWL. STEP is a standard for product model representation. The use of OWL enables semantic relationship to enrich product information, and improve the search and the understanding of this information using data integration.

The methodology used in this thesis can apply to other types of information, such as medical records.

Mots-clés : digital preservation, enterprise architecture, archival systems, product models, semantic web

SPIM

■ École doctorale SPIM - Université de Bourgogne/UFR ST BP 47870 F - 21078 Dijon cedex

■ tél. +33 (0)3 80 39 59 10 ■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

UNIVERSITÉ DE BOURGOGNE