LONG-TERM PRESERVATION OF 
PRODUCT LIFECYCLE METADATA 
IN OAIS ARCHIVES

Dissertation 
zur Erlangung des akademischen Grades 
Doktor rer. nat.

der Fakultät für 
Mathematik und Informatik 
der FernUniversität 
in Hagen

von 
Diplom-Informatiker, 
Diplom-Kaufmann 

Jörg Brunsmann 
Münster

Hagen, 2012
Abstract

The product lifecycle spans from idea generation, design, manufacturing, and service to disposal. During all these phases, engineers use their knowledge to fulfill their tasks. If engineers retire or leave a company, their embodied knowledge also resigns. To circumvent such loss of important company’s intellectual property, and to enable traceability and reuse of product data in the future, the engineer’s knowledge is captured as metadata and then used as annotation for product lifecycle data. Since such annotated product data models represent meaning in conformance with ontologies, product data can easier be explored, understood and reused by future product lifecycle actors. For business, contractual, and legal reasons, these semantically enriched models are ingested into OAIS (Open Archival Information System) based archives for later reuse.

Notably, it is not uncommon for a product service provider to operate products for several decades; even after the engineers whose embodied knowledge supports their operation retire or leave the company. This product longevity and volatile knowledge, alongside rapid technological innovations and evolving ontologies result in difficulties of reusing archived product data in the long-term. Especially, domain ontologies must reflect changes in real world phenomena which they describe. Therefore, special preservation processes are required to prevent semantic obsolescence of archived product data due to changes in domain conceptualization and to keep the archived product data and metadata interpretable.

While preservation of the data is concerned with product data normalization, validation and file format migration, the preservation processes for metadata are of a different nature given that referenced ontologies evolve independently from the products they describe. Although widely referenced, the OAIS reference model unfortunately does not observe ontology evolution or metadata preservation in any depth. This thesis therefore aims to introduce dedicated metadata harmonization functionality into OAIS archives, based on operational ontology update processing which together build a semantic digital archive system architecture that enables to preserve captured metadata under knowledge evolution.
Contents

1 Introduction .................................................. 7
   1.1 Scope of the Work ........................................ 7
   1.2 Motivation and Problem Description ..................... 9
   1.3 Research Questions ..................................... 9
   1.4 Thesis Contents Overview ............................... 11

2 Problem Statement and Approach of the Thesis ............. 15
   2.1 The Product Lifecycle ................................... 15
      2.1.1 The Design Lifecycle ................................ 16
      2.1.2 The Factory Lifecycle ............................... 17
      2.1.3 The Service Lifecycle ............................... 18
   2.2 Product Lifecycle Data and Metadata .................... 19
      2.2.1 Product Lifecycle Metadata ......................... 21
      2.2.2 Ontologies, Semantics, and Annotated Product Data 22
      2.2.3 Long-term Archival of Product Lifecycle Data .... 24
   2.3 The Metadata Preservation Lifecycle .................. 25
      2.3.1 Capturing ............................................. 25
      2.3.2 Annotation .......................................... 26
      2.3.3 Archival ............................................. 27
      2.3.4 Schema and Instance Evolution ..................... 27
      2.3.5 Exploration .......................................... 27
      2.3.6 Reuse ................................................. 28
   2.4 Problem Statement ....................................... 30
      2.4.1 Obsolescence of Archived Metadata Under Ontology Evolution 31
   2.5 Goal and Approach of the Thesis ....................... 34
      2.5.1 Capturing Operational Metadata Update Packages 35
      2.5.2 Active Harmonization of Archived Metadata .......... 36
   2.6 Summary .................................................. 37

3 State-of-the-Art and Related Work .......................... 39
   3.1 State-of-the-Art ......................................... 39
      3.1.1 Ontologies in PLM .................................... 40
         3.1.1.1 Ontologies as Knowledge Representation .......... 40
         3.1.1.2 RDF and RDFS ................................. 42
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1.6.1</td>
<td>Access for Display and Tracing</td>
<td>133</td>
</tr>
<tr>
<td>6.1.6.2</td>
<td>Query Mediation for Discovery</td>
<td>133</td>
</tr>
<tr>
<td>6.1.7</td>
<td>Reuse</td>
<td>134</td>
</tr>
<tr>
<td>6.1.7.1</td>
<td>Transformation for Embedding</td>
<td>134</td>
</tr>
<tr>
<td>6.2</td>
<td>Evaluation Results</td>
<td>134</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion and Outlook</td>
<td>137</td>
</tr>
<tr>
<td>7.1</td>
<td>Summary</td>
<td>137</td>
</tr>
<tr>
<td>7.2</td>
<td>Contributions</td>
<td>138</td>
</tr>
<tr>
<td>7.3</td>
<td>Future Work</td>
<td>142</td>
</tr>
</tbody>
</table>
1 Introduction

This chapter introduces the scope of the thesis, describes the problem which the thesis addresses, states the relevant requirements as well as research questions and finally provides an overview of the thesis structure.

1.1 Scope of the Work

Long-term digital preservation are continuous processes and technologies which guarantee that archived digital data remains accessible and usable for long time periods which might span over several decades or even centuries. Although this sounds easy at a first glance, one needs to keep in mind that digital data preservation is more than archiving bits and bytes. Today, accelerated innovations in information technology result in frequent changes of technologies including hardware, media, operation systems, application software, and file formats. This high technology obsolescence rate presents a major threat for archived digital data. One application domain which certainly needs to implement long-term digital preservation processes is the design and engineering domain.

The design and engineering domain includes a large variety of industry sectors like, e.g., electronics, energy, chemical, petroleum, and shipbuilding companies as well as aeroplane and automobile manufacturers and architects. Companies in these industry sectors design, manufacture, and operate products in a competitive and fast-paced fashion. The product lifecycle includes phases from idea generation, process planning, development, production, operation, service to disposal. This lifecycle is accompanied by complex, knowledge intensive, geographically distributed, cross-domain, and cross-company collaboration processes. Product lifecycle management (PLM) systems aim to integrate data, actors, systems, and tools as well as software of different vendors.

During the product lifecycle human and machine actors create a large amount of heterogeneous digital product data. While managing the quantity and variety of product data is one issue, understanding the data during exploration and reuse is another issue. For example, investigating an accident in which the product was involved in, requires to understand decisions which were made during product design or product
operation. Therefore, capturing the knowledge of engineers as company’s intellectual property is very important for future reuse. This digital knowledge preservation becomes even more evident when looking at the fact that many manufactured physical products (e.g., airplanes) have a long lifetime in which it is likely that engineers and their embodied knowledge retire or leave the company. To support the understandability of product data by future lifecycle actors, it must be semantically enriched. Such product data semantics is expressed as annotations which reference automatically or manually captured metadata that articulate machine processable knowledge of lifecycle actors.

When a product line reaches a specific milestone (e.g., end of production), the semantically enriched product data is archived and removed from active repositories, although manufactured products might still be in operation for several coming decades. Legal and economic requirements demand to guarantee the availability, understandability, and reuse of product data and metadata for as long as the products are in operation and even longer. Since nowadays product lifecycle data is maintained digitally by PLM systems, paper-based archival is inappropriate. Therefore, special long-term digital archival systems seek to ensure the continuous accessibility of archived data via transformation, migration, or emulation. Implementations of these systems are often based on the OAIS (Open Archival Information System) [23] reference architecture, which provides a common vocabulary and data model, as well as an outline of essential archival functionalities and responsibilities, facilitating more consistent thinking among practitioners. Despite the presence of these archival systems, the preservation of digital data in the long-term still remains a complex task, especially in the design and engineering domain.

While the loss of private data has just emotional effects and the loss of data archived in libraries is a cultural loss, the loss of data in the engineering domain could end in major economic and reputation damage. Unfortunately, the heterogeneous nature of the generated product data, the complex product lifecycle and workflow as well as the large amount of actors and data impose major challenges for implementing long-term archival into company processes of the engineering domain. To make challenges even harder, ensuring the archival of product data is only the first step towards true preservation of product data. It also has to be ensured that semantically enriched product data which have been archived can be discovered and understood in the long-term while technologies and terminologies evolve.
1.2 Motivation and Problem Description

Even if product data is annotated with metadata which adds semantics to the product data, reusing archived product data and metadata will become difficult in the long-term. While, e.g., archived (CAD-based) geometry data is threatened by the evolution of vendor specific file formats, archived product semantics depends to a large degree on the evolution in domain conceptualization and technology innovations. This evolution causes that archived product semantics expressed as metadata will lose interoperability with the future. Since metadata enables the discovery and understandability of data, metadata obsolescence leads to the loss of archived product data. Therefore, dedicated archival functionality must actively and permanently harmonize captured metadata. Studying current practices of metadata preservation reveals the following deficits:

- The few academic and industrial projects which tackle the archival of product lifecycle data do not study knowledge preservation under terminology evolution.
- Only a few publications and implementations are concerned with the capturing, archival, evolution, and harmonization of knowledge expressed with contemporary ontology based semantic technologies.
- Although widely referenced, the OAIS reference architecture does not take into account dedicated metadata archival and harmonization functionality.
- Impacts on archived metadata, queries, and ontology alignments during ontology evolution are not studied in deep detail.

This thesis fills these gaps by introducing concepts which record the provenance of domain ontology elements and exploit these in dedicated metadata harmonization functionality. Such archival functionality keeps product lifecycle data interpretable and enables to integrate long-term archival systems into the every day PLM workflow.

1.3 Research Questions

After presenting the problem which the thesis addresses, the following section states relevant requirements and research questions which are studied in this thesis.

**Semantic Annotation of Product Data** Independently created data of a particular domain of interest which is annotated to product data becomes metadata and provides meaning for data. Since PLM processes are knowledge intensive and include a variety of actors, a predefined amount of descriptive metadata is not sufficient
to preserve an unforeseeable range of knowledge generated in all product lifecycle phases. In order to fully understand and reuse product data in the future, a flexible number of domain metadata must be preserved. A corresponding research question capturing this need can be formulated as:

**How can product data be annotated with a variety of domain metadata to enable its discovery, traceability, and understanding in the future?**

**Product Lifecycle Knowledge Representation** To give product lifecycle metadata a community shared meaning (semantics) which supports the interoperability of human and machine product lifecycle actors at present and in the future, a suitable representation of product lifecycle knowledge must exist. A corresponding research question capturing this need can be formulated as:

**How can product lifecycle knowledge be represented for human and machine processing?**

**Archival of Annotated Product Data Models** Although different actors produce large amounts of data in various file formats during different product lifecycle phases, the coherence of aggregated product lifecycle data must be guaranteed. To ensure the discovery, exploration and reuse of archived digital product data, it must be archived together with the annotated metadata. A corresponding research question capturing this need can be formulated as:

**How can product lifecycle data and metadata be aggregated and ingested into long-term archives?**

**Integration of Long-term Archives into PLM Processes** Even if product data is removed from active data repositories, archived product data and metadata can deliver additional value during reuse. To enable such data and knowledge reuse, the long-term archive must remain constantly accessible and must be integrated into the daily PLM workflow. A corresponding research question capturing this need can be formulated as:

**How can a long-term archive be integrated as permanently accessible repository and knowledge base into the product lifecycle tool chain?**

**Archived Metadata is Threatened by Semantic Obsolescence** Product data contain references to metadata. Metadata instances conform to special domain ontologies which provide explicit and shared meaning for metadata. These ontologies have
to reflect changes in real world phenomena which they describe. If an ontology evolves in a non backward compatible fashion, archived metadata might become non-interpretable and lose interoperability with the future. Product data which has been annotated with this metadata is threatened to be excluded from reuse since it may not be discoverable anymore. Therefore, potential threats for archived metadata under ontology evolution have to be investigated. A corresponding research question capturing this need can be formulated as:

**How is the traceability of annotated product lifecycle data be threatened by ontology evolution?**

**Dedicated Metadata Harmonization Ensures Knowledge Reuse** Ontology evolution may cause problems for dependent applications, archived metadata and referencing ontologies. To overcome the threat of loosing semantic interoperability in the long-term, dedicated metadata harmonization functionality is required to enable product data reuse. A corresponding research question capturing this need can be formulated as:

**How can product lifecycle metadata be harmonized with OAIS archive functionality?**

**Metadata Harmonization Requires Traced Ontology Engineering** To maintain the ability of reuse of archived metadata under ontology evolution, ontology engineers must be forced and supported to record the semantics of ontology updates and to specify transformation rules which describe how to cope with ontology updates. A corresponding research question capturing this need can be formulated as:

**How can ontology engineering functionality support the long-term harmonization of archived metadata?**

### 1.4 Thesis Contents Overview

The rest of the thesis is structured as follows:

**Chapter 2** presents the problem statement and the goal of the thesis in more detail. Therefore, the chapter starts with an introduction to the product lifecycle containing three sub lifecycles in which various lifecycle actors generate large amounts of data and metadata across company, lifecycle, system, tool, and device borders. The second part of this chapter describes product data models and how annotations and metadata help
to add semantics to product data. The next section continues with a discussion of the preservation lifecycle of metadata and provides an overview of metadata reuse scenarios along the whole product lifecycle. The next section of the second chapter formulates the semantic obsolescence problem which occurs if product lifecycle metadata is archived under ontology evolution. The last section concludes with the goals and approach of the thesis.

Chapter 3 contains a discussion of related work and provides an overview of the state-of-the-art of relevant technologies. As product metadata conforms to domain ontologies, the chapter starts with a description of ontologies for knowledge representation. Especially product ontologies are studied in more detail as an example for ontologies used in PLM. The following section contains a description of ontology update patterns, their compatibility and their representation. The following section continues with a description of the OAIS reference model as well as an overview of existing functionality of metadata processing in OAIS based archives. The chapter continues with an introduction of previous industrial and academic projects which have addressed the digital preservation of engineering domain data. The studied projects are matched against important aspects regarding scope and functionality. The section also takes a closer look whether the projects have tackled the goals of the thesis and which challenges regarding long-term product data preservation remain open. The following section then provides an overview of the SHAMAN approach regarding the preservation of digital product data and discusses that this thesis - being located within the SHAMAN project context - addresses the challenge of preventing semantic obsolescence of archived metadata. The last section of the chapter finishes with an analysis and identifies missing concepts which are modeled in chapter 4.

Chapter 4 of the thesis presents a model of missing components for successful product lifecycle metadata preservation. The ingredients for this model are twofold. First, dedicated metadata harmonization functionality (syntactic and semantic normalization, mediation, transformation and migration) in OAIS based archives is modeled. Second, based on this harmonization functionality, requirements for ontology engineering are specified which are then used to model ontology engineering tool functionality. Such ontology engineering functionality includes recording the provenance of ontology schema and instance elements which supports the harmonization of metadata in long-term archives.

Chapter 5 provides an overview of the implementation of metadata preservation functionality along the product lifecycle. Therefore, the first section starts with the presentation of a special system architecture which integrates product lifecycle systems, ontology engineering tools, and long-term archives. The chapter then continues with the detailed description of relevant tool functionality for each phase of the metadata preservation lifecycle.
Chapter 6 provides an evaluation of implemented solutions for metadata preservation. Therefore, example application usage scenarios are presented in order to demonstrate how the tool functionality is able to cope with knowledge evolution. In the last section of the chapter, the implementation of metadata preservation and the usage scenarios are subjected to evaluation based on critical opinions of industry practitioners.

Chapter 7 concludes with a summary of the thesis. The chapter provides a review of the research questions and requirements which were posed in the first chapter and also outlines prospects for future research work and investigations.
1 Introduction
2 Problem Statement and Approach of the Thesis

This chapter provides an overview of the product lifecycle [131] and introduces three sub lifecycles. After presenting the features of each sub lifecycle, the generated product data and metadata is characterized. The chapter continues by describing the preservation lifecycle of metadata and how this metadata is used for annotating product data. Based on the analysis of the product lifecycle and the metadata preservation lifecycle, the problem statement is derived and the approach of the thesis is presented.

2.1 The Product Lifecycle

The product lifecycle spans from idea generation, development, production, operation, service to disposal [123]. In order to study data which is generated during the product lifecycle, [19] refined the whole product lifecycle into the product design, product manufacturing, and product service (operational) sub lifecycle. Figure 2.1 depicts these sub lifecycles as circles and the data flows (which are described in section 2.2) as arrows with numbers. Since the product and factory design is executed in parallel they are clustered into the manufacturing process block. Product lifecycle data is distributed across different borders:

- Different sub lifecycles exist in the whole product lifecycle.
- Various human and machine actors generate product data in each sub lifecycle.
- Product lifecycle data is exchanged between different systems.
- Human actors use different devices for generating product lifecycle data.
- Tools generate heterogeneous formats according to proprietary and open schemas.
- As products are in operation for several decades, product lifecycle data conforms to schemas that grow apart and evolve over time.
- Cooperating organizations execute different processes.
- (Virtual) organizations are geographically distributed around the globe.
During the execution of the sub lifecycle phases, Product Lifecycle Management (PLM) systems support the integration with Enterprise Resource Planning (ERP) and requirements engineering tools [62]. The sub lifecycles are now described in more detail.

![Figure 2.1: The Product Lifecycle and its Sub Lifecycles](image)

2.1.1 The Design Lifecycle

The design lifecycle starts with innovation processes that capture the initial product idea and requirements. The actual product design is sometimes executed through domain and enterprise collaboration, co-operations between virtual organizations and different engineering domains. Data which is created during the product design phase is stored
in special repositories according to customizable product data models. These so-called Product Data Management (PDM) systems [39] support data exchange between the design tools and store the relevant data in a central or distributed repository. A PDM system is an important component of a whole PLM system and is used for data exchange among all product lifecycle departments and actors.

During the design lifecycle, different “documents” are created which are related to the different design phases. Depending on the industry branch, specific phases are part of a design lifecycle. For example, in Printed Circuit Board (PCB) design, the following steps are executed: component library creation, schematic capture of component list and their connections (netlist), component placement and routing as well as circuit simulation [50]. In other design areas (mechanical design, system design, etc.), other specific data is generated according to the necessary process steps, like geometry, functional and simulation models, etc. Finally, during product design a variety of knowledge is created [17] (design rationale, constraints, history, review) including collaboration data of actors involved (votings, discussions, decisions, etc.). Finally, often references to standard product classifications are created in order to describe the properties of product components.

2.1.2 The Factory Lifecycle

The generation of product data lifecycle is not only restricted to the design sub lifecycle. Relevant data is also created during the factory sub lifecycle. A digital factory [72] is a virtual representation of a real factory based on a digital design model of all resources and processes (e.g., automotive or aircraft industry) of the factory. It can be used for modifying a manufacturing process, either due to new requirements or for efficiency improvements. The factory model covers supply chains, processes, workplaces, order plans, and employees [95] which are required for product production. By using these digital simulations, the manufacturing process can be optimized without investing in production materials and without affecting the current production. To describe the whole production process, the manufacturing model also has to contain the description of a number of non-technical processes (e.g., logistic processes modeling the material flow).

The factory lifecycle (see figure 2.2) has a similar outlook as the overall product lifecycle since it starts with the product design and its digital simulation and testing before it is physically manufactured and then used by customers. In parallel to the product design also product manufacturing has to be planned and designed. This may result in the building of a complete factory (e.g., for a car series) or in the adaptation of existing devices, e.g., by creating robot programs or by producing specific dies for a die forging production line. The “product” of the factory lifecycle is a factory which can produce the
actual product. The actual product is then manufactured by entering the operational phase of the factory.

![Diagram of the Parallel Factory and Design Lifecycle](image)

Figure 2.2: The Parallel Factory and Design Lifecycle. Adapted from [19]

Whereas the data which is generated in the design and manufacturing phase is mostly related to the concept of the product, during the operation of the digital factory, various product instance-specific data has to be produced as a result of creation of the instance, e.g., time and production line, where it has been produced, and many other data which allow to trace the product to its exact manufacturing environment. This information might become important during the use and disposal of the product.

### 2.1.3 The Service Lifecycle

In particular for mass products, the service lifecycle is quite separated from the other lifecycles, and it is still difficult or not yet possible to capture data from the operational phase due to non-existing technologies and processes. However, some companies do no longer focus on selling products alone. Instead, services around the product are offered as Product Service Systems (PSS) [33]. The key idea behind PSSs is that consumers do not specifically demand products (as tangible assets), but rather seek the utility of these products.

A physical automobile or a power plant customized by intangible services (e.g., maintenance, training, operation and disposal) is an example of such product and service combination. By using a service rather than a physical product, more customer needs
are met with lower material and energy requirements, since PSSs enable manufacturers and customers to decrease their service costs through cooperation. Another PSS example is a mobility service including recycling. The service transportation (measured in usage hours) may be provided by a car manufacturer to a customer rather than the product ownership. This implies the responsibility of the manufacturer for maintenance and recycling which enables both the customer and the producer to decrease their costs if detailed information about the status and usage of the car fleet is available.

Product lifecycle data is either static or dynamic [154]. Most of the data which is generated during the service lifecycle is dynamic and corresponds to different underlying information models [19]:

- As-built-model: exact information about the manufacturing of the single product instance (material, configuration, subcomponents).
- As-maintained-model: reflects all modifications which have been done during maintenance, e.g., replacements of components by other components.
- As-operated model: contains information about the operation, e.g., number of kilometers on paved roads and on dirty roads, current state of brake pads, etc.

An important aspect for data to follow up products over the service lifecycle is the quantity in which a product is produced and also its customization. Some products are unique (i.e., only one product instance exists), e.g., factories or power plants. Other products are produced in big quantities (nails and screws). For a power plant a lot of operational data is captured which is usually not done for a single screw.

Further information is maintained during the operation and service of a product. After its deployment, users need to be trained in order to operate a product. During the training phase, immediate feedback can be collected manually from users or service personnel. Furthermore, operating data may be recorded automatically by sensors in order to improve usability and reliability, but also to test whether the product operates within expected parameters. A flight simulator model is an example for illustrating the complexity of data which might be needed for the training phase.

### 2.2 Product Lifecycle Data and Metadata

Various data flows can be observed between the different sub lifecycles of the product life cycle (see figure 2.3). The design and factory sub lifecycles are depicted in one box, since very often it is not visible for the outside whether data comes from design or manufacturing or whether data flows to design or manufacturing. Design-internal and
manufacturing-internal flows are out of scope here. Hence, four different data flows are of interest:

1. Factories and products are designed simultaneously by using distributed processes and data. Collaboration takes place within the same or across different companies. The factory designer receives data (like product part specifications) from product design and creates and optimizes manufacturing processes. If manufacturing simulation (see section 2.1.2) reveals problems, the product designer needs to get feedback and has to adapt the design.

2. After the product has been manufactured, it is delivered to the customer or operator. During this hand over, important data flows between the two sub lifecycles. Service and training personnel receive product documentation (e.g., initial training material) and specific product instance meta-data (e.g., a serial number) is transferred to service actors.

3. Service providers need information about the product usage to improve their services, e.g., by planning maintenance in time, ensuring availability of spare parts, replacing products before they stop working, providing up to date training and maintenance material, etc. Their goal is to improve the product availability while decreasing the costs.

4. Data that is created during service and operation can be fed back into the product design process to support product redesign or variation. In addition, product service lifecycle data might have ecological and economic influence on the product design and hence on the evolution of the manufacturing and digital factory processes themselves.

Data which is created during these different product sub lifecycles is stored in special repositories according to customizable product data models.
2.2 Product Lifecycle Data and Metadata

2.2.1 Product Lifecycle Metadata

The engineering domain maintaining the product lifecycle is characterized by knowledge intensive processes which produce not only geometry data. Indeed, during all product lifecycle phases, different agents (e.g., human actors, computers, and sensors) [71] collaborate across countries, companies, time, and technologies to produce the following valuable metadata:

Factory Simulation Metadata
To shorten the time from product idea to manufacturing, sometimes digital factories are used to simulate the manufacturing of products. If such manufacturing simulation reveals problems, the product designer needs to get feedback to adapt the design. The provenance of factory simulation which reflect the evolution of simulation runs can be described by metadata.

Ideation Metadata
Product innovations are created by executing an innovation process. Most of these innovations are not manufactured at all, but remain important intellectual property of a company. Metadata is used to describe the ideas for future reuse.

Product Part Specifications
References to standard product classifications which are attached to product parts also represent reusable knowledge.

Product Design Review and History
During a collaborative design review, arguments and justifications are exchanged by experienced expert designers which have to be made explicit by annotating the design. In addition, if a design is changed because of a failure report or new requirements, the history of the design is important to capture.

Product Design Rationale and Decision
Sometimes, electronic (ECAD) and mechanical design (MCAD) processes are executed simultaneously as cross-domain collaborations [116]. Incremental design change proposals and constraints are exchanged bi-directional, evaluated, and finally accepted or rejected. The reasons for rejecting or accepting are knowledge that has to be made explicit as relevant design rationale.

Operational Metadata
During product service and operation, knowledge about the behavior of a product is obtained from customers or service personnel. In order to maximize the satisfaction of a customer, this knowledge can be reused for a product redesign or variation.

Project Related Metadata
This kind of metadata includes, for example, a list of project participants and their social network.
All this metadata add meaning to specific product data and enable the data exploitation during later reuse in other product lifecycle phases. Several use case scenarios of such knowledge reuse are provided below after describing how metadata is annotated to product data.

### 2.2.2 Ontologies, Semantics, and Annotated Product Data

Sometimes it is difficult to differentiate between data and metadata. Metadata is defined as "data about data" [1] which "describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource." [100] Adapting these definitions to product lifecycle processes is easy. For example, an independently created hierarchy of products which allows systems to communicate unambiguously is called a *product classification* and represents data on its own. However, it can be regarded as *metadata* when it is used as annotation of other data. Such annotation requires that metadata has a *unique identifier* [76] which can be referenced by the annotations. Linking product data with metadata by means of annotations (figure 2.4) follows the idea which was described for the annotation of documents created in the semantic web [130].

![Figure 2.4: An Annotated Product Lifecycle Data Model](image-url)
2.2 Product Lifecycle Data and Metadata

The separation of data and metadata which are linked by annotations makes it possible to maintain data and metadata separately. An annotation can be referenced more than once and can be edited separately from data and metadata. Annotations reference newly created (e.g., collaboration metadata) or predefined (e.g., product classification) metadata. Annotations and referenced metadata are created manually or automatically. Often metadata has to be captured explicitly at the moment of its availability otherwise it might be lost and cannot be recreated. For example, if the reason for a design change is not captured instantly, it is most likely lost forever.

An example will illustrate the annotation of product data. Assuming that an engineer wants to indicate that parts of his product design consist of a specific resistor. The engineer will select all relevant product parts and will launch special tool functionality which allows to browse a product classification in order to select a specific resistor which has a unique identification. The generated annotation will link all selected product parts (which themselves have a unique identification) with the identification for the resistor. The properties (e.g., name, technical specifications) of the specific resistor can later be used for discovery of product design parts.

Product data, annotations as well as metadata conform to specific schemas.

- Product lifecycle data is described by proprietary models and different representations of product lifecycle data (e.g., geometry, netlist, routing) are independently modeled.

- The annotation schema (gray box labeled Schema A) which contain date and author as well as references to data and metadata.

- The metadata which is referenced by annotations conform to individual metadata schemas.

In particular, the metadata schemas are relevant for the thesis and are now described in more detail. To express a common understanding, metadata conforms to a schema which models a particular real or imagined application domain of interest (e.g., forests, products, animals, furniture). In figure 2.4, the individual domain schemas are depicted above the metadata (e.g., Ideation, Design, and Collaboration Schema). A metadata schema may be of any kind. For example a relational database model, a XML schema [45], a Document Type Declaration (DTD) [10] or a Resource Description Framework Schema (RDFS) [12] based ontology. Ontologies enable to model controlled vocabularies, taxonomies and thesauri [135] [139] through the specification of classes, properties, instances, and their relationships [104]. Ontologies are "a way for a community to agree on common terms for capturing meaning or representing knowledge in some domain." [138] Often, an ontology based schema has been built in a consensual process by several people,
companies or organizations and has therefore gained the status of a *public standard*. Annotations may reference such publicly recognized standard meaning which is expressed by ontologies. By doing so, these metadata annotations ensure the understanding of product data and add additional meaning (*semantics*) to the product data. The usage of metadata schemas for annotating archived digital content ensures consistent interoperability. If archives and accessing software agree on using the same metadata schema, they are able to interoperate. However, if archives use different schemas or different versions of the same schema, interoperability is in danger. Therefore, [2] gives an overview of four different approaches to enable metadata interoperability.

This thesis observes three different ways how semantic annotations relate product data and product metadata:

**Data to Metadata Annotation** Product data parts are annotated with newly created or predefined metadata (e.g., product classification).

**Data to Metadata Schema Annotation** Product data parts are annotated with metadata schema elements. For example, it might be useful to annotate a set of geometrical instances with the information that these objects represent a punching hole in the complete geometry of a complex mechatronic printed circuit board.

**Data Schema to Metadata Schema Annotation** Internal product data schema elements are directly related to external metadata schema elements which provides a mapping between the different schemas. For such a mapping, different kinds of annotations may be used for expressing the relationship between the schema elements in the different models (e.g., equivalence, subsumption, etc.). Whereas data to metadata annotations specify meaning of a specific instance, in this case the semantics applies to all instances of data schema elements.

### 2.2.3 Long-term Archival of Product Lifecycle Data

From the previous sections one can derive the significant characteristic of the huge size and heterogeneity of the product data. Requirement documents, simulation results etc. from different lifecycle phases are part of the product data as well as metadata which provides meaning to product data. Despite these characteristics, engineering domain companies are forced to archive product data [62] in order to

- conform to several legal regulations (e.g., product liability).
- comply to negotiated contracts between cooperating companies regarding keeping data accessible.
- reuse knowledge which is generated by employees or customers.
To meet these legal, contractual, and economic requirements, the product data has to be archived and preserved for very long periods. Unfortunately, product data models are usually proprietary, and for competitive reasons, it is common that single vendors will invent non backward compatible file formats. Therefore, it is desirable that 3D CAD file formats are transformed to open standard representations (e.g., STEP [64], PLM Services 2.1 [106], or Product Life Cycle Support (PLCS) [67]) before they are stored in an archive. Since standard data models are described by open specifications and are less often modified than vendor specific proprietary models, they are more suitable as format for long-term archiving. Some parts of the product data model may need to be stored in native form because no globally agreed standard model exists. In any case, it is reasonable to transform proprietary product data models into a standard product data model before archiving in order to increase the probability of successful interpretation during future archive access.

2.3 The Metadata Preservation Lifecycle

According to the previous sections, a variety of metadata is required for expressing the knowledge which is created during the product lifecycle. To provide meaning, the metadata conforms to schemas that make common domain knowledge explicit and usable for machines and humans [109]. Often, these schemas are expressed as ontology-based schemas and enable interoperability of systems, actors, and tools. In this section, the thesis defines an idealized metadata preservation lifecycle in the context of digital product data archival. As an refinement of the information lifecycle [13] the metadata preservation lifecycle includes the phases of capturing, annotation, archival, evolution, exploration, and reuse.

2.3.1 Capturing

The creation of metadata is done either automatically or manually. Automatic metadata extraction has to be executed at the time of data production-time because metadata cannot be recreated later on (e.g., simulation run with specific model parameters or metadata for project meetings). For metadata preservation processes it is required that only archival relevant metadata is captured. In collaboration between mechanical and electrical engineers, the exchanged collaboration data also contain 2D and 3D geometry data which describe proposals regarding component placement which aims to avoid collisions between printed circuit boards and mechanical housing [116]. For metadata preservation processes not every of these incremental transactions needs to be captured.
However, this thesis states the requirement to capture the final decision made by cooperating engineers. The following additional requirements for capturing product lifecycle knowledge can be derived:

**Real-time Capturing** Product lifecycle knowledge has to be captured and archived at that point in time when it is initially created. Otherwise the knowledge is lost forever since it cannot be recreated.

**Product Lifecycle Based Capturing** Relevant product lifecycle knowledge needs to be captured in all phases of the product lifecycle.

**Non-intrusive Capturing** Since knowledge capturing is costly, the process should not interfere with normal business activities and knowledge extraction should be automated to minimize intrusion.

**Formal Capturing** If a product design has to be accessed and therefore needs to be searchable and findable in the future, formalized and automated knowledge capturing, representation, and management is necessary because good search results can only be achieved if product data is archived accompanied by their knowledge in a formal representation.

### 2.3.2 Annotation

As described in section 2.2.2, data can be regarded as metadata when it is used as annotation of other data. Metadata which is referenced by the annotations conform to domain ontologies in order to provide meaning. In this thesis, annotation is regarded as the process of creating new or using existing metadata which conforms to an ontology and relating this metadata with a (set of) product data parts which are supposed to be annotated. Some examples how data of different product lifecycle phases are annotated with metadata were also provided above. The product data is held in special repositories which might reside on top of a relational database [5]. The metadata can be stored and maintained external to the PDM repository in a dedicated metadata storage [145]. Since the product data and metadata are distributed in different repositories, unique data and metadata identifier have to be referenced by the annotations to link the product data and metadata. The data and metadata identifier also have to be persistent even if new versions of the ontology are introduced [76] or new product data model versions are invented.
2.3 The Metadata Preservation Lifecycle

2.3.3 Archival

The process of submitting data to an archive is called ingestion [23]. The time of ingest of the annotated product data into a long-term archive depends on the product lifecycle. However, the archival of product data does not begin when the time of ingest has come. Rather, metadata which is captured along the entire product life cycle is particularly relevant for successful archival, since it enables the searchability, traceability, exploration, and reuse of long-term archived product lifecycle data. Thus, preservation of product data starts when metadata is captured. When a product data model is ingested into a long-term archive, special care needs to be taken with respect to annotations and referenced metadata.

2.3.4 Schema and Instance Evolution

As described in section 2.2.2, a schema models a domain of interest and metadata is associated with a schema to which it conforms to. Often, this metadata schema is expressed as ontology. Especially in the engineering realm, the modelled domains are continually changing due to technology innovations and knowledge explosion [119]. An ontology has to reflect these changes. Ontology evolution [53] is the process of modifying an ontology in response to a certain change in the domain. Ontology evolution has different reasons, semantics, and consequences [75]. New versions of existing ontologies are generated or new domain ontologies are being invented. Sometimes, an ontology is extended (e.g., by adding classes or properties) which is not critical for the existing metadata because the metadata keeps valid and interpretable. However, sometimes ontology elements are refined. For example, properties of a class receive new meaning. These changes are crucial for archived metadata since it does not conform to the current ontology anymore.

2.3.5 Exploration

When archived product data is accessed, it is possible that the archive consumer does not have an idea what has been ingested. The archive consumer only knows the goal of his archive exploration ambition [16]. By using domain ontologies and its metadata for annotation, product data can be easier understood by future archive consumers. For exploration, archived ontologies might be used. However, if the archive is integrated in daily business workflows, contemporary ontology versions have to be used. In this case, archived metadata might not conform to the current ontology version and special archive functionality is needed to support exploration and subsequent reuse.
2.3.6 Reuse

Reuse has to be understood in a broader sense. A physical product can be reused in a different geographical places. This physical product reuse has to be accompanied by the reuse of product data. Another kind of data reuse occurs when a product design data is reused for variation in order to save development costs and to deliver new products faster. In both cases, product data reuse means to embed the complete archived product data into the current system environment. In addition to the product data, also archived semantics which has been annotated to product data has to be reused for full understanding. Metadata is reused anticlockwise in the same or in a previous product lifecycle phase (figure 2.5). Metadata reuse is the last phase and final goal of the metadata preservation lifecycle. Some use cases with legal, contractual and business motivations illustrate the additional value which semantically enriched and archived product data can provide.

![Figure 2.5: Product Lifecycle Metadata Reuse](image)

**Ideation Metadata Reuse**

During the early innovation management phase, innovation lab engineers discuss and rate their ideas [59] which are tagged with metadata (e.g., business category). Although nearly all of the ideas are not realized, they remain as valuable intellectual property of a company which needs to be preserved for possible future reuse. The semantic tags can be used to find an archived idea in order to avoid reinventing the wheel or to identify the reason why an idea was rejected.
Collaborative Design Metadata Reuse

Along the product lifecycle agents (human actors and systems) collaborate across countries, enterprises, domains and technologies [73]. These collaborations are executed during all product lifecycle phases and produce valuable metadata which can be reused in other lifecycle phases. For example, the documents of a design review represent important knowledge because during such reviews arguments and justification are exchanged by expert designers [17]. In addition, if a design is changed (e.g., failure report, new requirements), the history of the design is important to document. Also, experienced designers are often asked by colleagues about non documented design features. Such social search knowledge is important to made available for future searches [14]. Therefore, it is necessary to capture, annotate and archive the design metadata in order to enable future design engineers to reuse and fully understand the product designs later (e.g., product variation).

Service Metadata Reuse

As described in [19], during product service, human actors (e.g., product owner, mechanics) gain valuable knowledge about the product behavior. If such knowledge is annotated as metadata, it can be reused for improvements in product design or manufacturing. In addition, when sensors monitor product (e.g., car) status information in real-time, it can be broadcasted to the manufacturer who could semantically enrich and collect big amounts of such annotated data to plan efficiently the maintenance scheduling which lead to process improvements.

Project and Provenance Metadata Reuse

Long-lived airplanes (e.g., Boeing B-52) are in operation for several decades and have high maintenance rates and tend to cause technical problems [78]. If unintended malfunctions occurred, an accident investigation requires the exploration of all relevant product data to find the malfunction reason. Over several decades, it is likely that employees have retired or have left the company. If it is assumed that the accident was caused by a specific product part, the investigation also includes the question, why, when and by whom the part was used and modified. Project related metadata assist in this investigation, since it contains a list of project participants including their social network and their participation in collaboration sessions.
Problem Statement and Approach of the Thesis

Product Part Metadata Reuse

If a long-life product contains a broken part which is not manufactured anymore, a service mechanic has to search for contemporary spare parts [57]. This search for alternative suppliers and manufacturers is based on the archived metadata (e.g., technical requirement specifications) which is represented as standard product classifications which are annotated to product parts as metadata.

Collaborative Cross Domain Metadata Reuse

Electronic (ECAD) and mechanical design (MCAD) processes are executed simultaneously to reduce costs and time during product development in industries like automobile, consumer electronics, aviation and space industry. During such cross-domain collaborations, incremental design change proposals are exchanged bi-directional, evaluated and finally accepted or rejected. An open XML-based EDMD (Electrical Design Mechanical Design) collaboration model was developed as industry standard [116]. This data model permits to exchange incremental electromechanical data synchronously and asynchronously, to analyze the effects of a change originating from one domain to the other domain and to continuously commit the accepted changes in the domain-specific design repository. The reasons for rejecting or accepting incremental design change proposals are archival relevant design rationale and represent important collaborative generated knowledge.

These metadata reuse scenarios illustrate the importance of preserving product lifecycle metadata starting with capturing, annotation and later archival. However, the long-term preservation of metadata for future reuse is cumbersome due to reasons explained in the following section.

2.4 Problem Statement

The previous sections gave an overview of the product lifecycle and the metadata which is created during different lifecycle phases by engineers, tools and sensors. The sections also showed that metadata has its own preservation lifecycle in which it is annotated to product data. It was also emphasized that the reuse of knowledge can deliver additional value and is therefore a major motivation for capturing and archiving product lifecycle data and metadata. Current practices will “archive” the product data on some media and delete it from the active data repositories despite the fact that the physical product is still in operation [59]. However, to support the described reuse scenarios it is required
that the product data is ingested into a long-term archive which can be permanently accessed by actors in all product lifecycle phases.

As annotated product data contain the knowledge for exploitation during reuse, these semantically enriched product data are suitable for long-term archival and future reuse. However, the archival of annotated product data in a constantly accessible archive is only the first step to full product data preservation in the long-term. While product data may be archived and reproduced easily by copying bits, ensuring that the product data is usable in the long-term is a harder problem. The reason for this is that technology is advancing quickly (formats, semantics, software, hardware, etc.) and that the lifespan of a product exceeds these technologies. As a consequence, it cannot guaranteed that the archived product data is reusable in newer and current engineering environments.

Product data is always created and stored according to a certain model which may be proprietary or standardized [155]. These models change over time, either in an evolutionary way or by a replacement with a new model. Archived product data is threatened by these changes. Current research projects which are described in the next chapter tackle the preservation of product data by introducing a normalization step. The normalization transfers product data into a standard product data representation before it is ingested into long-term archives. In addition, archive functionality monitors engineering file formats and helps to migrate the archived data in order to maintain their interpretability with contemporary technologies.

In addition to the product data, this thesis also observes product metadata regarding preservation aspects since product data is connected to metadata by annotations. Maintaining the semantics of archived metadata is a prerequisite of reusing product data since metadata ensures the discovery of data. Hence, in addition to the pure CAD product data (geometry, netlist), full preservation of product lifecycle data requires that product lifecycle knowledge expressed as metadata is also preserved for future access. Metadata preservation is different than data preservation and the next section explains why.

### 2.4.1 Obsolescence of Archived Metadata Under Ontology Evolution

As described in section 2.2.2, product data parts are annotated with a reference to metadata which conforms to a specific domain ontology version. According to the metadata preservation lifecycle, after archival of the annotated product data the ontology to which the metadata conforms to will evolve because domain knowledge is not static. Due to this ontology evolution, the metadata terminology which was used to annotate the product data differs from the terminology which is used to query the archive. Hence, reusing
archived product data will become cumbersome since domain ontologies have to reflect changes in real world phenomena which they describe.

Ontology updates come in a variety of patterns and are not restricted to the schema level [132]. Classes can be merged or split, properties might move between classes and even predefined instances which are part of the domain ontology can evolve. Notably, ontology schema and instance updates include non backward compatible changes which have influences on the archived metadata. For example, while adding a property does not influence the discovery of the archived metadata, deleting a referenced domain ontology instance without alternative will be a problem because it cannot be part of a search query anymore.

A simple example for ontology instance update which is depicted in figure 2.6 will illustrate the problem. In year 1975 a domain ontology which models countries is created. In year 1980 this domain ontology is used to annotate a product part (e.g., resistor) with a reference to the domain ontology instance *German Democratic Republic* which indicates the country of the manufacturer. In 1986 the product data and metadata is archived and removed from the active repository. As West-Germany has been reunified with the German Democratic Republic in year 1990, the ontology instance will be deleted and replaced by the instance *Germany* without further notice. In year 2010, product parts are

Figure 2.6: Semantic Obsolescence of Archived Metadata under Ontology Evolution
being searched according to the country of the manufacturer. Since the retrieval tools use the contemporary version of the ontology, *German Democratic Republic* will not be part of the search query anymore. Thus, the archived metadata and in consequence the data will be omitted in the search results because the deleted ontology instance has not been adjusted to the newly modeled semantics.

Things are even worse, as domain ontologies can reference one or more upper-ontologies and can import other domain-ontologies. All these referenced ontologies are subject to change. In addition, in distributed systems knowledge even evolves concurrently, independently and changes are applied without notice [75]. This knowledge progression creates a semantic gap for the archived metadata. Another threat for archived product semantics exists, since different people have different opinions, modeling targets and perceptions. For example, various standards exists for describing product classifications. Hence, the real world is modelled with different conceptual models. In the long-term is it not unlikely that an ontology which was previously accepted as standard in some domain is replaced with another ontology. All these fact lead to *heterogeneity* conflicts. The work [60] provides an overview of various heterogeneity conflicts. These conflicts are also described in [32] which also offers a review of existing heterogeneity classifications and distinguishes between *language* mismatches and *ontology* mismatches. Whereas ontology mismatches (semantic heterogeneity) which occurs due to ontology evolution and different conceptualizations were described above, this thesis also looks at language mismatches.

Language mismatches (syntactic heterogeneity) will occur in the long-term due to the existence of different ontology representation languages [51]. If a change of the ontology representation language is required, e.g., because a new system with a different representation language is invented, then a translation of archived metadata from one language into another language has to be executed. Such a system change can also happen in the short-term but will be more likely in the long-term. The syntactic heterogeneity might also originate from the same ontology representation since over long time periods it is not unlikely that the syntax of an ontology representation language may change. If the language expressiveness changes or is even reduced, valid translations of archived semantics will be cumbersome.

Summarizing, one can state that knowledge evolution via syntactic and semantic heterogeneity pose a obsolescence threat for archived semantics and other metadata (e.g., queries) which are modeled according to a specific ontology version. In consequence, the product lifecycle data is in danger of being omitted during search and retrieval. Additionally, this semantic obsolescence will lead to the loss of interpretability and traceability of archived product data. Such traceability loss will lower the value of archived data and will contradict the amount of work that was spend during creation.
2.5 Goal and Approach of the Thesis

The last section showed that semantic and syntactic heterogeneity poses a threat for archived metadata since it may become invalid and lose interoperability with contemporary ontology versions. This metadata obsolescence will hinder a long-term archival system to act as a permanently accessible and interpretable knowledge base. Therefore, this thesis aims to describe necessary techniques, methods and functionality to overcome the obsolescence of metadata and to seamlessly integrate long-term preservation systems into product lifecycle processes. Figure 2.7 (taken from [152]) depicts how a long-term archival system can be embedded into PLM processes.

![Figure 2.7: PLM Environment with Long-term Archival Functionality [152]](image)

The lifecycle actors and PLM tools which are shown at the top of the figure access the PLM system during all lifecycle phases (from ideation to recycling). The PLM system still controls the engineering and product development processes over the entire product lifecycle. The existing core PLM system functionality already provide basic functionality like version and configuration management and process management which may produce information relevant for digital preservation. The core system functionality is extended by long-term archival functionality which enables the PLM system to connect...
2.5 Goal and Approach of the Thesis

with service interfaces (e.g., ingest, access) of a preservation system while the core system functionality still accesses the native product data repository. The native repository might reference archived product data in the preservation system but not vice versa. This limitation is essential since the preservation system should be autonomous and product data which has been archived in a preservation system should be accessible by PLM systems originating from different vendors.

In such an infrastructure, the collected and archived product data remains accessible and delivers additional value during reuse. The successful integration of long-term archives into PLM processes requires that all relevant product data and metadata is collected during the various lifecycle phases and then ingested together into the archival system. Assuming that relevant archive access interfaces exist and that bit preservation of data is guaranteed, it is necessary to investigate the preservation of product metadata under rapid innovations and knowledge evolution in more detail. This investigation is necessary, because metadata enables the discovery, traceability and final reuse of product data. Hence, the preservation of metadata is as important as the preservation of product data (e.g., CAD file formats). Therefore, it has to be assured that product metadata which has been ingested into a long-term archive can be understood in contemporary environments. This objective requires special functionality for metadata preservation under ontology evolution to prevent obsolescence of product lifecycle data.

2.5.1 Capturing Operational Metadata Update Packages

Domain conceptualization change processes are reflected in updates made to ontologies executed by the responsible actor called ontology engineer. For some domains (e.g., product classifications) ontology evolution is even a problem in the short-term [61] and will therefore be definitely a problem in the long-term. However, there is an additional issue in the long-term: often, additional knowledge (e.g., semantics of the ontology update or knowledge about archived metadata) is required for successful metadata preservation. In the short-term it is possible to use existing human knowledge in case of doubts because this information may still exist in the heads of product lifecycle actors. But in the long-term it is likely that this human knowledge is lost. Therefore it is required that relevant knowledge surrounding an ontology update is captured right at update-time for later archival.

Dependent applications, ontology users and referencing data have to gain knowledge about three relevant parts of an ontology update: they need to know that an update has occurred, what has changed and how to cope with that update. The main problem in ontology update management is not to detect that an update has happened. The bigger challenge is to find out what has changed and to understand what modifications are needed in order to comply with a new ontology version. In this thesis, these two parts
(what and how) are combined together in an operational metadata update packages which can be queried by accessing an update feed which references these update packages. These concepts will be described in more detail in section 4.3.3. To support the creation of these metadata update packages, it is required that ontology engineering functionality exists which supports to describe the rationale and the execution order of ontology updates. Also, to be processable by other systems, it is required that update packages themselves are described with a specific ontology and published for interested applications. One of the dependent application type which is interested in metadata update packages are long-term preservation systems which archive metadata. Dedicated metadata harmonization functionality which is described in the following section is able to exploit metadata update packages.

### 2.5.2 Active Harmonization of Archived Metadata

According to the previous sections, metadata preservation begins when metadata is annotated for later reuse and continues when annotated product data is ingested into long-term archives. After archival, metadata is threatened by ontology evolution and dedicated metadata harmonization functionality is required. As defined in [15], metadata harmonization across time is regarded as the continuous processes that guarantee the correct long-term understandability of both ingested content and metadata for consumer access both now and in the future. Figure 2.8 displays the interaction of the metadata preservation lifecycle and the metadata harmonization functionality which is part of the long-term archival system.

![Figure 2.8: Metadata Preservation Lifecycle and Metadata Harmonization. Refined from [16](image)](image)
The metadata harmonization functionality executed within an archive which maintains the interpretability of product data retrieves and processes metadata update packages which were created during ontology engineering. By processing the updates packages it is guaranteed that product data can be found and interpreted in the long-term. As harmonized metadata conforms to contemporary ontologies, it is ready for access during the phases of metadata exploration and reuse. Since the OAIS reference architecture is accepted as a common reference architecture for reasoning about long-term preservation strategies, this thesis will specify missing metadata harmonization functionality as extensions of the existing OAIS model.

2.6 Summary

This section summarizes the problem, goal, and approach which the thesis address. First, the following list sums up the goals of the thesis:

- Preserving machine processable and ontology-based knowledge generated throughout the whole product lifecycle.
- Enhancing PLM processes by integrating long-term archives which contain knowledge enriched product lifecycle data into the daily workflow.
- Ensuring the discovery, understandability, and reuse of archived product data over long time periods.

These goals are threatened by the issue that over time, the future interoperability of archived product metadata is handicapped because the knowledge is not captured and ontologies which provide a shared meaning for metadata evolve. As a consequence, annotated product data cannot be retrieved, understood, and reused anymore. To overcome this issue, the thesis proposes the following approach:

- To support future understandability, metadata preservation has to start with ontology-based capturing right at product data creation-time.
- Ontology engineering tools have to generate machine processable metadata update packages.
- Operational metadata update packages have to be exploited by dedicated metadata harmonization functionality executed in OAIS-based long-term archives.

In order to implement the thesis goals using the described approach, the next chapter reviews existing concepts, technologies, and functionalities of relevant research areas.
2 Problem Statement and Approach of the Thesis
3 State-of-the-Art and Related Work

Having presented the problem statement and the goal of the thesis in the previous chapter, this chapter provides an overview of the state-of-the-art and related work in the relevant research areas. The analysis of the state-of-the-art includes the investigation of the evolution of product ontologies, metadata processing functionality in OAIS archives, and scope of previous projects in the long-term preservation of product lifecycle data. The description of related work regarding the preservation of archived metadata follows when the SHAMAN approach [125] of long-term preservation of product data is described and the thesis goal is embedded into the SHAMAN framework. Finally, based on the investigation of the state-of-the-art and related work, the last section identifies open challenges for long-term preservation of product metadata.

3.1 State-of-the-Art

As described in the previous chapter, the thesis goal is to explore how to initiate and maintain the understandability of product data by means of annotation and preservation of metadata. Therefore, three different areas have to be investigated.

1. Since the metadata often conforms to an ontology, the state-of-the-art of ontology use in product lifecycle processes is relevant and is therefore described in the next section 3.1.1. Special attention will be laid on product ontologies and their evolution.

2. Since most of the engineering archival solutions are based on the OAIS reference model (which will become clear later on), section 3.1.2 provides an overview of OAIS archive functionality and existing OAIS metadata management.

3. Since the annotated product data has to be ingested into a long-term archival system, the current state-of-the-art of product data preservation will be provided in section 3.1.3 by describing the scope, features, and outcomes of previous academic and industrial projects in this realm.
3.1.1 Ontologies in PLM

The product lifecycle is supported by collaborative processes in which humans create, apply and exchange knowledge [97]. Examples of such knowledge were given in section 2.3.6 including the usage of ontologies for product classifications, design rationale [94], design history, and design reviews. Since all this PLM knowledge is essential for later understanding and reuse of the product data [3], it has to be captured, annotated to product data parts, and represented in machine readable languages to enable its automatic processing and discovery. Ontologies are a possible knowledge representation mechanism [11] and are described below.

3.1.1.1 Ontologies as Knowledge Representation

Different conceptual models exist to describe knowledge about some real world phenomena of interest, including dictionaries, XML schemas, database schemas and ontologies [137]. In philosophy, the term ontology is concerned with the nature of being and existence [81]. In computer science, ontology is a technical term that describes an artefact which enables the modeling of knowledge about some real or imagined domain [147]. An ontology contains assertions which define a set of concepts, their relationships and axioms of a particular domain as a result of a community agreement and public consensual knowledge. Ontologies provide a shared and common understanding of a domain which can be communicated across people and applications. The usage of ontologies has several advantages [144]:

**Human Understanding** An ontology makes community meaning explicit, it provides meaning and structure to the underlying data and supports knowledge exchange among collaborators.

**Machine Understanding** An ontology enables reasoning for computer programs.

**Precise Communication** By defining a shared and common domain concept, ontologies help both people and machines to communicate precisely.

**Discovery** An ontology is a vocabulary that enables semantic search.

Ontologies are different from schemas. The works of [55] and [103] elaborate, why.

- Ontologies have richer data models (cardinality constraints, inverse properties, transitive properties, disjoint classes) than database schemas which have consequences on the possible changes.
- Ontologies include instances and classes which are not always clearly separated.
• The development and evolution of ontologies is a collaborative and decentralized process.

• Ontologies often reuse and import other ontologies and are therefore dependent on the changes. In other words, ontologies are used as schemas for dependent and decentralized knowledge bases and schemas might evolve without update propagation.

Probably the most cited definition for the term ontology defines an ontology as ‘the specification of a conceptualization’ [52]. Figure 3.1 depicts this definition and is now described in more detail.

A concept of a specific domain can be interpreted in many ways depending on the perspective. For example, the domain forest will be interpreted in different ways by humans who have different intensions. A human who wants to paint a forest will concentrate on the colors of the leaves and trunks whereas a human who aims to sell the trees will be interested in the height and diameter of the trunks. Each of these conceptualizations can then be defined in different specifications. For example, the colors of the leaves could be defined as properties expressed as literal values or alternatively the colors could be defined as predefined classes of the ontology specification. The specification can be represented by one or more machine-readable formal languages (e.g., RDF [12] or UML [107]). Finally, the representation can be expressed in different syntaxes (e.g., RDF has as graphical and textual syntax). The ontology modeling tiers are now described in more detail.

**Conceptualization** A domain conceptualization is an abstract and simplified interpretation of a specific real world domain phenomenon. The conceptualization consists of explicitly defined classes, their relationships that hold between those concepts and their constraints. Different users have different perspectives of the same domain and different ontologies may model the same concepts in different ways.
Specification  The specification level contains the precise specification represented in a language using the following structural components:

- **Classes** represent important concepts of the domain which are organized in a superclass-subclass (is-a) hierarchy.
- **Properties** define binary links between classes or between classes and literal values.
- **Instances** represent individuals of the real-world domain as incarnations of classes. For example, if *Country* is a class, then *Germany* is an instance of this class.
- **Relations** are instantiations of properties and connect two instances or one instance with a literal value.
- **Axioms** express assertions which are always true and which are used to infer new knowledge.

Representation  The specification is expressed on the representation level. Various machine interpretable languages exists like KIF, OIL, SHOE, RDF, DAML, and Topic Maps. All these languages are described in [26].

An an example for representing ontologies, the next section describes the Resource De-

3.1.1.2 RDF and RDFS

The Resource Description Framework (RDF) [91] has a simple data model that can be used as a graph-based knowledge representation language. The data model is simply specified as triples of the form (*subject*, *predicate*, *object*). The triple assigns the *predicate* as property with value *object* to the *subject*. For example, to assert that a specific telephone has a pink color, one will specify the triple (*myTelephone*, *hasColor*, *Pink*). The nodes in a RDF graph are Universal Resource Identifiers (URIs) [69] and an object may be a literal. An URI references and identifies a resource unambiguously. Arcs between the nodes, labeled with URIs, represent relationships between resources. More formally:

- \( U \) is the set of Uniform Resource Identifiers (URIs).
- \( L \) is the set of RDF literals.
- A statement is an RDF triple in \( U \times U \times U \cup L \).
- In a triple \((s, p, o)\), \(s\) is called the subject, \(p\) the predicate and \(o\) the object of \(s\).
A RDF graph $G$ is a set of triples.

A named graph $NG$ is a pair $(u; G)$ of an URI $u$.

RDFS [12] is a special vocabulary that permits the definition of classes, properties, instances and various predefined relations between resources which include the definition of property domain (rdfs:domain), range (rdfs:range), subsumption relationships between classes (rdfs:subClassOf) and between properties (rdfs:subPropertyOf) and instantiation relationships (rdf:type) between instances and classes or between pairs of instances and properties. The RDFS vocabulary is used on top of the RDF data model for the definition of explicit triples like $(Telephone, rdfs:subClassOf, Product)$ which states that each telephone instance is a product. A special vocabulary called prdf [99] is a minimalistic vocabulary to define a class hierarchy which includes the terms sp, sc, type, dom, range.

The specified semantics of the RDFS language can be used to infer new knowledge. For example, the RDFS specification states that the rdfs:subClassOf predicate is transitive. Such semantics can be used to infer new implicit triples. For example, the triples $(Dog, rdfs:subClassOf, Mammal)$ and $(Mammal, rdfs:subClassOf, Animal)$ imply the fact $(Dog, rdfs:subClassOf, Animal)$. Another inference example is through type inference by using the predicates rdfs:range and rdfs:domain. Given the triples $(teaches, rdfs:domain, Teacher)$ and $(teaches, rdfs:range, Student)$ as well as the triple $(Bob, teaches, Alice)$, one can imply the facts that Bob is a Teacher and Alice is a Student, $(Bob, rdf:type, Teacher)$ and $(Alice, rdf:type, Student)$.

### 3.1.1.3 Product Ontologies

According to the section 2.1 of the previous chapter, product lifecycle processes are executed by various human and machine actors who collaborate across different borders. In order to allow such collaborations, reliable communication is necessary [46]. A prerequisite of communication of computers is sharing the same syntax and semantics. The expression of meaning to syntax is semantics which both partners have to agree upon prior to communication. For successful exchange of product data it is necessary to agree upon a product classification standard [82]. For example, during purchasing of a product it is necessary to query a product in the manufacturer’s database based on some properties. These properties are standardized by some product classification schemes.

Figure 3.2 displays relevant product ontology modelling dependencies in the engineering industry in more detail. The meta-model tier describes the meaning of classes, properties and their relationships (e.g., RDFS or PLIB [65], [66]). The domain ontology tier describes characteristics of products (e.g., product classification eClass [37] or UNSPCS
For example, on this tier a pump with the properties **height** and **weight** would be specified. Instances of such domain ontologies are created in the **logical instance tier** (e.g., a product with attribute values which are edited by a manufacturer). For example, this tier would specify a concrete pump with order number 1906, a height of 1 meter, a weight of 200 kg and a price of 2000 Euro. The fourth **physical instance tier** represents manufactured instances of products (e.g., a product with serial number which contains the attributes which are described on the logical tier). The following subsections will give a more detailed description of the data model of two widespread product domain ontologies.

**UNSPSC**

The UNSPSC (United Nations Standard Products and Services Code) [136] whose data model is depicted in figure 3.3 is a pure hierarchical classification with no properties. The classification consists of four levels including **segment**, **family**, **class**, and **commodity**. Each level contains a two-character numerical value and a textual description. An additional 2-digit suffix indicates the business function identifier.
3.1 State-of-the-Art

The eClass standard [37] whose data model is shown in figure 3.4 is a hierarchical system for grouping products according to a four-level class structure. The first level is called the segment, the second is the main group, the third is the group and the fourth level is the commodity class. Late releases of the eClass ontology are based on the PLIB metamodel. Whereas UNSPSC only allows defining a hierarchical system for products, eClass also allows defining properties for the description of products which are associated only to the fourth level commodity class. Hence, eClass is an ontology which allows describing the meaning of product properties.

![Diagram of eClass Data Model](image1.png)

Figure 3.4: eClass Data Model. Adapted from [79]
3.1.1.4 Product Ontology Evolution

Ontology evolution [55] [48] [141] deals with the fact that ontologies are typically not static but rather evolve over time. The work [75] identified that changes made to ontologies are executed at each of the levels of figure 3.1:

**Conceptual Change** A change in the interpretation of the domain of interest. Changes in the domain occur when some new concepts or new relations between them appear or are modified according to the real world evolution.

**Specification Change** A specification change can be independent from a change in the conceptualization or it can be a result of a conceptualization change. Changes in conceptualization result from a changing view of the world or from a change in usage perspective.

**Representation Change** Changes in representation only takes place when an ontology is translated from a knowledge representation language to another one, possibly with different semantics and expressive power.

Domain and conceptualization changes are typical for everyday product ontology maintenance and are very frequent. According to [61] the average amount of new and modified classes per month is 279 for eClass and 233 for UNSPSC. Table 3.1 provides an overview of updates [40] made to the ETIM (ElektroTechnisches InformationsModell) product ontology [41]. Relocating a class means to move the class up or down in the hierarchy or horizontally and merging two classes means two classes become one new class or one class is appended to another class.

Product ontology updates are documented in spreadsheet documents and are distributed to ontology users [40]. The documents contain hints, how to rearrange dependent databases. However, the hints are not operational and cannot be interpreted by computer software. Therefore, it is necessary for ontology users to execute these ontology updates on their own.

**Ontology Updates and Alignments**

As a consequence of ontology evolution, the definition of classes and relations on the specification level has to be updated [75]. Special updates packages define the relationships between different versions of the same domain ontology [77]. Such update packages are expressed as structural diff, patch or log of changes between two versions of the same ontology [112]. Updates can be captured by tools while changing an ontology [133] or it can be computed by algorithms without knowing about the actual
changes [43]. Ontology updates have to be represented in some language to be operational. [127] provides a decent overview of existing approaches of ontology update representations.

This thesis regards *ontology alignments* [38] (or *ontology mappings*) as the core for representation of metadata update packages. Ontology alignments are the outcome of *ontology matching* [55]. Ontology matching is the process of identifying the relationship between versions of different domain ontologies. Ontology matching is relating similar (according to some metric) concepts or relations from different sources to each other by relation (e.g., equivalence) and is needed for determining differences between two ontologies that represent different conceptualizations. Alignments can be computed by machines or can be edited by humans. Often alignments are adjusted by humans after they have been generated by machines. Therefore, alignments represent a valuable resource which should be reused during the creation of new ontology versions.

### Table 3.1: Product Ontology Update Patterns. From [40].

<table>
<thead>
<tr>
<th>Level</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value Changes</strong></td>
<td>Change value title</td>
</tr>
<tr>
<td></td>
<td>Delete value from a property</td>
</tr>
<tr>
<td></td>
<td>Add value to a property</td>
</tr>
<tr>
<td><strong>Property Changes</strong></td>
<td>Add property for a class</td>
</tr>
<tr>
<td></td>
<td>Delete property for a class</td>
</tr>
<tr>
<td></td>
<td>Change of value list of a property</td>
</tr>
<tr>
<td></td>
<td>Change of property title</td>
</tr>
<tr>
<td></td>
<td>Change of unit of a property</td>
</tr>
<tr>
<td><strong>Class Changes</strong></td>
<td>Add new class</td>
</tr>
<tr>
<td></td>
<td>Change class name</td>
</tr>
<tr>
<td></td>
<td>Delete class</td>
</tr>
<tr>
<td><strong>Hierachy Changes</strong></td>
<td>Relocate class</td>
</tr>
<tr>
<td></td>
<td>Relocate whole subtree</td>
</tr>
<tr>
<td></td>
<td>Merge two classes</td>
</tr>
<tr>
<td></td>
<td>Split class</td>
</tr>
</tbody>
</table>
Alignments have to be represented in language to be interpreted by machines or humans. [9] provides an overview of existing mapping approaches and proposes a mapping language that includes transformation rules based on standard vocabularies. The standard vocabularies for alignment expression include `owl:sameAs` [54] for instance equality as well as `owl:equivalentClass` and `owl:equivalentProperty` [143] for schema element equivalence. Another possibility is to use N3 rules [83] for transformation of RDF graphs. The work of [8] describes the problem of comparing two RDF graphs, generating a set of differences, and updating a graph from a set of differences. It provides an update ontology for patch files for RDF and discusses experience with proof of concept code. The work of [42] expresses alignments by a dedicated vocabulary. When using this alignment API it is required to define the two entities which are related to each other, the arity and the relation type of the alignment. The work of [90] uses a semantic bridging ontology (SBO) to describe the relationships of entities and to specify transformations to translate instances. Finally, [56] provides a good overview of techniques for metadata interoperability techniques based on meta-model layers.

**Compatibility of Ontology Updates**

Compatibility is a relation between two ontologies of the same domain [103]. Two ontologies can be backward or forward compatible with each other. *Backward compatibility* is the ability of a new system to accept data that worked under a previous version of the system [149]. Backward compatibility is transitive and only applies to succeeding versions of an ontology. *Forward compatibility* is the ability of an old system to accept data generated by a new version of the system [151]. An ontology update could lead to incompatibility of data that was modelled according to a previous ontology version. If data becomes incompatible, systems which process this data might lose the ability to work together (loss of interoperability).

When making a statement about compatibility the perspective is relevant. An ontology $O_t$ at time $t$ can only be backward compatible with previous versions at time $s, s < t$. Whereas forward compatibility is a relation that has to be seen from the opposite perspective. An ontology $O_t$ at time $t$ can only be forward compatable with future versions of the ontology at time $s, s > t$. Therefore, when performing a change at time $t$ it is impossible to say that an ontology is forward compatible, since nobody knows which changes will be done in the future. Thus, the latest version of an ontology is never forward compatible. If a change is performed on $O_t$ that results in an ontology $O_{t+1}$ one can make two statements:

1. $O_t$ is (or is not) forward compatible to $O_{t+1}$
2. $O_{t+1}$ is (or is not) backward compatible to $O_t$
3.1 State-of-the-Art

<table>
<thead>
<tr>
<th>Translations</th>
<th>Major Release</th>
<th>Minor Release</th>
<th>Service Pack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correction of clerical errors</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Addition of classes at all levels</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Addition of keywords / synonyms</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Addition of properties in sets of properties</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Replacement of properties in sets of properties</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Addition of values in sets of values</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Replacement of values in sets of values</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Change of preferred names</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Removal of keywords / synonyms</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Moving of classes and class structures</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Splitting of classes and class structures</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 3.2: Product Ontology Updates and Release Planning. From [36]

The backward compatibility of ontology updates can be used for release planning. For example, the release planning for eClass (table 3.2) includes the notion of major releases, minor releases and service packs. The changes that are allowed in such releases are different in their consequences and in their frequency. For example, version 5.1.2 denotes the fifth major release (every 3 - 4 years), the first minor release (once or twice a year) and the second service pack.

Some ontology updates are backward compatible and some updates are not backward compatible. For example, adding a class or relation to an ontology is a backward compatible change. However, deleting a class from an ontology is a non-backward compatible ontology update. Archived metadata is threatened by non backward compatible updates. Since semantics are necessary for a complete understanding of product designs, archived semantics have to be regarded as regular data that needs to be preserved. However, the described distributed nature and the various dependencies make the preservation harder than just to transform data of a file format from one version to the successor file format version.

3.1.2 OAIS Archives

Digital preservation systems are archiving systems that aim to solve such issues by providing build-in preservation support. Often these preservation systems are based on
the Open Archival Information System (OAIS) [23]. This ISO standard specifies a high-level reference architecture for a long-term archive that consists of an organization of people and systems that have the responsibility to preserve information for a designated community of a specific domain. The OAIS provides a conceptual reference architecture and does not specify a concrete implementation. Therefore, the OAIS reference model has the flexibility to be used in a wide range of environments. Implementations of the OAIS reference model have to detail the workflows described in the OAIS standard including preservation planning methods like emulation and migration. This is necessary since the applicability of preservation methods depends on the application domain, the use cases, and the archived digital objects. The OAIS model consists of an environment, a functional and an information model.

### 3.1.2.1 Environment Model

The OAIS environment model (figure 3.5) consists of the OAIS archive and three external actors. A *producer* is a person, organization or system that provides the digital information to be preserved. A *consumer* is a person, organization or system that accesses the OAIS systems to find and acquire preserved information. The *designated community* which may be composed of multiple user communities is an identified group of potential consumers who should be able to understand a particular set of information. Finally, the *management* are those actors or organizations who set the overall OAIS policies in the sense of management’s responsibilities, not the daily operational archive administration which is shown in figure 3.6.

![Figure 3.5: The OAIS Environment Model. From [23].](image)

Archive consumers in the area of product lifecycle data can be identified according to the reuse use cases which were described in the previous chapter. The consumers include future engineers, investigators, cooperating companies, governmental authorities and regulatory agencies.
3.1 State-of-the-Art

3.1.2.2 Functional Model

The OAIS functional model consists of six major functional entities. In figure 3.6, these entities are depicted as rectangles, data flows are represented as arrows and different data types which are described in more detail below are represented as circles. The Ingest functional entity accepts one or more Submission Information Packages (SIPs) and creates Archival Information Packages (AIPs), which it provides to Archival Storage for preservation. Ingest also sends Descriptive Information (DI) to the Data Management functional entity. Consumers interact with the Access function, which uses the descriptive information to find the content information of interest. Access retrieves AIPs from Archival Storage and sends Dissemination Information Packages (DIPs) to the consumer. Administration oversees the day-to-day operation of the archive, and it receives advice from Preservation Planning on strategies and mechanism for preservation. All six functional entities are further broken down in the OAIS reference model.

OAIS-based preservation systems might use some of the following preservation methods:

- Migration is the continuous translation of data between different versions of the same data formats as well as other data formats.

- Transformation is the conversion of data from one format to another format that is assumed to be more preservable.

- Emulation (virtualization) is the implementation of functionality of a system running on another system.

![Figure 3.6: The OAIS Functional Model. From [23].](image-url)
3.1.2.3 Information Model

The OAIS reference architecture also defines an information model. The information model defines the information types (both content and metadata) that are required in order to preserve and access the data and provides a high-level description of the information objects managed by the archive. As described above, a Submission Information Package (SIP) is ingested by a producer into the archive and an Archival Information Package (AIP) is generated and then stored and preserved by the archive while a Dissemination Information Package (DIP) is delivered to the archive consumer. Figure 3.7 displays a SIP and related standard preservation metadata schemas which are now described in more detail:

![Figure 3.7: A SIP Package and Standard Preservation Metadata Schemas. Extended from [87].](image)

**Preservation Description Information** enables adequate preservation of content information and the PREMIS (Preservation Metadata Implementation Strategies) [114] standard is used to describe provenance (migration logs), context (related data), reference (persistent id) and fixity (checksums) of the content information.

**Packaging Information** aggregates and identifies the constituents of an information package. Here, METS (Metadata Encoding and Transmission Standard) [84], OAI-ORE (Open Archives Initiative Object Reuse and Exchange) [108], XFDU (XML Formatted Data Unit) [25], and BagIt [101] are used to describe the ingested information package.

**Descriptive Information** supports the search and retrieval of archived information. The standards MODS (Metadata Object Description Schema) [85] and DC (Dublin Core) [35] are used as metadata schemas. Descriptive metadata may be captured manually or extracted automatically [92] by inspecting the data object at data creation or submission time.
3.1 State-of-the-Art

**Content Information** contains the digital object (or, Content Data Object) to be archived and the **Representation Information** which, accompanying a digital object, provides it with meaning and allowing for the recreation of its significant properties. Representation information may contain references to other representation information. It associates high level meanings with one another and can have complex inter-relationships, building what is termed a *representation information network*. It contains semantic information that needs a fixed schema in order to be understood.

### 3.1.2.4 Metadata Management

It was described above that the OAIS reference architecture defines both a high-level functional model and an information model. The functional model outlines the functionality which has to be provided by a compliant preservation system - access, administration, archival storage, data management, ingest, and preservation planning. The data management entity provides all functions required for maintaining the descriptive metadata which identifies archived digital content. In addition to descriptive metadata, the data management also processes preservation metadata. Although preservation metadata and descriptive metadata schemas have mainly been created for the library domain, they are also used for archiving product data models. Further, the OAIS model was designed to be applicable to "any archive" or organization with information requiring long-term preservation [23].

### 3.1.3 Long-term Digital Preservation in PLM

Long-term digital preservation has always been on the agenda for libraries [58]. Several projects aim to study, develop, and implement methodologies for digital preservation. Some of the major digital preservation research projects include CASPAR (*Cultural, Artistic, and Scientific Knowledge for Preservation, Access and Retrieval*) [21], PLANETS (*Preservation and Long-term Access through NETworked Services*) [111], DPE (*Digital Preservation Europe*) [34], and APARSEN (*Alliance Permanent Access to the Records of Science in Europe Network*) [4]. An overview of these projects is provided in [134].

In recent years long-term digital preservation has also become a topic for the whole industry that uses information technology [63]. One application domain is the design and engineering domain which includes the industry sectors like automobile, aero plane, construction, etc. The engineering domain faces several important legal and economic requirements for implementing long-term digital preservation processes [62]. Several past and ongoing projects which are engaged with the preservation of engineering information
exists [18]. These projects include LOTAR [115], Knowledge and Information Management (KIM) [74], VDA Recommendation 4958 [140], MOSLA [96], Digital Engineering Archives (DEA) [78], Sustaining Engineering Informatics (SEI) [89], AncarPLM [7], and FACADE [44]. Below, the scope, motivations, goals, identified problems, and proposed solutions of these projects are studied. In order to distinguish the characteristics and to extract the key features of the studied projects, the next section specifies important perspectives regarding scope and functionality.

3.1.3.1 Engineering Domain Perspectives

As the OAIS model does not give guidance, how to implement the preservation of a concrete application domain, it is required to study preservation projects which focus on the archival of product data created in the engineering domain. These preservation projects have been evaluated based on aspects which were identified as important for the digital preservation of product lifecycle data. In this section these preservation aspects are presented in order to categorize the considered projects in later sections and to highlight their key project features.

Industry Sector The design and engineering domain is characterized by a large amount of industry sectors which include the aeroplane, automotive, chemical, electronics, energy, shipbuilding, and construction industry. It is not surprising that these industry sectors differ. The generated product data, the tools, the use cases (e.g., 3D architecture vs. aeroplane industry) for accessing a product data preservation system, and finally the executed processes heavily depends on the industry sector. Based on these industry sector characteristics, different motivations and requirements for digital preservation can be derived which should be considered by long-term preservation processes.

OAIS Awareness As described above, the OAIS is a conceptual reference architecture for arguing about long-term preservation systems. Such a reference model provides a common vocabulary, high-level data model, responsibilities as well as functionality, and it also helps to promote consistent thinking and approaches among the people who use it. In addition, existing OAIS frameworks and implementations can be used and enhanced. Therefore, it is preferrable to embed project outcomes into the OAIS reference architecture.
3.1 State-of-the-Art

**Product Lifecycle Data Awareness**  The product lifecycle is assisted by software of a PLM tool chain which spans from idea generation over production to recycling. A large variety of engineering data is generated during execution of various product lifecycle phases. The heterogeneity of product lifecycle not only originates from different phases, but also from different formats, data models, tools, and tool versions. This large amount of heterogeneous data and metadata is necessary for later reuse. Long-term preservation of engineering data is only successful if data generated throughout the whole product lifecycle is supported.

**Preservation Strategy**  Archiving product data in native format contains the risk that new software releases are not able to interpret the data. Therefore, the product data which has been ingested into long-term archival systems has to be preserved by using different strategies. The strategies include constant migration, emulation, initial transformation, and system preservation [146]. System preservation ensures the long-term accessibility of data byarchiving the software (tools, operating system, etc.) which is able to interpret the data. However, such system preservation implies to archive the people that are able to work with the software. The preservation method has to be selected under the guarantee that substantial tool support exists and that during initial transformation and subsequent migrations minimal information loss is likely. It has to be verified which of the preservation strategies were chosen by the different projects to keep engineering data and metadata up-to-date with contemporary technologies. Since the preservation success heavily depends on the preservation method, it is worth showing which preservation method was chosen by the various projects.

**Archive Distribution**  In PLM processes it is common that products are developed in geographically distributed environments. Different parts of product data are exchanged by collaboration partners and the data is stored and archived in different physically distributed repositories and archives [152]. Whereas physical archive distribution uses several archive systems for storing the whole product model, logical archive distribution uses only one archival system. In both cases it is required that different company rights on parts of the product model are maintained and respected. This access management is an issue to solve since the archived data is subject to intellectual property rights of individual companies. For example, if access is granted to a manufacturer by a supplier to view archived data then it might need to be assured that some parts are invisible and are protected from unauthorized access. Also, one company might want to protect their knowledge in the general case. However, in some use cases this knowledge needs to be disclosed. In addition to access management, there is another issue to solve when archives are distributed. If archives are distributed it is not guaranteed that each collaboration partner uses the same file format, tool, tool version, and preservation
system. Therefore, consolidated migration steps are necessary in order to keep the distributed data in a consistent state. Therefore, for successful long-term preservation has to consider issues regarding archive distribution spanning over company and archive boundaries.

**Copyright and License Management** Vendors of commercial or proprietary software often use copyright for file formats and use software licenses to restrict the customers. If native file formats are archived, a vendor dependency exists as long as the data is kept in the preservation system [98]. Copyrights also apply to external data from third parties which needs to be archived for full understanding of the main product data. Also, the short-term duration of software licenses for tools which allow to work with the product data is a threat for long-term future reuse of archived product data. These non-technical issues are important for long-term preservation since preservation can only be successful if continuous access to archived product data is guaranteed. Hence, preservation processes should be aware of licence management which allows to support all necessary steps to preserve archived data.

**Formal Semantics** The engineering domain is characterized by knowledge intensive processes [142]. If this knowledge is not captured manually or extracted automatically, it will be difficult to reason about, search for, and reuse this knowledge. Product life-cycle knowledge should be captured formally by ontology-based metadata in order to provide machine readable meaning. Therefore, for better product understandability and search results it is required that engineering semantics is annotated to product data with knowledge representation languages like ontologies. Preservation processes should support this kind of formal knowledge capturing and should support the metadata harmonization after the knowledge has been archived.

**Engineering Preservation Policies** Preservation of product data is important due to legal and business motivations which might have major economic impact. Even small mistakes during preservation can cause critical consequences. Therefore, since preservation is new and important to companies which use product lifecycle data it is desirable to execute data preservation activities along a well predefined preservation process. For example, if a specific industry sector requires to archive a product data for at least ten years, the preservation systems have to execute this business requirement. The individual business requirements can be expressed as executable preservation policies which also can specify policies expressing local company demands.
3.1.3.2 Project Descriptions

The following subsections will provide an overview of the projects mentioned in section 3.1.3.

LOTAR

The *LOng Term Archiving and Retrieval* (LOTAR) project [115] studies the applicability of the STEP (*STandard for the Exchange of Product model data*) standard (ISO 10303) [64] and OAIS standard (ISO 14721.4) [23] for archiving digital technical product data which is exchanged between partners and suppliers within the aerospace industry. The LOTAR project derives motivations for long-term archiving from legal requirements (certification, product liability) and business requirements (retain intellectual property, product reuse, support of product operation). The business need becomes evident when considering that the product lifecycle of an aircraft is up to 70 years and product data that is relevant for certification has to be archived for up to 99 years.

The LOTAR project defines archiving relevant subsets of product lifecycle data such as common PDM information (organization, person, date, time), product structure information (part, assembly, views, relationships), change management information, configuration management information, and process management information. LOTAR proposes a transformation of the original product data (geometry and technical data) to a standardized and vendor neutral format for long-term archiving with a minimized loss of semantic information. The transformation to STEP process might result in a loss of information since a product data is interpreted by software. As a consequence, it needs to be assured that a new software generation correctly interprets a previously archived product data. To achieve this demand of information archiving without loss of semantic information it is necessary to *validate* and *verify* the data before it has been ingested to the archive (figure 3.8).

The *validation* process compares two product data models against each other. While transforming the native format, a consistency check based on mandatory and optional validation properties is executed. Therefore, key characteristics of the model are defined which help to verify the model after transformation by checking if verification properties have been retained. Example validation properties for geometric models include surface area, volume and centre of gravity. The validation process calculates these geometric properties by using the values provided by the product data and compares the source and the transformed data against each other. The *verification* process compares a model against a set of rules. In addition to validation, a verification of both the source and
State-of-the-Art and Related Work

Figure 3.8: Data Verification and Validation. From [22]

the transformed data takes places. An example of a verification rule is to check whether there exist cracks within the shape.

LOTAR defines a mandatory set of these verification rules for the CAD model and validation properties to be created during the ingestion and to be checked during the retrieval process. For 3D-CAD data these verification and validation properties are based on thresholds. The threshold values are not fixed since the results are subject to numerical errors in the algorithms of the CAD software. LOTAR defines adaption points where each participant can define thresholds respecting their own specific processes and products.

KIM

The Knowledge and Information Management (KIM) project [74] (University of Bath, Cambridge, Heriot-Watt, Imperial College London, Lancaster, Leeds, Liverpool, Lough-
3.1 State-of-the-Art

borough, Reading, Salford, and Strathclyde) explores strategies to handle information and knowledge over the lifetime of product services. One of the main investigations is the long-term preservation of digital product data, documentation of design processes and rationale in the engineering domain. Motivations of long-term archiving of engineering data are derived from legal requirements (accident investigation, failure analysis, mergers and acquisitions, patent infringement), operational support requirements (maintenance, replenishment of spare parts, recycling, disposal) and product development and management requirements (tracing design rationale, design reuse, customization and upgrade, reverse engineering, test and validation). In addition to the legal and business requirements the following preservation challenges are identified.

- Product data which is generated during PLM processes spans from highly structured (geometric models) to completely non-structured (design reviews) and also contains knowledge (design rationale). However, the heterogeneous data and the associated knowledge are also subject of long-term archiving.

- Tools that are used along the PLM phases are not well integrated since they are sold by different vendors which use native file formats. Native file formats are incomplete because relevant information which is needed to interpret product data correctly is kept within the software. In addition, native file formats are closed because the vendors do not publish specifications about their formats. A standardized product data format would help to integrate existing and future tools.

- Since the product lifecycle lasts for several decades or even longer, the traceability and retrieval of information and knowledge has to be assured even if a new tool generation is introduced.

- Nowadays, collaborations are executed in geographically distributed environments. The product data representations need to be tailored for such collaborations because during such collaborations valuable information and knowledge is exchanged.

In addition to these challenges, the project also chooses the preferred preservation method due to the following arguments: emulation of old software is difficult since these tools have to be integrated into existing systems along the PLM phases. A major issue with migrating old product data to newer formats is that there is always the risk of information corruption and the costly data validation after migration. The transformation of product data into the STEP standard is regarded as the right choice. However, for some applications, STEP has shortcomings:

- The lack of interoperability between complex and overlapping STEP Application Protocols (AP) lead to the development of a STEP modular architecture.
• The inability to capture the design intent (user-defined features and constraints) and construction history.

• The difficulties associated with implementing STEP (long time period between standardization and vendor support).

• The level of tool support can vary between vendors so that model exchange is difficult because it is not affordable to create one’s own STEP-based solutions.

• The exchanged STEP-based models are difficult to modify.

To solve the described requirements and challenges without using the STEP standard, the KIM project proposes a solution based on two key features. The first key feature is the use of Lightweight Models with Multilayered Annotations (LiMMA), i.e., a lightweight product data representation as an exchange and preservation strategy. The second key feature is the Registry/Repository of Representation Information of Engineering (RRoR-IfE). Both features are now described in more detail.

Lightweight representations are application independent and do not try to retain all information of a full CAD model since they only contain as much information as a participant needs. Significant properties which are important for the model validity are retained. The KIM project extends its focus to beyond just the early PLM phases and tries to cover the whole PLM process. Lightweight models are considered both for the product lifecycle (XML, EXPRESS/XML, PLM XML) and for 3D visualizations (U3D, HSF, XGL/ZGL, X3D, 3D XML, JT). All these lightweight models are described in more detail in [30].

Using lightweight formats has several advantages:

• By using open, lightweight formats for data exchange and preservation it is more likely that the design work remains readable and usable in the future than fully-featured, closed, proprietary CAD formats.

• Since transformation to lightweight formats is executed at design time, a validation can be performed.

• The transformation to lightweight formats results in a reduced file size which makes collaboration exchange and preservation easier.

• Since only as much information as needed is retained, it is easier to implement new software which is able to interpret the data.

• Lightweight formats offer customized views (depending on PLM phase) and security of intellectual property.

• Since there is no restriction to the use of only one lightweight format, information can be divided and linked between several lightweight formats.
As an important aspect, LiMMA combines open and lightweight CAD-formats with text annotations to augment geometry data with knowledge information from different PLM phases. Those annotations are not described in a formal way. Annotation layers exist for different phases (design, manufacture, service and marketing) as well as different security and access levels. Annotations include design rationale, context, provenance, and access restrictions. Annotations are held externally and are not included in the product data itself. This is an advantage because the same annotation can be used for different representations and annotations can be applied differently in several contexts. All users throughout the product lifecycle are able to create their knowledge and experiences through annotations rather than modifying the CAD-model itself.

The second key feature of the KIM project is the Registry/Repository of Representation Information of Engineering (RRoRIfE). RRoRIfE is a representation information registry (adopted from concept of representation information that is used in the OAIS reference architecture) for engineering specific file formats. Representation information is any information required to render, interpret, process, and understand data (software tools for interpretation, file format specifications, algorithms, standards). RRoRIfE has three major use cases. The first use case is the preservation planning which helps to migrate from one format to another format by providing possible migration paths between two formats. During such migration it is checked if information is at risk. The second use case is querying for file formats matching a given set of functional requirements. The third use case is looking up the characteristics of various file formats so RRoRIfE can be regarded as a reference tool. All these use cases are possible, since RRoRI stores two different information sets. The first set is the set of significant properties of CAD file formats (2D and 3D geometry entities, modeling history, size, compression, geometry-related metadata like tolerances) and the second set is the set of processing software characteristics (record of how well each property is preserved for each format conversion).

VDA Recommendation 4958

The VDA (Verband der Automobilindustrie) recommendation [140] aims to define requirements regarding the processes, data and organization roles necessary for long-term archival of digital product data which is generated in the automotive industry sector. When the recommendation was created in 2005 the long-term archiving of product data took place in the form of 2D drawings although most of the product data were available as 3D data. In the automotive industry sector both manufacturers and suppliers are responsible for the long-term archival of their respective product data. The requirements include:
• Each document has to be identified in a unique way.
• Each author or responsible person has to be identified.
• Each version of produced documents has to be archived.
• Each document has to be signed according to law.
• Data has to be preserved at least for 12 years after milestone 'end of production'
• After expiration of archive duration, the deletion of the documents is possible if this action is recorded.
• Documents in a long-term archive are read only and cannot be changed.
• Manipulations on archived data have to be evident and need to be recorded.
• Migrations to new data formats lead to new archived documents.
• A source document of a converted document can only be deleted if the conversion has been executed without loss of information.
• Move of archived data has to be recorded and commented.
• Organization and operation of archive system need to follow well-defined instructions of process in order to guarantee quality of data.

It is recommended that such archiving requirements need to be agreed upon contractually between manufacturer and supplier. This is necessary to ensure compliance with product-liability, documentary-proof requirements, and spare parts service. An automotive manufacturer has to be able to prove that vehicle data is kept for authority questions and law cases. Securing the quality and long-term availability of the data and the company know-how is also a business requirement. The fundamental methods (verification, validation after transforming to STEP) and processes which are used for preservation are the same that are used in the LOTAR project.

**MOSLA**

The *Maturity of Standard for Long-term CAD data Archiving* (MOSLA) project [96] specifies a digital long-term archive standard for 2D- and 3D-CAD data based on STEP standards focussing on the requirements of the aircraft and automotive industry. MOSLA defines LCAD202 and LCAD203 as a standard for long-term CAD data archiving. LCAD202 and LCAD203 are based on AP203 (Configuration controlled design), AP202 (Associative draughting), and Part 21 (Implementation methods, clear text encoding of the exchange structure) of the STEP standard. The LCAD standards divide the existing STEP standards into unsupported items (configuration design), ambiguous or
inconsistent items and specified items (indispensable product design management items like path that represent a product). Constraints are defined on the ambiguous or inconsistent items. In addition to the agreement document, the project also has created two software applications. The first is a “STEP Viewer” that supports the visualization of the defined standards and the second is the “STEP file creator” which creates files which conform to the standard.

Digital Engineering Archives

The Digital Engineering Archives (DEA) project [78] [120] was initiated at the Drexel University and examines methods for long-term archiving of 3D-CAD data. The project studied the possibility of long-term archiving of a huge amount of CAD models of the National Design Repository which is a digital library of computer-aided design data from domains such as mechanical design, architecture, and electronic design. During these studies, the following challenges were identified:

- The data size of the CAD models.
- The preservation of data change (temporal aspect) is important, particularly where something changed and why.
- The heterogeneity of data ranging from conceptual documents, CAD design, assembly, functional, behavioral, simulation, analysis, manufacturing, service, and maintenance to disposal.
- Only native, proprietary data formats were archived. These data formats cannot guarantee the readability within 50 to 75 year product lifespan.
- If 2D drawings were archived then knowledge in 3D CAD models about features, manufacturing processes, and artifact behavior are simply lost in translation.

To meet these challenges the following solutions are proposed: since native file formats are hard to preserve, the project proposes the use of shape information for preservation. These shape-based representations are annotated with formal models of engineering semantics. Therefore, the standards Ontology Web Language (OWL) [143] and Process Specification Language (PSL) [68] are used to represent engineering function and behaviour. The relationships of shape and form, structure and function, and behavior and semantics has to be captured in digital engineering archives. The availability of engineering semantics will enable the creation of systems for archival, retrieval (finding and accessing information easier) and reuse of engineering knowledge. To exploit those archives which include engineering semantics, an OAIS-based digital engineering archive was created which transforms the native format into the vendor-neutral
STEP format. A special Digital Archive and Retrieval Tool (DART) allows to create queries on these digital engineering knowledge archives. In addition, a format registry for engineering file formats is proposed. This Engineering Format Registry contains format descriptions which are described by the Resource Description Framework (RDF) [91].

Sustaining Engineering Informatics

The Sustaining Engineering Informatics (SEI) project [89] initiated by the NIST (National Institute of Standards and Technology) studies methods and metrics in the curation of engineering data. In [89] a definition is for the term SEI is provided: “Engineering informatics is the discipline of creating, codifying the syntax and semantics of, exchanging, sharing, processing, making decisions about, storing, and retrieving digital objects characterizing the multi-disciplinary domain of engineering.” The project aims to implement a framework which is more generic than LOTAR but more domain-specific than OAIS. The requirements of the designated community are defined by the “3Rs”:

- **Reference** is the ability to view and visualize the engineering data.
- **Reuse** means taking the data and to modify or to reengineer it.
- **Rationale** is the ability to display information such as construction history or design intent which is information that goes beyond the design itself. The rationale are the *why* questions that can arise from a design. Data models like STEP do not support rationale information.

AncarPLM

The AncarPLM (Analysis and characterization of PLM solution) project [7] of the university of Lyon attempted to model the data of a product along the different phases of its lifecycle. The model contains concepts of traceability. The traceability is ensured by archiving the initial data and the change in the data (delta). These concepts where implemented as a first prototype on the DSpace platform [128]. Unfortunately, only little information about the project is available.
3.1 State-of-the-Art

FACADE

The FACADE (Future-proofing Architectural Computer-Aided Design) project [44] researches methods and best practices to capture, describe, manage, preserve, and make available digital architectural data (e.g., CAD models) which are created by architects during house building projects. The use cases of the FACADE project are completely different than those of the other projects due to the different industry sector. The archiving is done by using the DSpace [128] digital archive. The archived objects not only include data and metadata but also relevant CAD software packages for future processing.

3.1.3.3 Summary

In this section the state-of-the-art of long-term preservation of product data is summarized by highlighting the perspectives of the engineering domain. First, the following table provides an overview of the project scopes which are then discussed below in more detail.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Project</th>
<th>Industry</th>
<th>Sector</th>
<th>OAIS</th>
<th>Awareness</th>
<th>Product Lifecycle</th>
<th>Awareness</th>
<th>Preservation</th>
<th>Method</th>
<th>Archive</th>
<th>Distribution</th>
<th>Copyright and License Management</th>
<th>System Preservation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Industry Sector Some projects restrict the application industry sector. LOTAR aims for the aerospace industry and VDA concentrates on the automotive industry whereas the MOSLA project spans over both industries. The FACADE project concentrates on construction and 3D architecture data while all other projects do not specify an industry sector preference.

OAIS Awareness All projects except MOSLA and FACADE embed their research and solutions into the OAIS reference architecture.

Product Lifecycle Data Awareness Only the KIM and the AncarPLM project concentrates on long-term archiving of data which is generated during the whole product lifecycle. The LOTAR and VDA project partially support product lifecycle data since they support change, configuration, and process management information.

Preservation Method There seems to be a consensus [110] that transformation into vendor-neutral file format is the preferred preservation method assuming that these formats are not evolving as fast as native formats. STEP usage is proposed in the majority of projects whereas the KIM project proposes to use lightweight formats due to explicit formulated arguments. The KIM and DEA project also aim to migrate different file formats by the use of a format registry. Only the FACADE project discusses the method of system preservation.

Archive Distribution Logical and physical archive distribution and in consequence the management of intellectual property rights (access management) is not the current state-of-the-art.

Copyright and License Management The management of licenses was in the scope of the FACADE project because the FACADE project also archives the relevant software which is able to interpret the archived data. Such commercial software is usually associated with software licenses.

Formal Semantics The DEA project uses semantic web technologies to describe function and behaviour of product designs. Special software has been implemented which allows easier searching of product designs via knowledge that has been captured in a formal way. However, it has not been discussed how this archived formal knowledge itself is preserved under knowledge evolution. The KIM project uses the concept of knowledge annotations where product data is annotated with process and knowledge data in all PLM phases. However, these knowledge annotations are not described by formal semantics.

Engineering Preservation Policies Although the VDA states explicit requirements for product data preservation, the definition of such requirements as industry sector specific policies which are defined and executed within the preservation system have not been treated by the existing projects.
After studying the state-of-the-art in the realm of engineering data preservation, it is evident that a lot of open challenges remain. While for some aspects a common denominator has been found, some other aspects have not been studied at all:

- The non-technical preservation management issues of copyright and license management and archive distribution remain open challenges.

- Aggregation of product parts which have been archived in distributed preservation systems into a whole product data model need to be investigated. During such aggregation it has to be taken into account that distributed data might not be in a consistent state regarding file format or format version. Also methods need to be found which allow to express and implement the protection of product data parts for access management.

- Specific engineering industry sector requirements expressed as preservation policies have not been touched. It needs to be verified, if the specification and implementation of such policies is possible using existing OAIS-based preservation frameworks and if such implementations allow company specific policy redefinition.

- All projects focus on the preservation of 3D visualization formats and there seems to be an agreement that transformation of data to vendor-neutral formats is the preferred preservation method. In the future it has to be verified if other preservation methods like system preservation or emulation achieve the same or better results with lower efforts and risks.

In addition to the project summary provided above, it is necessary to review if and how the current state-of-the-art of product data preservation has considered the thesis goals which were described in section 2.5. This analysis is executed below in section 3.3.

### 3.2 Related Work

Since this thesis was executed in parallel with work done in the integrated project SHAMAN funded by the European Commission, the next section describes how the SHAMAN framework is applied to support the development of the thesis goals. Therefore, the product lifecycle processes executed in the engineering domain and the generated product data and metadata are embedded into the SHAMAN framework.
3.2 Related Work

3.2.1 The SHAMAN Project

SHAMAN (Sustaining Heritage Access through Multivalent ArchiviNg) [125] is an EU project under the 7th framework programme. The project started at the end of 2007 and finished in November 2011. The goal of SHAMAN was to investigate and develop a long-term digital preservation framework in a grid environment to ensure an effective and distributed management of archived data. SHAMAN investigated the requirements of different application domains which are known as Integration and Demonstration Subprojects (ISP) including:

1. Memory institutions including scientific publishing in libraries and documents in governmental archives
2. Industrial design and engineering industry
3. Data resources used in e-Science applications

The processes of all these application domains produce large amounts of data in different file formats including the well known HTML, PDF, JPEG, PPT, and XLS, as well as DVI (Device Independent File Format) [150], SVG (Scalable Vector Graphics) [153], and JT [70]. As an outcome of the SHAMAN project, a multivalent engine [24] is capable of accessing and displaying archived documents of this large variety of file formats without altering or migrating. Another outcome is a concept for managing security within long-term archives based on executable policies [118]. In addition to these outcomes, the central SHAMAN approach to ensure that existing data will be understood by future users is the archive-centric information lifecycle of SHAMAN [13].

The archive-centric information lifecycle which is displayed in figure 3.9 includes the following phases:

**Creation** During the initial creation phase, new information comes into existence.

**Assembly** The assembly phase denotes the collection of objects relevant for archival. Assembly requires in-depth knowledge about the designated community in order to determine objects relevant for long-term preservation together with information about their reuse later in the future.

**Archival** During the archival phase the digital data is stored and maintained in a preservation system.

**Adoption** This phase encompasses processes by which information provided by the archive is examined, adapted, and integrated for reuse. The adoption phase might be regarded as a meditation phase to reshape data accessed from an archive to be used in contemporary software tools.
Reuse The exploitation of information in the interests of the consumer is reuse. In particular, these may comprise purposes other than those which the digital object was originally created for.

Pre-Ingest During pre-ingest which spans from the information lifecycle phase creation to assembly all activities are executed which have to be taken prior to the ingestion of data into the preservation system.

Post-Access The post-access phase which consists of the information lifecycle phases adoption and reuse comprises all activities which are needed for preparing the final access of the preserved data.

Being located within the second industrial design and engineering application domain, this thesis adapts the information lifecycle to the product lifecycles processes as follows.
3.2 Related Work

SHAMAN Information Lifecycle for Product Lifecycle Processes

In the engineering domain, information is created during all phases of PLM processes. In addition to the product data, product metadata is also relevant for archival since it is essential for maintaining the understandability of product data. Metadata has to be captured at data creation-time. Hence, the pre-ingest phase starts right at the data creation-time when metadata is captured and annotated to data in order to be archived as knowledge later on. All relevant product data and metadata is stored in various repositories which might span over system and even company borders. Since product data and metadata reference each other and is distributed over several repositories, it is necessary to assemble and aggregate the product data and metadata before archival. All the product data and metadata which is needed to correctly interpret a product design (geometry, coordinate list, metadata, product component libraries, simulation, and verification data) is collected and aggregated into one big packet and finally ingested. Since it is more likely that open and well specified file formats can be interpreted by software in the long-term, the SHAMAN project proposes the transformation of product data into the vendor neutral JT file format [70]. The JT file format has an open specification which can be implemented by interested parties. For example, the multivalent browser is able to display data in this file format.

After transformation of product data, special archival functionality which can be executed from the PLM system connects the PLM processes with the preservation system. To initiate the archival, a service interface of the preservation system can be used by the PLM system. After archival and removal from active repositories, archived product data has to be constantly accessible by the PLM system. Another service interface of the preservation systems allows to search and access product data. As the long-term archive is integrated into the PLM workflow, product data has to be interpretable by current system environments. Therefore, the archived data formats are adopted to formats which are interpretable with current tools. Native file formats and even standardized file formats need to be migrated. These post-access activities also include transformations that keep the metadata up-to-date. The adoption of product data and metadata enables its reuse. For example, an archived product design can be imported into a proprietary tool for creating product variations. If the annotated product metadata also conforms to contemporary standards, the product design can be understood more easily.

It has to be emphasized that in such preservation aware product lifecycle processes, the archival of metadata neither begins when product lifecycle data is ingested into the archive nor ends when data is accessed from the archive. Metadata preservation begins immediately after the creation of data which needs to be annotated with metadata (pre-ingest). If ontology-based metadata is captured, exploration, human and machine
State-of-the-Art and Related Work

understanding as well as reusing product data in daily PLM processes is easier. Since ontology evolution is frequent and archived product data and metadata has to be ready for continuously reuse, metadata harmonization processes are required. As OAIS-based archives currently lack the necessary functionality for harmonization of metadata, the following section reviews the related work.

### 3.2.2 Metadata Harmonization

To support the understandability of archived product data, metadata is required. Efforts do exist to create dedicated preservation schemas for product metadata. The works [87] and [88] aim to investigate "library metadata and packaging standards and how they relate to product metadata". However, the engineering domain is characterized by a variety of applications, heterogeneous lifecycle data, a diversity of actors, and a multitude of industry sectors. These facts make it very complicated and nearly impossible to define one metadata schema standard that covers all use cases for product lifecycle metadata exploitation. Therefore, it has to be expected that domain specific metadata schemas which are not controlled by the archive are used to annotate product data. These metadata schemas are represented by the semantic information that accompanies the representation information. Unfortunately, the preservation of such metadata has not been detailed in the OAIS reference model so far.

Some work has already been done to fill this gap and establish metadata preservation techniques. The work of [80] focuses on changing descriptive metadata when metadata evolve. According to [80] evolving metadata occurs "when the value of metadata changes over time, or when the value of metadata becomes obsolete". The work [28] introduces metadata schema registries into digital preservation systems. Registries "can disclose authoritative information about the semantics and structure of the data elements that are included within a particular metadata scheme" [28] and "registries could also contain authoritative mappings between different standards, thereby helping to facilitate the exchange of metadata or information packages between repositories and end users" [28]. The work of [126] is close to work of this thesis as it describes conceptual extensions of OAIS archives which preserves and manages metadata for "digital metadata curation".

### 3.3 Analysis and Discussion

After reviewing the state-of-the-art and related work, it can be stated that the full long-term preservation of metadata which is created during all product lifecycle phases is far from satisfactory. The following list summarizes the missing gaps.
3.3 Analysis and Discussion

State-of-the-Art of Preservation Focus on Geometry Data The review of the state-of-the-art revealed that long-term preservation of product data is mainly concerned with the archival issues regarding CAD-based geometry data.

Archives are not well Integrated into Daily PLM Processes While archives are used in the engineering domain as a safe haven for product data which only has to be accessed rarely, the full integration of long-term archives as a valuable company knowledge base into day-to-day PLM activities has not been considered yet. This archive integration ensures that at any given time a company can access the product data to give proof about compliance with regulations and quality.

Informal PLM Knowledge Annotation While some initial investigations for knowledge capturing by means of annotations have been executed, it has not been described how these annotations can be expressed with contemporary ontology-based semantic technologies throughout the whole product lifecycle.

Missing Consideration of Knowledge Evolution While capturing and annotation of metadata is the first step towards knowledge preservation, the long-term preservation of metadata after its archival has not been studied yet. Possible threats for archived semantics are currently not investigated and it is left untouched how to harmonize the captured and archived knowledge under terminology evolution and ontology evolution which are very common in the engineering domain.

Related Work Lack Completeness The study of the related work revealed that all of the related works generally lack completeness since they do not consider metadata harmonization during metadata schema (e.g., an ontology) evolution and do not try to achieve practical methods for automatic metadata harmonization.

OAIS Model Lacks Metadata Harmonization The investigation demand regarding methods for metadata harmonization across time is even bigger, since even the OAIS reference architecture also does not specify metadata harmonization functionality.

Due to all these identified gaps regarding long-term preservation of product lifecycle knowledge, this thesis considers these issues in more detail. Since the product lifecycle knowledge is expressed as ontology-based metadata, threats for archived metadata and appropriate harmonization strategies have to be investigated. Therefore, the next chapter describes functionality for product lifecycle metadata preservation. This preservation includes dedicated metadata harmonization functionality which supports ontology updates which might be backward incompatible. This harmonization functionality provides concepts which keep the understandability of metadata with current technologies and languages. In addition to harmonization functionality also required functionality for ontology engineering tools is described.
4 Modeling Metadata Preservation

The last chapter revealed that the state-of-the-art in long-term product data preservation did not focus on the preservation of knowledge which is generated during the product lifecycle. It also became evident that OAIS archives currently lack dedicated metadata harmonization functionality. Therefore, automation support concepts and corresponding system components are needed which guarantee successful metadata harmonization under ontology evolution. The first section of this chapter provides a top-down description of the main system components which are involved in metadata preservation. Then, a description of the necessary metadata harmonization functionality is provided. Based on the characteristics of this functionality, requirements for ontology engineering system functionality and functionality in a corresponding representation model and their interfaces are derived. This functionality includes the recording of the history of domain ontologies which is then used in open archive information systems for the management of archived metadata.

4.1 Introduction and Overview

The metadata preservation system components which are shown in the system model in figure 4.1 allow to implement important features which were identified as absent during the analysis of the state-of-the-art in long-term preservation of product data. These missing features include the

- integration of the long-term archive information systems into daily business work-flows
- manual and automatic capturing of knowledge throughout the product lifecycle
- product lifecycle knowledge expression as formal ontology-based metadata
- semi-automatic OAIS-based harmonization of archived metadata under knowledge evolution

The functionality and interaction of the following three main system model components allow to implement these missing features:
PLM Systems (at the left in figure 4.1) use and execute the workflow tools (e.g., CAD design software) which generate data and metadata. The data is stored in special PDM repository systems and a special metadata repository system stores the metadata. Additional tools allow accessing the repository system and the underlying open archive information system via browsing of metadata schemas. These tools are able to query both the repository as well as the metadata repository because the metadata references the active data repository via annotations. By querying and finding metadata, product data parts can be explored.

Ontology Engineering Systems (at the right in figure 4.1) allow editing the metadata and the associated schemas. The tools trace ontology evolution and document the impact of ontology updates. These tasks are modeled as Domain Knowledge Provenance Graphs (DKPGs) which are maintained by ontology engineers and domain experts. While editing the ontology, update packages are maintained semi-automatically. Update packages which have been generated during the evolution phase can be exploited in dedicated harmonization functionality. In addition, ontology alignments which map two different domain ontologies are created by ontology matching tools and further maintained by alignment engineers.
Long-term Archive Information Systems (at the bottom in figure 4.1) store the data and metadata which have been ingested from the active repository system when specific points in time of the business workflow are reached. The long-term archive also contains an access and query service which allows to explore the archived metadata. The metadata harmonization functionality uses operational ontology updates in dedicated functionality which is operated by archive curators and local domain experts. Operational ontology updates which are published by ontology engineering tools need to be communicated to long-term archives.

The characteristics of product lifecycle processes were already described in section 2.1 and an overview of generated product data and metadata was provided in section 2.2. According to the analysis of the state-of-the-art, it is evident that a description of archive system functionality for the harmonization of metadata is missing. Therefore, this chapter continues in section 4.2 with the description of necessary metadata harmonization functionality. This functionality impose requirements for ontology engineering which are detailed in section 4.3.

### 4.2 Metadata Harmonization Functionality

After the archival of annotated product data, metadata harmonization across time is regarded as the set of continuous processes that guarantee the correct long-term understandability of both ingested metadata for consumer access both now and in the future. As the metadata is referenced by annotations that identify ingested content, metadata harmonization also is a key instrument to maintain the ability for discovery of ingested content. This section describes the integration of necessary harmonization functionality in OAIS archives. Archives which contain metadata harmonization functionality have to be able to

- mediate between different metadata schemas
- cope with previously unknown metadata schemas
- handle domain as well as preservation schemas
- process different schema versions
- retrieve, store, and execute ontology updates
- preserve metadata, queries, and annotations under ontology updates
- understand and translate between different representation languages
- recognize and trigger necessary data updates after processing metadata updates
As described in section 3.1.2.2, the original architecture of the OAIS functional model splits the SIP into an AIP and descriptive information. Since it is not predictable which metadata will be included when a product data model is ingested into the archive, this thesis proposes a special metadata harmonization functional entity and metadata storage functional entity which are marked as light gray in figure 4.2. Special metadata processing of the ingest functionality will divide the SIP into an AIP and a metadata information package (MIP) which will flow to the harmonization functionality.

Figure 4.2: The Metadata Harmonization and Storage Entities. Derived from [23].

**Metadata Storage**  The metadata storage entity provides the functionality for the storage and querying of MIPs and associated metadata schemas. It receives the MIP and metadata updates packages (MUP) from the metadata harmonization and it provides the MIP to the access functionality entity in order to create a DIP. If a consumer queries or accesses the archive, the metadata harmonization functionality translates queries or transforms the metadata. Metadata can be stored and referenced in different ways [113]. In this thesis it is assumed that the metadata and the schemas are stored within the archive using the metadata storage entity.

**Metadata Harmonization**  The metadata harmonization functional entity (see figure 4.3) handles all descriptive, preservation, and domain metadata which is included in the SIP.

Metadata harmonization includes the following sub functionality:

- *Metadata Normalization*. Metadata conforming to a global (externally maintained and independently evolving) schema can be normalized into metadata conform-
4.2 Metadata Harmonization Functionality

Metadata Transformation. During access and upon request by the consumer, metadata update packages can be exploited to transform metadata conforming to schema X into metadata conforming to schema Y or to transform metadata from one version of schema X to another version of the same schema X.

Metadata Query Mediation. Instead of the metadata, an incoming query is rewritten so that it conforms to the archived schema. The result set is also processed for correct interpretation.

Metadata Migration. Upon request from the administration entity, metadata can be migrated so that it conforms to a new version of the same metadata schema or to another domain schema.

Metadata Monitoring. If ontologies evolve, the ontology engineering tools publish updates. Metadata harmonization functionality will monitor ontologies which have been archived.

Metadata Update Management. When ontologies evolve, metadata update packages (MUPs) can be identified and then be used to migrate the archived metadata, to rewrite a query or to transform the metadata at request time.

These subtasks are described in more detail below. The following abbreviations are used throughout the following sections:

Figure 4.3: Dedicated Metadata Harmonization Functionality
### 4.2.1 Ingest

The ingest functionality will divide the SIP which it receives from the producer into an AIP and a MIP. In order to do so, the ingest functionality could parse the file types of the ingested data collection or could demand that all metadata is stored in a special sub folder of the SIP. Also, the packaging information can describe relevant metadata packages. Additional MIPs can also be created automatically during ingest by producing descriptive and preservation metadata.

![Quality Assurance and Normalization During Ingest](image)

Figure 4.4: Quality Assurance and Normalization During Ingest

Figure 4.4 displays the ingest process. First, the quality of the MIP is assured, then a semantic normalization of the MIP is executed which transfers the metadata into a local schema. After that, the metadata representation might be changed. As semantic and syntactic normalization are based on the preferences of the local archive administration they are optional steps. Finally, the last step of the ingest functionality is to deliver the MIP to a special metadata store. These ingest functionality is now described in more detail.

#### 4.2.1.1 Quality Assurance

After identifying the metadata packages, the quality has to be assured by parsing the metadata representation for syntactic validity. The MIP has to contain ontology metadata that identifies the schema version to which the metadata conforms to. The schema might be a part of the MIP and can also be archived. But it is also possible, that schema information is not part of the archived content because it can be referenced and accessed
outside the archive system. This handling of metadata schemas can be described by a policy. However, it should be recommended that schema information is archived together with the metadata since in the long-term it is not unlikely that maintenance of domain schemas can be discontinued or that a schema hosting server is unavailable for an unforeseeable period of time.

Algorithm 1: Retrieve and Ingest Schemas Referenced by a MIP

<table>
<thead>
<tr>
<th>Data:</th>
<th>MIP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result:</td>
<td>MS with archived schemas which are referenced by the MIP.</td>
</tr>
</tbody>
</table>

1 begin
2 \( S \leftarrow \) The set of schemas referenced by the MIP
3 foreach \( s \in S \) do
4 \( SG \leftarrow \emptyset \)
5 \( v \leftarrow \text{version}(s) \)
6 \( \text{URI}(SG) \leftarrow \text{URI}(s) + v \)
7 if \( \exists MS.\text{select}(\text{URI}(SG)) \) then
8 \( \quad \) continue
9 end
10 \( SG \leftarrow \text{URI}(s).\text{retrieveSchemaVersion}(\text{time}(v)) \)
11 \( MS.\text{insert}(\text{URI}(SG), SG) \)
12 end
13 end

Algorithm 1 will ensure that all schemas which are referenced by the MIP are retrieved and stored in the metadata storage. First, line 3 will determine all referenced schemas. Line 3 will iterate over all these schemas. Line 4 will create an empty schema graph (SG) and line 6 will concatenate the schema name and the version identifier. If the schema version has not already been stored, it is retrieved in line 10. The function \( \text{time}() \) will map a version identifier to the creation time of the ontology. Finally, the schema version is stored in the metadata storage in line 11.

4.2.1.2 Syntactic and Semantic Normalization

To lower the risk of syntactic and semantic heterogeneity, a MIP normalization is executed before the MIP is stored in the metadata storage. Metadata normalization adjusts instances and relations into local terminology. During normalization it has to be ensured that content which is contained in the SIP and which is referencing MIP-metadata still reference the correct metadata. In order to do so, the annotations which contain the metadata references have to be changed accordingly. If, however, the identifiers of the
metadata (e.g., an URI) are not changed, the annotations do not need to be modified. By using annotations, it is not necessary to modify the data.

**Syntactic Metadata Normalization** Since some parts (e.g., preservation metadata) of the metadata is represented by XML and their schemas are modeled with XSD, a syntactic normalization can be applied based on the preference policies of the local archive curator. Since it is assumed that the representation is fixed even in the long-term, the effort to keep archived metadata up-to-date with current technologies will be lowered. The chosen metadata representation language (syntax) has to be supported by the ingest functionality. Different types of syntactic normalization exists:

- **Transcription** The same model but a different language representation is used. Transcription includes serialization and compression. For example, metadata available in RDF/XML can be serialized to RDF/N3 for easier parsing. Also, a special compact language representation can be used to compress RDF to binary RDF [47].

- **Translation** A different model and a different language representation is used. For example, RDF-based metadata is translated to UML.

Syntactic normalization is an important issue and the decision which syntactic normalization is executed has to be taken carefully while respecting possible long-term consequences and the ease of migration. Executing such a syntactic transformation (especially translation) can be tedious or even impossible if the models differ in their expressiveness. However, if the expressiveness of the target model is equal or greater than the source model and once normalization rules from the source model to the target model are available, the normalization can be executed automatically by machines. If such normalization rules are impossible, human intervention is inevitable.

**Semantic Metadata Normalization** The aim of semantic normalization is to make preservation more controllable by the local archive curator. Therefore, dependencies to artefacts outside the archive have to be minimized since a global schema is externally maintained and evolves independently. Performing semantic normalization is to transfer the metadata conforming to a global schema into metadata conforming to a local schema which is under control of the archive curator. Changes to the local schema might be less frequent, occur at well defined points in time and the schema update semantic is always known. The semantic normalization process uses metadata update packages that are maintained by ontology engineers. The alignments of the metadata update packages can be used to create a functionality that translates metadata [43].
4.2 Metadata Harmonization Functionality

4.2.2 Preservation Planning

The goal of preservation planning is to guarantee the archive contents’ accessibility and readability for a designated community by specifying relevant activities which are required to preserve the archived data. For metadata harmonization, preservation planning functionality will automatically monitor metadata schemas with regard to updates.

4.2.2.1 Update Monitoring

In order to keep archived metadata up-to-date with contemporary ontology versions, the preservation system monitors ontology updates and checks if these updates affect the archived metadata. Therefore, the preservation system has to query the interfaces of the ontology engineering environment in order to find out whether or not the updates influence the semantics that have been archived or if only non-archived classes and properties are affected by the ontology update. The semantic preservation system accesses the ontology engineering tool autonomously in order to recognize that an update has occurred. Based on a negotiated policy, the preservation system then decides how to handle the identified updates. One of the following policies might be implemented:

- Identify the updates and send alert to the archive curator that an ontology update has occurred.
- Download and archive updates for each newly available update.
- Download and apply updates autonomously for each newly available update and demand human intervention in case of doubts.
- Identify, download and apply updates on demand during the time of actual archive data reuse.

The processing of retrieved updates with regard to archived semantics is executed by the administration functional entity which is described next.

4.2.3 Administration

Administration include day-to-day activities which are needed to ensure the interpretability of archived metadata. This includes the maintenance of archive and preservation policies. For example, it has to be negotiated which metadata serialization format should be used during ingest and how a syntactic and semantic normalization should take place.
The administration functional entity also coordinates the operations of other functional entities, e.g., it monitors preservation planning. If the preservation planning functional entity detects updates in an ontology, administration is responsible for migrating the archived metadata.

### 4.2.3.1 Metadata Migration

Upon request from the administration entity, metadata migration is the process of transferring archived metadata so that it conforms to a new version of the same ontology. Due to ontology updates which are not backward compatible (e.g., instances or relations might be deleted or moved), migrating metadata can be a complex task that also involves human decisions. As the long-term archive is part of the daily PLM workflow, metadata migration ensures to maintain the ability for applications to reuse the metadata in contemporary environments and to be able to execute queries based on current schema versions.

**Algorithm 2:** Migrate Metadata of an Archive Metadata Storage

Data: $MIP$.

Result: $MS$ with migrated metadata conforming to contemporary schemas.

```
1 begin
2     $S \leftarrow$ The set of schemas referenced by the MIP
3     $\forall s \in S$ do
4         $MUPs \leftarrow URI(s).retrieveUpdates(lastMigration(MIP), now)$
5         $nMIP \leftarrow \emptyset$
6     foreach $u \in MUPs$ do
7         $T \leftarrow MS.select(o, p, u.subjectOfUpdate)$
8         if $u.isBackwardCompatible \vee T = \emptyset \vee u.updateRule = \emptyset$ then
9             $\forall$ triple $\in u$ do
10                $nMIP \leftarrow nMIP \cup$ triple
11         end
12     else
13         $nMIP \leftarrow nMIP \cup MUM.processMUP(u.updateRule)$
14     end
15     end
16     URI(nMIP) $\leftarrow$ generateUniqueSIPName
17     lastMigration(nMIP) $\leftarrow$ now
18     $MS.insert(URI(nMIP), nMIP)$
19 end
```
4.2 Metadata Harmonization Functionality

Algorithm 2 displays the metadata migration process. Line 3 iterates over all schema that are referenced by the MIP which has to be migrated. The execution of line 4 requires a function that is able to determine which ontology changes were made since the time of metadata archival. The algorithm then iterates in line 6 over the retrieved update set. Line 7 determines if the ontology element which has to be updated is used in the local archive metadata storage. If it is not used or if the change is backward compatible (e.g., class insertion) or no update rule exists (line 8), the update is made persistent in the local metadata storage immediately in line 10. Otherwise, the specified update action is executed which might include human intervention in line 13 by special MUM functionality which is described below. Line 17 will generate a new unique identifier for the migrated MIP, line 18 will set the time of last migration, and finally line 19 will insert the migrated MIP into the metadata storage.

4.2.3.2 Handling Effects of Data Migration

Since annotations contain references to unique identifiers of the product data, the process of migrating data might have influences on annotations which reference this data. If product data is migrated and the identifier of a product part which is referenced by an annotation is changed or deleted, the annotation becomes invalid. Since an annotation also references metadata which can be part of the result set of a semantic search, the annotation will get meaningless. To cope with these potential inconsistencies, at least the referenced annotations have to be modified after product data has been migrated. Due to the separation of annotations and metadata it is not required to migrate the metadata. However, as annotations might be deleted, metadata which is not referenced anymore might be deleted as well.

Algorithm 3: Processing Changes Due to Data Migration

Data: \( L = \text{List of deleted product data IDs.} \)
Result: Consistent set of annotations

\begin{verbatim}
begin
foreach \( dId \in L \) do
    foreach \( t \in MS.select(AG, s, hasData, dId) \) do
        MS.delete(AG, s, hasData, dId)
end
end
\end{verbatim}

Algorithm 3 receives a list of unique product data identifiers. In line 2 the algorithm iterates of this list. Then, line 3 checks if these identifiers are used as a refer-
ence in the set of all annotations (Annotation Graph, AG). If so, the annotations are deleted from the metadata storage in line 4 and can not be part of a search result anymore.

4.2.3.3 Query Migration

Important or complex queries might be part of the metadata that is ingested into the archive system. The queries can be stored as literal values or can be encoded with a special vocabulary which is able to represent queries. If queries are part of the metadata set which is migrated, also the archived queries can be migrated. The same mechanism can be used which are used for query mediation (see section 4.2.4.3). The difference between query mediation and query migration is that during mediation, the queries are not transformed permanently.

4.2.4 Access

The access functional entity is a service which enables the designated community to discover and request archived content. Queries are forwarded to the metadata storage which includes descriptive metadata as well as metadata that is used as semantic representation information for ingested content. The access function is also responsible for performing transformations of both data and metadata. These transformation are executed on demand of the consumer and are executed before the result set is delivered for dissemination.

4.2.4.1 Semantic Query Execution

Queries are executed against the metadata storage which was introduced as a functional entity in the harmonization enabled functional model. For querying the metadata storage a query language like SPARQL [117] can be used. According to the annotation data model (section 2.2.2) the result set of the query has to be used to query the set of annotations (AG).

Algorithm 4 displays the execution of a semantic query. First, the query is executed on the metadata storage in line 2. To do so, the Graph $G$ and the query $q$ is used which were provided by the calling function. After querying the metadata storage, the next step is to iterate over the result set (line 3) and to find annotations which reference the subjects found in the result set (line 4). Since the annotations also reference the content, the semantic query identifies the content. As the result set contains both references to
4.2 Metadata Harmonization Functionality

Algorithm 4: Semantic Query Execution Against Annotations

Data: $G \in MS$, Query $q$.
Result: Annotations that reference subjects found in the result of query $q$.

```
1 begin
2    RS = MS.executeQuery($G, q$);
3    foreach $r \in RS$ do
4        $A = MS.select(AG, s, hasData, r.subject)$
5        $R = R \cup A$
6    end
7    return R
8 end
```

the data as well to the metadata, the client has to be able to navigate and visualize the references.

4.2.4.2 Metadata Transformation

During access, metadata transformation is the reverse process that was executed during ingest. The goal of the transformation is to transfer both content and metadata into a state so that it is complete and ready for reuse in a contemporary environment. Transformation might include both a format migration of content as well as migration of metadata. Transformations have to be as automatically as possible. Three different transformation types exist.

**Syntactic Transformation** Upon request, syntactic transformation will change the representation of the metadata. For example, compressed metadata has to be uncompressed or Topic Maps are transformed to RDF [29].

**Cross Schema Transformation** If metadata update packages are available, semantic transformation also supports the transformation of the archived metadata from one domain schema to another.

**Versioned Schema Transformation** The transformation process transfers archived metadata conforming to a specific schema version into metadata conforming to a contemporary schema version requested by the consumer.

In contrast to migration, transformation does not generate a new SIP. The transformation rather generates a DIP which is ready for reuse in a contemporary environment. For the execution of transformation the same functionality which was described for syntactic and semantic normalization can be used. However, during access the archive consumer
might request that metadata is transformed so that it conforms to a particular version of the schema which is referenced by the archived metadata. This will most likely be the case for versioned schema transformations.

### 4.2.4.3 Query Mediation

The functionality of metadata migration will translate the metadata within the archive and will generate new metadata which conforms to another (versioned) schema. The functionality of metadata transformation will generate new metadata for reuse without ingesting it in the archive. During query mediation, instead of the metadata, an incoming query is rewritten so that it conforms to another schema version or to another domain ontology. Unlike migration, query mediation does not generate a new metadata information package which has to be ingested into the archive. The mediation temporarily mediates the query and returns the result set to the calling application. In some cases, also the result set has to be processed for correct interpretation. Figure 4.5 displays different types of query mediation which were identified in this thesis as relevant for product lifecycle processes.

![Figure 4.5: Different Types of Query Mediation](image)

**Horizontal Query Mediation** This kind of query mediation finds metadata based on other equivalent ontology elements of other ontologies. In product lifecycle processes, this kind of mediation allows to use one product ontology (e.g., UNSPCS) to query for metadata which conforms to another product ontology (e.g., eClass).
4.2 Metadata Harmonization Functionality

**Backward Vertical Query Mediation** This kind of query mediation finds archived metadata based on a vocabulary which conforms to the current ontology version. In product lifecycle processes, this kind of query mediation is required since product ontologies evolve frequently.

**Forward Vertical Query Mediation** This kind of query mediation finds current metadata based on a vocabulary which conforms to an archived ontology version. In product lifecycle processes, archived metadata which models product part specifications and which conforms to an old ontology version is used to query contemporary product catalogues for spare parts.

Backward vertical query mediation is a key instrument to maintain the interpretability of metadata and to enable the integration of the long-term archive into daily workflows. Forward query mediation supports the use cases described in the previous chapter when archived specifications are needed to search for contemporary metadata. To implement query mediation, alignments can be used to create a functionality which mediates metadata [43]. To illustrate the idea of query mediation, algorithm 5 displays on a high level how to execute a forward query mediation.

**Algorithm 5: Forward Query Mediation**

```plaintext
Data: Query q, versionId
Result: Mediated Query.
1 begin
2 targetTime ← time(versionId)
3 E ← extractElements(q)
4 foreach e ∈ E do
5 G ← schema(e)
6 P ← URI(G).retrieveProvenance(e)
7 currentProv ← P.oldest
8 while P.hasNext ∧ ¬currentProv.wasValidAt(targetTime) do
9     currentProv ← P.next()
10 end
11 if currentProv.wasValidAt(targetTime) then
12     q.replace(e, currentProv.element)
13 end
14 end
15 return q
16 end
```

Algorithm 5 will receive a query which conforms to an old schema version as well as an identifier which indicates the version to which the query should be mediated. Instead of
the version identifier also a time stamp would be possible as parameter. The algorithm first iterates over all elements (e.g., properties) which were extracted from the query in line 4. Line 5 determines the schema to which the element belongs to. Then, the provenance of that specific elements is retrieved from the schema hosting environment in line 6. In line 7 the oldest provenance record of the element is determined. Line 8 will then iterate over the provenance of the schema element and find the element definition which was valid at the target time. If the element was valid at target time (line 11), the element is replaced in the query in line 12.

4.2.5 Metadata Update Management

The metadata update management functionality is responsible for processing metadata update packages which have been published by ontology engineering applications. The archival functionality will monitor and download the ontology updates and provide these to the update management. The following functionality is relevant for the application of ontology updates.

4.2.5.1 Processing Metadata Update Packages

The update monitoring functionality which is part of the preservation planning entity that was described in section 4.2.2.1 may download and store metadata update packages. These metadata update packages are useless if they are not applied by processing functionality. Based on the product lifecycle processes, this thesis identified three different types of metadata update packages:

**Automatic Update Rules** In this case, the ontology engineer is able to clearly specify a rule which is valid in all dependent archives. For example, every instance of class $A$ will be typed to be an instance of class $B$.

**Case Dependent Update Rules** Here, specific rules are specified by the ontology engineer which inspect the structure and values of the metadata. For example, if the property value is beyond a specific threshold, move instance to class $A$, otherwise to class $B$.

**Archive Local Update Rules** The ontology engineer is not able to specify a rule which can be executed automatically, since it depends on local archive domain knowledge. Therefore, the local archive curator uses the local domain knowledge to specify a automatic or case dependent update rule.
4.2 Metadata Harmonization Functionality

It is necessary to process the retrieved update packages. In the second and third case it is possible that even human decisions are necessary during update processing. Processing of ontology updates is displayed in algorithm 6.

Algorithm 6: Processing Metadata Update Packages

<table>
<thead>
<tr>
<th>Data: ( u \in MUP ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result: ( MS ) with migrated metadata.</td>
</tr>
<tr>
<td>begin</td>
</tr>
<tr>
<td>1 if ( u.updateRule \neq \emptyset ) then</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5 else</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8 mIDs ( \leftarrow ) determineUpdatedMetadataIdentifier(( u.subjectOfUpdate ))</td>
</tr>
<tr>
<td>9 foreach mid ( \in mIDs ) do</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14 end</td>
</tr>
</tbody>
</table>

In line 2, algorithm 6 first checks whether the update package contains automatic rules which specify how to transform metadata stored in the local metadata storage. These update rules were specified by the ontology engineer together with the domain expert who were able to determine automated or case dependent rules. Each of the transformations rules available in an update package is then applied in line 4. If these rules are not available, the local domain expert has to specify rules to migrate the whole local metadata set. This migrating even includes the possibility that is has to be decided for each metadata record how it can be migrated (line 7). Finally (from line 9), updates might have consequences on the annotations and referenced content which then has also to be a migration target to avoid inconsistencies. For example, if metadata is deleted which was identified by a subject URI and which is referenced by an annotation, then the annotation has to be updated as well.
4.2.6 Metadata Storage Functionality

According to the metadata harmonization functionality, the metadata storage has to include the functionality for inserting, deleting, updating, and selecting metadata triples. A triple store like JENA [20] can be used as metadata storage. The implementation of this functionality has to respect the syntactic normalization that was applied during ingest. If a compression of metadata has been executed, the metadata storage has to be able to handle this representation.

4.2.7 Derived Ontology Engineering Requirements

After describing the essential functionality of metadata harmonization, it is necessary to extract the requirements for ontology engineering. These functionality, interface, and update representation requirements are described below.

4.2.7.1 Functionality Requirements

**Release Specific Ontology Versioning** The release planning for ontologies should include the notion of major releases, minor releases, and service packs. The updates that are allowed in such specific releases are different in their consequences and in their frequency. For example, the notion of different compatibility from section 3.1.1.4 can be used.

**Update Semantic Oriented Ontology Engineering** Complex ontology updates like moving a class in a class hierarchy include atomic operations like the deletion of the class and adding a new class. These atomic operations should be aggregated into one transaction whose semantic is easy to understand by consumers of ontology updates. The usage of update hierarchies will be useful [6].

**On-the-fly Alignment Generation** If a new version of an ontology is derived, all existing alignments can be reused for the new version. If the update semantic is known, new alignments can also be created automatically.
4.2.7.2 Interface Requirements

As the metadata harmonization functionality retrieves metadata update packages, an ontology engineering tool has to provide the following programming interface.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Parameter</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>retrieveUpdateFeed</td>
<td>Time ( t_1 ), Time ( t_2 )</td>
<td>Retrieve list of ontology updates between ( t_1 ) and ( t_2 )</td>
</tr>
<tr>
<td>retrieveUpdates</td>
<td>Time ( t_1 ), Time ( t_2 )</td>
<td>Retrieve metadata update packages between ( t_1 ) and ( t_2 )</td>
</tr>
<tr>
<td>retrieveProvenance</td>
<td>URI element</td>
<td>Retrieve provenance of a specific ontology element</td>
</tr>
<tr>
<td>retrieveOntology</td>
<td>Time ( t )</td>
<td>Retrieve ontology snapshot of a specific time</td>
</tr>
</tbody>
</table>

4.2.7.3 Update Representation Requirements

The previous sections also required that an update made to an ontology is represented by the following parts of a metadata update package.

<table>
<thead>
<tr>
<th>Element</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>subjectOfUpdate</td>
<td>The ontology element identifier which has been modified.</td>
</tr>
<tr>
<td>isBackwardCompatible</td>
<td>Indicator, if the update is backward compatible or not.</td>
</tr>
<tr>
<td>updateRule</td>
<td>Rules for dependent systems how to cope with an update.</td>
</tr>
</tbody>
</table>

The next section presents a conceptual representation model that allows to implement a corresponding information model which can support the interface, functionality, and representation requirements that have been outlined by the system model so far. Furthermore, corresponding operations that will be the basis for processing this information model in a later implementation will be derived, too.

4.3 Ontology Engineering for Metadata Harmonization

According to the system landscape which was described in section 4.1, the metadata harmonization functionality which was presented in the previous section 4.2 depends on special characteristics of the ontology engineering environment. In such an environment,
an ontology engineering tool manages the lifecycle of an ontology. Its main task is to support the ontology engineer. In addition, the tool provides the service interfaces which were described above and can be accessed by machine and human clients. One of the machine clients is a long-term preservation system which archives semantics expressed as ontology-based metadata. Based on the functionality, interface, and update representation requirements derived in the previous section, this section develops a model which allows to author, document and maintain ontology updates and the provenance of domain ontologies.

4.3.1 Domain Knowledge Provenance Graphs

As the world is changing, the data that describes relevant domains of interest has to change, too. Since an ontology is a specification of a real world conceptualization, classes and properties at schema level as well as instances and relationships at data level evolve over time. In order to preserve metadata which reference these evolving ontologies, the history of ontology elements (e.g., classes, properties, instances) has to be recorded so that their provenance (e.g., How has a class evolved over time?, What was the state of the class at time $t$?) can be queried and exploited for archival functionalities.

A Domain Knowledge Provenance Graph (DKPG) is a conceptual model to express such kind of ontology history and follows the ideas presented in [93], [86], and [49] but extends these approaches to the ontology schema level. A DKPG is a named graph containing both schema and instance elements of a particular domain. A set of DKPGs is called a Domain Knowledge Base (DKB). Since a triple store holds metadata conforming to a DKPG-based ontology, it is dependent on the updates made to a DKPG. A metadata storage which is part of a long-term preservation system is such a dependent triple store. A triple $t$ within a DKPGs can be annotated with a value $\lambda$ from a specific domain $D$, $(s,p,o) : \lambda^D$. Examples for annotation domains are time and provenance which are described in more detail below.

**Time Annotation Domain** In the time domain $T$, an annotated triple $(s, p, o) : [t_1, t_2]^T$ would indicate that the triple is valid between $t_1$ and $t_2$. The special symbol now indicates that the triple is valid at current access time.

**Provenance Annotation Domain** A triple annotated with the provenance domain $P$ is expressed as $(s, p, o) : [p_0]^P$. A single provenance annotation $p_0$ holds the update type (e.g., deletion, addition), reason, author, date, previous provenance and following provenance as well as the corresponding metadata update package.
4.3 Ontology Engineering for Metadata Harmonization

Figure 4.6: A Simple Ontology

For the illustration of a DKPG, figure 4.6 displays a simple ontology in textual representation which uses RDFS as schema definition language.

The ontology defines a single named graph called SimpleOntology and two classes A and B and two properties p and q. The property p is a subproperty of q. The domain of this property p is class B and values for this property can contain literal values. Also, an instance i is defined which contains a relation of property p with value v.

RDF-based ontologies also have a graphical representation. The following sections will use these symbols to represent an ontology:

<table>
<thead>
<tr>
<th>Element</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="rdfs:Class" /></td>
<td>Ellipses depict elements of the schema definition language.</td>
</tr>
<tr>
<td><img src="image" alt="A" /></td>
<td>Classes are depicted as rectangles.</td>
</tr>
<tr>
<td><img src="image" alt="p" /></td>
<td>Properties are shown as diamonds.</td>
</tr>
<tr>
<td><img src="image" alt="a" /> <img src="image" alt="r" /> <img src="image" alt="b" /></td>
<td>Relations are depicted as arcs between nodes.</td>
</tr>
<tr>
<td><img src="image" alt="i" /></td>
<td>Instances are depicted as circles.</td>
</tr>
<tr>
<td><img src="image" alt="v" /></td>
<td>Literal values are depicted as stars.</td>
</tr>
</tbody>
</table>

When using these symbols, the ontology defined in figure 4.6 can graphically be represented as shown in figure 4.7.
4.3.2 Tracking Ontology Evolution with DKPGs

A DKPG can be used to track updates and thus keep the history of a domain ontology. In the following sections, several simple update patterns using the RDF schema definition language demonstrate how a DKPG can be used to record the history of a domain ontology. Basic ontology updates can be modeled with the manipulation of only one triple. Since triples can only be inserted or deleted, updates are considered as the sequence of triple deletion and triple insertion. As the insertion of ontology elements (both on schema and instance level) does not have effects on archived metadata, the following delete operations are considered as simple examples of relevant ontology evolution.

<table>
<thead>
<tr>
<th>Deletion</th>
<th>Level</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triple</td>
<td>Data</td>
<td>Triples which were referencing the subject become invalid.</td>
</tr>
<tr>
<td>Relation</td>
<td>Data</td>
<td>Queries might become invalid.</td>
</tr>
<tr>
<td>Instance</td>
<td>Data</td>
<td>Triples which reference the instance as object become invalid.</td>
</tr>
<tr>
<td>Property</td>
<td>Schema</td>
<td>All relations at data level become invalid.</td>
</tr>
<tr>
<td>Class</td>
<td>Schema</td>
<td>All instances of type class become invalid.</td>
</tr>
</tbody>
</table>
4.3.2.1 Delete Triple

First, basic functionality to mark a triple as deprecated is required to maintain ontology evolution in $DKPGs$. Algorithm 7 which is outlined below receives an identifier $G$ for a $DKPG$ stored in a DKB as well as a triple including identifiers for subject, predicate, and object. In addition, a provenance record $p_1$ (e.g., update rationale) has to be provided by the calling function.

Algorithm 7: Delete Triple

<table>
<thead>
<tr>
<th>Data:</th>
<th>$G \in DKB; s, p, o, p_1 \in U$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result:</td>
<td>Updated $DKB$.</td>
</tr>
</tbody>
</table>

1. begin
2. $t_1 \leftarrow currentTime$
3. $DKB.select(G, s, p, o) : [t_0, now]^T, [p_0]^P$
4. linkProvenance($p_0, p_1$)
5. $DKB.delete(G, s, p, o) : [t_0, now]^T, [p_0]^P$
6. $DKB.insert(G, s, p, o) : [t_0, t_1]^T, [p_0]^P, [p_1]^P$
7. end

In line 2, the function $currentTime$ will determine the time of execution and store it in $t_1$. Line 3 will retrieve the values $t_0$ and $p_0$ in order to enable line 4 to link the provenance records $p_0$ and $p_1$ with each other. The provenance record $p_1$ will be the successor of $p_0$. In addition, $p_0$ will be the predecessor of $p_1$. In line 5, the triple is deleted. Finally, in line 6, a new triple with time annotation $t_0, t_1$ (indicating that the triple will not be valid anymore) and provenance $p_0$ and $p_1$ will be inserted.

4.3.2.2 Delete Relation

A property which is defined at the schema level of an ontology might be used as relation at the data level of an ontology.

Figure 4.8: Deleting Relation $p$ at Time $t_1$ with Provenance $p_1$
Figure 4.9(a) displays a triple \((s, p, o)\) at time \(t_0 + 1\). The triple is annotated with \([t_0, now]\) from the time annotation domain indicating that the triple is currently valid. The triple is also annotated with provenance \(p_0\) (e.g., time and reason of triple creation). Figure 4.9(b)) displays the triple at time \(t_1 + 1\) after the delete relation operation has marked the relation as deprecated by changing the time annotation to \([t_0, t_1]\) and adding provenance \(p_1\). To execute the delete relation operation, algorithm 7 can be used.

### 4.3.2.3 Delete Instance

As a domain ontology may also contain instances on the data level (e.g., an instance \(\text{Germany}\) of class \(\text{Country}\)), updates executed on instances (e.g., updates regarding unification of \(\text{germany}\)) have also to be observed.

![Ontology Diagrams](image)

(a) Ontology at \(t_0 + 1\)  
(b) Ontology at \(t_1 + 1, t_1 > t_0\)

Figure 4.9: Deleting Instance \(s\) at Time \(t_1\) with Provenance \(p_1\)

In figure 4.9, the instance \(s\) which is of type \(C\) is deleted at time \(t_1\) with provenance \(p_1\). The instance is not physically deleted but it is logically deleted by changing the time annotation. The delete instance update might not have large impacts on the domain ontology itself. The valid time of all referencing objects in the same domain knowledge base have to be updated. However, deleting an instance might have larger impacts on a metadata storage which is referencing the instance. If a new ontology version is released for public consumption, the instance identifier (e.g., an URI) is not valid anymore. For dependent knowledge bases, update rules have to be formulated.

Algorithm 8 will execute the delete instance operation. The algorithm will receive the subject to be deleted \(s\), the provenance record \(p_1\) and also a \(\text{DKPGG}\) which is maintained in the DKB. Line 2 will iterate over all type instantiation definitions which are currently valid. Line 3 will update the time annotations of the triple which defined the subject. Finally, line 5 will iterate over all elements of the same domain ontology which were referencing \(s\) and will update the time annotation in line 6.
Algorithm 8: Delete Instance

Data: \( G \in DKB; s, p_1 \in U \).
Result: Updated \( DKB \).

1 begin
2 \hspace{1em} foreach \( \text{triple} \in DKB.\text{select}(G, s, \text{type}, cl) : [t_0, now]^T, [p_0]^P \) do
3 \hspace{2em} deleteTriple\((G, s, \text{type}, cl, p_1)\)
4 end
5 \hspace{1em} foreach \( \text{triple} \in DKB.\text{select}(g, o, p, s) : [t_0, now]^T, [p_0]^P \) do
6 \hspace{2em} deleteTriple\((g, o, p, s, p_1)\)
7 end
8 end

4.3.2.4 Delete Property

The delete property operation will mark a property which is in the domain of a specific class as deprecated. The operation will also deprecate the range definition of the property. In addition, all relations on the instance level which are valid at update time \( t_1 \) are deleted. Figure 4.10 displays the effect of the delete property operation.

At time \( t_0 \), the property \( p \) references as domain the class \( A \) and as range the class \( B \). At time \( t_1 \), this property will be deleted by updating the time annotation to be valid until \( t_1 \) and by annotating the provenance \( p_1 \).

Algorithm 9 displays the execution of the delete property operation and will receive the reference \( s \) for the property and a provenance record \( p_1 \). Line 2 will deprecate the type definition. From line 3 to 6 the domain and range definitions are selected and marked as deprecated. Finally, line 7 and 8 will deprecate every relation which has been defined in the same \( DKPG \) using the delete triple operation. For simplicity, it is assumed that the property has no super and no sub properties. A more sophisticated algorithm has to examine the property hierarchy carefully.

Figure 4.10: Deleting Property \( p \) at Time \( t_1 \) with Provenance \( p_1 \)

Algorithm 9 displays the execution of the delete property operation and will receive the reference \( s \) for the property and a provenance record \( p_1 \). Line 2 will deprecate the type definition. From line 3 to 6 the domain and range definitions are selected and marked as deprecated. Finally, line 7 and 8 will deprecate every relation which has been defined in the same \( DKPG \) using the delete triple operation. For simplicity, it is assumed that the property has no super and no sub properties. A more sophisticated algorithm has to examine the property hierarchy carefully.
Algorithm 9: Delete Property

Input: \( G \in DKB; \ s, p_1 \in U \).
Output: Updated DKB.

1 begin
2 \( \text{deleteTriple}(G, s, \text{type}, \text{rdf:Property}, p_1) \)
3 \( \text{DKB.select}(G, s, d, o) : [t_0, \text{now}]^T, [p_0]^P \)
4 \( \text{deleteTriple}(G, s, d, o, p_1) \)
5 \( \text{DKB.select}(G, s, r, o) : [t_0, \text{now}]^T, [p_0]^P \)
6 \( \text{deleteTriple}(G, s, r, o, p_1) \)
7 foreach \( \text{triple} \in \text{DKB.select}(G, i, s, o) : [t_0, \text{now}]^T, [p_0]^P \) do
8 \( \text{deleteTriple}(G, i, s, o, p_1) \)
9 end
10 end

4.3.2.5 Delete Class

Figure 4.11 displays the effects of the delete class operation in a DPKG.

![Ontology at t_0 + 1](a) Ontology at t_0 + 1

![Ontology at t_1 + 1, t_1 > t_0](b) Ontology at t_1 + 1, t_1 > t_0

Figure 4.11: Deleting Class \( C_2 \) at Time \( t_1 \) with Provenance \( p_1 \)

When deleting a class in a DPKG, different alternatives approaches with regard to changing the class hierarchy and properties are possible:

1. Deleting an inner class could induce the deletion of all of its sub-classes from the class hierarchy. Alternatively, the sub-class relation of direct sub-classes of the class which is to be deleted is moved to the upper-class.

2. Properties of the class to be deleted could be moved to a super class which would require that a single super-class exists. Alternatively, all of the properties of the deleted class and all triples which reference the class are deleted.
Figure 4.11(a) displays a class hierarchy at time \( t_0 \). Class \( C_3 \) is a sub-class of \( C_2 \) and \( C_2 \) is a sub-class of \( C_1 \). In addition, an instance \( s \) of type \( C_2 \) is defined. For clarity, the type definition for class \( C_3 \) is not shown since this definition keeps valid. Also for clarity, possible domain or range definitions are not shown. Figure 4.11(b) displays how the class \( C_2 \) is deleted which is located in the middle of a class hierarchy.

The delete class operation deletes a single leaf class or a single inner class from the class hierarchy, changes the sub-class relation of direct sub-classes, and deletes the properties. However, if the class which should be deleted is the root of the class hierarchy, it can not be decided without human knowledge which class should be the new root of the class hierarchy. Therefore, this operation can not be automated and is not studied further.

**Algorithm 10: Delete Class at Schema Level**

**Input:** \( G \in DKB; s, p_1 \in U \).

**Output:** Updated DKB.

1. \( t_1 \leftarrow \text{currentTime} \)
2. \( \text{deleteTriple}(G, s, \text{type}, \text{rdfs:Class}, p_1) \)
3. \( UC \leftarrow DKB.\text{select}(G, s, sc, o) : [t_0, now]^T, [p_0]^P \)
4. \( \text{deleteTriple}(G, s, sc, o, p_1) \)
5. \( SC \leftarrow DKB.\text{select}(G, o, sc, s) : [t_0, now]^T, [p_0]^P \)
6. \( \text{deleteTriple}(G, o, sc, s, p_1) \)
7. \( \text{if } SC \neq \emptyset \text{ then} \)
8. \( \quad \text{if } UC \neq \emptyset \text{ then} \)
9. \( \quad \quad DKB.\text{insert}(G, SC, sc, UC) : [t_1, now]^T, [p_1]^P \)
10. \( \quad \text{else} \)
11. \( \quad \text{end} \)
12. \( \quad \text{end} \)
13. \( \text{end} \)
14. \( \text{end} \)

Algorithm 10 will execute the delete class operation. It will receive the reference \( s \) to the class which has to be deleted, a provenance record \( p_1 \) and also a \( DKPG \) \( G \) which is maintained in the DKB. Line 3 will delete the type definition. Line 4 will select all upper classes and will mark them as deprecated in line 5. Line 6 will select all sub classes and will mark them as deprecated in line 7. From line 8 to 12, the class hierarchy is changed. If the class to be deleted does not have any subclasses (line 8), the class hierarchy does not need to be changed. If the class to be deleted has subclasses as well as an upper class (line 9) the hierarchy is changed accordingly. If there exists no upper class, the sub class relation is deleted in line 12.
Algorithm 11: Delete Class at Data Level

**Input:** $G \in DKB; s, p_1 \in U$.

**Output:** Updated DKB.

```
begin
  for each triple $\in DKB.select(G, o, p, s) : [t_0, now]^T, [p_0]^P$ do
    if $p = \text{rdf.type}$ then
      $DKB.deleteInstance(G, o, p_1)$
    else
      if $p = \text{rdfs.d} \lor p = \text{rdfs.r}$ then
        $DKB.deleteProperty(G, p, p_1)$
      else
        deleteTriple$(G, o, p, s, p_1)$
      end
    end
  end
end
```

After rearranging the class hierarchy, also the $DKPG$ at data level has to be updated. In line 2, algorithm 11 iterates over all triples which use the class $s$ as object. In line 3, it is verified whether the relation of this triple is a type definition. If it is a type definition, the instance is deleted in line 4 by using algorithm 8. If the relation is a domain or range definition the property is deleted in line 7 by using the algorithm 9. Otherwise, line 9 marks the triple as deprecated for each other case by using algorithm 7.

### 4.3.3 Querying and Accessing DKPGs

According to section 4.2.7.2, several requirements for service interfaces exists which have to be provided by an ontology engineering tool that supports the long-term harmonization of dependent metadata. The required parameters for these interfaces have already been provided. Below, algorithms for possible realization based on $DKPGs$ are provided.

#### 4.3.3.1 Ontology Snapshot

A $DKPG$ can be used to query a domain ontology at a specific point in time. The time annotations and the vocabulary of a schema definition language can be used to retrieve the valid ontology elements. By using $t = now$ the ontology can be retrieved which is
valid at current access time. Since sometimes only a version identifier is known, it is assumed that a function exists which maps version identifiers to time. This mapping function can be used to execute algorithm 12 which retrieves an ontology at a specific time.

Algorithm 12: Retrieve Ontology Snapshot

\[
\begin{align*}
\text{Input: } & G \in DKB; \ t \in [t_0, t]^T \\
\text{Output: } & \text{Ontology at Time } t \\
\text{1 begin} \\
\text{2 } & S \leftarrow \emptyset \\
\text{3 } & \text{foreach } p \in RDFS \text{ do} \\
\text{4 } & \quad S \leftarrow S \cup DKB.select(G, s, p, o) : [t_0, t]^T \\
\text{5 } & \text{end} \\
\text{6 } & I \leftarrow \emptyset \\
\text{7 } & \text{foreach } (\text{element}, p, o) \in S \text{ do} \\
\text{8 } & \quad I \leftarrow I \cup DKB.select(G, s, p, \text{element}) : [t_0, t]^T \\
\text{9 } & \text{end} \\
\text{10 } & \text{return } S \cup I \\
\text{11 end}
\end{align*}
\]

Line 3 of algorithm 12 will iterate over the RDFS schema definition vocabulary. Then, in line 4 all triples which are valid at time \( t \) are selected. As an ontology also contains instances, line 7 will use all subjects found on the schema level to select instances which are valid at time \( t \) in line 8.

4.3.3.2 Ontology Element Provenance

Handling terminology evolution requires the knowledge about the provenance of the ontology elements. This provenance can be exploited during query mediation by rewriting the properties of queries. Depending on the direction (forward or backward mediation) the query has to be rewritten with older or younger elements. Algorithm 13 displays how to retrieve the provenance of a specific ontology element.

Algorithm 13 first determines all existing properties of the ontology element in line 3. It then iterates in line 4 over all properties and selects the captured provenance records of each property in line 5.
Algorithm 13: Retrieve Ontology Element Provenance

Input: $G \in DKB$, $e \in U$
Output: Provenance of Ontology Element $e$

begin
  $P \leftarrow \emptyset$

  $PROPS \leftarrow DKB.$select($G, e, \text{distinct } p, o$)

  foreach $p \in PROPS$ do
    foreach $t \in DKB.$select($G, e, p, o : [t_1, t_2]^T, [p_0]^P$) order by $p_0$.time do
      $P \leftarrow P \cup p_0$
    end
  end

  return $P$
end

4.3.3.3 Metadata Update Representation

As described in section 4.2.7.3, metadata harmonization functionality requires a special representation of the update packages. These update packages are authored with the support of ontology engineering tools and are communicated between different systems.

Figure 4.12: Operational Metadata Update Package

Metadata update packages which are depicted in figure 4.12 are the conceptual model for this update exchange. A metadata update package consists out of an alignment as well as a transformation rule and are referenced by an update feed. These three constituents are now described in more detail.

Alignment The first part of update packages are ontology alignments [38] which describe the relationships between ontology elements. Whereas ontology alignments originally described the relation of ontology elements which stem from two different ontologies, this thesis also regards alignments for ontology elements of different versions of the same ontology. This thesis continues in the fashion of [56] by introducing alignments
4.3 Ontology Engineering for Metadata Harmonization

on different model layers which are needed for functionality in metadata harmonization. Figure 4.13 displays the Meta-Object Facility (MOF) [105] which is a standard for model-driven engineering.

The architecture contains four different layers. Since the M3 Meta Meta Model layer is not applied in this thesis, it is not shown in figure 4.13. The M2 Meta Model layer contains the necessary elements (class, property) for the definition of ontologies. The figure displays the RDFS [12] and UML [107] schema vocabularies for the definition of ontologies. The M1 Schema layer contains the structure of an ontology. On this layer, the elements from the upper layer are used to define a schema. The bottom M0 Data layer contains the instantiations of the ontology.

Alignments can be defined on each of the layers. Bidirectional dotted arrows visualize similarity on the schema (ls:isSimilar) and model (lm:isSimilar) level, whereas on the data level it visualizes linked data identity (owl:sameAs). The namespaces for similarity (ls: and lm:) are fictional and currently lack a specific vocabulary definition. Linked
schema maps classes and properties which are semantically similar. Linked schema can be used to map between versions of schemas and to map between different domain schemas. As the data is an instance of a schema, a schema is an instance of a model [56]. In consequence, a schema element could also be ready for dereferencing to provide a provenance of queried metadata schema elements (e.g., the history of a schema class or property).

Usually, alignments between two ontologies are maintained by an independent ontology engineer. As ontologies evolve, the alignments are also affected. For example, if a class is deleted which was referenced in an alignment, the alignment will not be valid anymore. Alignment themselves can use update packages to keep the alignment definitions up to date. Alignment maintenance will be easier, if the alignments are authored within the same tool which maintains a domain ontology. This approach has the advantage that during evolution of ontologies, alignments can be updated semi-automatically when a new version of an ontology is created. In addition, the ontology engineering tool can detect alignment inconsistencies during ontology updates (e.g., deleting a property which is referenced by an alignment).

### Transformation Rule

After identifying the ontology elements which are affected by the ontology update, it is necessary to describe the modifications which are needed in order to comply with a new ontology version. This is known best by the ontology engineer whereas a dependent ontology user can only guess or try to derive differences by comparison of ontology versions. The ontology engineer therefore has to define operational transformation rules which are valid for all dependent applications. If the ontology engineer is not able to clearly specify a transformation rule, then he has to be able to mark the update accordingly so that during update processing human intervention is supported. As a third possibility an archive curator can specify a transformation rule which preserves the metadata in a way which is appropriate for the local archive.

### Ontology Update Feed

To recognize that an ontology has changed, an ontology engineering tool which hosts one or more domain ontologies has to provide a public update propagation feed. This feed can be based on web standards like RSS [121] or Atom [102] so that it can be consumed both by computers and humans. Such an update feed would describe the changes that have to be made to the ontology and it would provide references to the metadata update packages which contain transformation rules that are to be applied for incorporating the ontology update into archived semantics. Individual ontology updates can then be retrieved and processed by dependent applications.
Algorithm 14: Retrieve Ontology Update Feed

**Input:** $G \in DKB; s \in U; t_0, t_1 \in T$

**Output:** Ontology Update Feed Between Time $t_0$ and $t_1$

1. $F \leftarrow \emptyset$
2. **foreach** $t \in DKB.select(G, s, p, o) : [t_0, now]^T, [p_0]^P$ **order by** $p_0.time$ **do**
3.   **if** $p_0.time \leq t_1$ **then**
4.     $F \leftarrow F \cup p_0.description$
5. **end**
6. **end**
7. **return** $F$

For retrieving metadata update packages of a specific ontology element or of the whole ontology which were authored between time two points in time, algorithm 14 can be used. Instead of $p_0.description$ in line 5 the metadata update package has to be used.
5 Implementing Metadata Preservation

The last chapter described a system model of relevant metadata harmonization archival functionality and supporting ontology engineering tool functionality which is required for metadata harmonization. Both components are combined and interact to enable semi-automatic metadata harmonization. This chapter provides a high-level overview of implemented software tools along the whole metadata preservation lifecycle in the context of product lifecycle processes.

5.1 PLM Metadata Preservation Tool Chain

Tools along the metadata preservation lifecycle are needed for metadata creation, evolution, and reuse. First, an overview of the interaction of PLM tools along the metadata lifecycle is presented. Then, the functionality of the tools is described in more detail.

5.1.1 Introduction and Overview

Figure 5.1 displays how different product lifecycle actors (e.g., product designer, service personell, supplier) use the execute PLM tool functionality to execute processes of the various product lifecycle phases. The tools will use the different phases of the metadata lifecycle to exploit the core harmonization functionality. The tool functionality for each of the metadata lifecycle phase will be described in more detail in the following sections:

- **Annotation** Special functionality will annotate metadata either automatically or manually to the product data. The relevant annotation tools are described in section 5.1.2.

- **Archival** The metadata lifecycle continues when a repository administrator will archive the annotated product data at specific points in time with the help of archival functionality that is integrated in PLM workflows. How the archival functionality collect, aggregate, and ingest the relevant metadata is described in section 5.1.3.
Implementing Metadata Preservation

• **Evolution** A domain ontology engineer is responsible for authoring updates to ontologies. During this evolution, a dedicated ontology engineering tool which is described in section 5.1.4 collects operational metadata update packages. The updates packages can be pushed by the ontology engineering tool to the archive or they can be pulled by the archive automatically or manually by the archive curator. Upon request of administration, metadata is migrated within the archive.

• **Exploration** Product lifecycle actors are able to explore archived product data and metadata by using special semantic exploration tools which are described in section 5.1.5. These tools allow to browse domain ontologies and to execute queries which might be mediated because contemporary ontologies have evolved.
- **Reuse** The archived metadata can be transformed during access of an archive consumer so that the metadata conforms to contemporary ontologies (section 5.1.6). The transformation can be regarded as the creation of new metadata and the lifecycle starts from the beginning.

### 5.1.1.1 PLM Metadata Preservation System Architecture

After presenting an overview of the tool chain, this section fits these tools in a semantic digital archive system architecture which consists of three layers:

[Diagram of Semantic Digital Archive System Architecture]

The **tool layer** contains the workflow tools (e.g., CAD design software) and a special data explorer tool that allows accessing the repository and the archive via browsing of metadata schemas. Finally, the evolution tool allows editing the metadata and the associated schemas. While editing the metadata, alignments are maintained semi-automatically.

The **active (meta)data repository layer** contains the data repository and a triple store such as [20] which holds the metadata. While the workflow tools interact with the data repository, the data explorer is able to query both the repository as well as the metadata repository because the metadata references the active data repository via annotations. By querying and discovering metadata, product data can be explored.

The **archive layer** at the bottom contains the long-term archive functionality. The data from the active repository and the metadata is ingested into the long-term archive on demand when specific points in time of the business workflow are reached. The long-term archive also contains an access and query service that allows the data explorer to
access the archived metadata. Finally, an update service is able to accept operational updates from the metadata triple store.

5.1.2 Capturing and Annotation

The first step in the long-term preservation of metadata is to capture metadata. Sometimes it is necessary to capture the metadata in real-time because it can not be recreated anymore. Below, one example for manual metadata creation and one example for automatic metadata extraction is described.

5.1.2.1 Manually

Manual metadata capturing has been implemented for the PLM tool ARAS Innovator [5]. Figure 5.3 displays a hierarchical product design structure where each row represents a single product part that can be inspected by double clicking.

![Figure 5.3: Product Design Data](image)

When a user double clicks a product part, a new window will open that displays the characteristics of the product part. As can be seen from figure 5.4, the standard functionality of this window has been extended [145]. These extensions enable to create one or more semantic annotation for each product part. At the bottom, figure 5.4 displays a new sub window that contains the existing annotations. The annotations can reference metadata
from different ontologies. The existing annotations can be edited and deleted by clicking the symbols on the right hand side of each annotation.

By clicking the plus sign, a new window will open that enables to create a new annotation. Figure 5.5 displays this window that enables a product lifecycle actor to annotate the selected product design part.

The actor first selects a domain ontology whose elements should be used for annotation. Then, the appropriate ontology element (e.g., predicate, type, instance) is selected which is then used as a metadata annotation of a product data part.
5.1.2.2 Automatically

The example for automatic metadata capturing in the realm of product lifecycle data is taken from the execution of tests for manufactured engines [129]. These tests runs are performed to control the function of engines. The data which have been generated during these tests has to be collected in order to document that test which are required by law were executed. Therefore, the test data (e.g., model parameter) has to be annotated with metadata (e.g., date of test run, engine number) and then archived for future reuse (e.g., during accident investigation).

Figure 5.6: Automatic Metadata Extraction

Figure 5.6 displays a screenshot of a tool that enables to automatically extract metadata from test runs. At the top left corner the user will enter a unique identifier for the engine which was tested (e.g., serial engine number). This identifier is used as a name for the generated container. Below of this field, the directory is specified which contains the generated test run files and which should be scanned. After pressing the start button,
the tool inspects the generated files from a test run, extracts relevant metadata, annotate the data with metadata and generate a SIP for ingestion in the specified output folder. A test run consist of a inspection report that contains the measurements for pressure and temperature. Since the inspection report is available as a PDF document, special functionality is needed to extract the metadata. The execution of individual process steps are indicated by marking the steps as done on the right hand side of the tool. More processing messages are shown at the bottom left corner. The user is also able to enter individual metadata as name and value pairs.

### 5.1.3 Archival

After annotating a product model with metadata, it has to be archived for reasons explained in section 2.3.6. The archival workflow execution in a PLM environment which includes long-term preservation functionality is depicted in figure 5.7.

![Figure 5.7: Archival Workflow within a PLM Environment](image)

Figure 5.7: Archival Workflow within a PLM Environment
At the top level it displays the leading PLM system which stores the generated product data in a PDM repository. The PLM system is extended with archival functionality that is able to connect to external systems (ES) which maintain domain specific metadata (e.g., ideation metadata, design rationale, product classifications). Each external system might have its own repository and might access the archive. The bottom layer displays the long-term archive and dedicated access interfaces whereas the right side displays the annotated product data prepared for ingestion.

5.1.3.1 Archival Workflow

Figure 5.7 includes the following workflow steps:

1. **PLM Workflow execution.** A leading PDM system executes a workflow for product realization including phases like requirements engineering, design, etc. In addition, the PLM system stores the product lifecycle data in the PDM repository.

2. **Archival triggering.** The PLM workflow execution triggers the archival functionality. This archival can be triggered automatically at special point in time (end of life, release for production) or manually as shown in figure 5.8.

3. **Reference external systems.** The annotated product data contains references to external systems which enable the creation of product relevant metadata.

4. **Product data collection.** The archival functionality iterates over all connected external references which implement a special service interface in order to collect
archival relevant data and metadata using a well defined interface that has to be implemented by external systems.

5. **Product data aggregation.** After collecting the product data from the PDM repository and from the external systems, the product data is dumped into a hierarchical structure in the file systems.

6. **Product data ingestion.** The whole product data and metadata collection is ingested into a long-term archive.

7. **Store archived product data reference.** A unique archive identifier is created by the archive after successful ingestion and stored in the PDM repository. After archival, the product data might be deleted from the PDM repository.

8. **Query and access archived product data.** By using the long-term archive access interfaces, archived product data and metadata can be queried and accessed from the PLM workflow.

The long-term archive remains an active ingredient in daily PLM workflows. Therefore, the metadata that is used to query the archive have to be kept interpretable under knowledge evolution. This is even more important as the product data might be deleted from the active repository after it has been ingested into the long-term archive. The following evolution phase of the metadata preservation lifecycle which is described in the next section will make it harder to keep the archived data interpretable.

### 5.1.4 Evolution

After the annotated product data and metadata has been archived, ontologies will evolve. The evolution phase is supported by the EVO (*Evolving Ontologies*) software application which allows to trace updates made to ontologies. This section provides an overview of the tool functionality.

#### 5.1.4.1 The EVO Software Tool

The software application EVO provides functionality to create ontologies and to keep track of ontology elements (schema and instances) during ontology evolution. During evolution of instances and schemas, alignments are generated which can be used to preserve metadata. EVO allows semi-automatic generation of schema and instance alignment when a new version of a schema is generated. Since the alignments are stored in the same named graph as the schema and the instances and are described by a dedicated vocabulary, they are operational and can be exploited during preservation. In addition,
the tool allows the detection of alignment inconsistencies during editing schema updates and it allows capturing the rationale and the provenance of schema updates. The main functionality of the tool is now described in more detail.

- **Ontology Management** This functionality allows to create a new ontology which is identified by a name and a namespace. This ontology can then be edited by creating and modifying schema elements. During each ontology update, the update rationale is captured. It is also possible to define ontology instances (a dataset) which conforms to the currently loaded ontology. The tool therefore can be used for creating data that becomes metadata when it is being annotated. During edition, the tool tracks changes made to ontologies and instances. In addition, existing ontology alignments are created on the fly and are kept up-to-date.

- **Querying Functionality** This functionality allows to query the local metadata storage or the long-term archive or both of them. The user will edit and execute SPARQL queries which are stored for later reuse. The tool mediates between new and old ontology versions during query execution on archived knowledge.

- **Provenance Information** This functionality enables to display the schema or the instances of a specific ontology. In addition, the visualization of schema elements updates and instances is possible by using a timeline widget (figure 5.9). Green circles visualize the creation of ontology elements, blue mark a modification, and red circle indicate the deletion.

![Figure 5.9: Visualization of Ontology Element Provenance](image)

- **Alignment Management** This functionality allows the creation of alignments for ontologies and the migration of queries that are stored in the system.
• **Archival Functionality** This functionality allows to ingest an ontology into a long-term archive. The ontology will be deleted from the local triple store but can be retrieved from the archive later on.

### 5.1.5 Exploration

When a product lifecycle actor seeks data from an archive and lacks a priori knowledge of the contents of that archive, ontology based exploration helps to understand and discover the searched content. This is different from the search use case where a fixed set of description information has been annotated.

![Ontology Based Search](image)

Figure 5.10: Ontology Based Search of Product Lifecycle Data

Figure 5.10 displays an example of a possible archive exploration tool. At the top, the actor is able to select the ontologies that have been used to annotate the product data. After loading the ontology, the classes and properties can be browsed at the left of the window. After selecting the relevant classes and properties which should be used for querying, it is possible to execute the query. The query is submitted both to the active repository and the long-term archive. The query will first find the metadata and then
Implementing Metadata Preservation

the annotations which are by the metadata. These annotations the lead the path for the relevant annotated product data.

Another special semantic exploration tool has been implemented which allows to search for ontology based metadata (figure 5.11).

![Semantic Exploration of Archived Product Data](image.png)

Figure 5.11: Semantic Exploration of Archived Product Data

This tool displays on the left-hand side a domain ontology as a two dimensional graph. The ontology has been loaded from the archive or from the repository. By selecting a class from the ontology, its properties are displayed on the right-hand side. For each of the properties, concrete values have to be entered for filtering. These value constraints are used to query the metadata storage. A checkbox can be used to indicate that the search should also be executed on the content that has been ingested into a integrated long-term archive. The tool then connects to a special service interface of the long-term archive which enables query mediation. After executing the query, the instances that conform to the property constraints are inserted into the graph as nodes with a special color. The properties of these instances can then be further inspected.
5.1.6 Reuse

After exploration and discovery of product metadata, the referenced data can be reused. Figure 5.12 displays a simple use case for product data reuse.

![Figure 5.12: Display of Television Case with Multivalent Technology [24]](image)

A user might wish to display a relevant product design which has been annotated with metadata. The display of, e.g., a 3D-graphical product-component CAD object which has been archived in the JT file format can be executed, i.e., interpreted and therefore visualized within a viewing tool with multivalent technologies. As the product design may contain references to metadata, this metadata has to be transformed to the contemporary ontologies before displaying.
5 Implementing Metadata Preservation
6 Evaluating Metadata Preservation

The last chapter described the system model, i.e., its architecture and corresponding exemplary implementations of tools supporting the metadata preservation lifecycle including implementations for dedicated metadata harmonization and ontology engineering functionality. This chapter will first evaluate these metadata preservation components by applying them in some example use cases by means of a cognitive walkthrough [148]. The following section of this chapter then presents the results of an evaluation experiment which was executed among industry practitioners.

6.1 Evaluation Scenario

This section illustrates a PLM knowledge evolution scenario using the implemented tool functionality which was described in the previous chapter. First, an introduction to the evaluation scenario is provided by stating which different metadata and product lifecycle phases are touched by the scenario. The following sections then define ontologies which are relevant for the understanding of the scenario. Finally, the tool functionality which was described in the previous section is used to execute the evaluation scenario.

6.1.1 Introduction and Overview

Figure 6.1 provides an overview of the evaluation scenario. First, ontologies are created for different product lifecycle phases including innovation management and product design. In these product lifecycle phases, metadata is used as annotations for product data models. Ideas are annotated to indicate the business category of the idea and product parts are annotated in order to attach technical specifications. The metadata and annotations are stored in the active PLM data and metadata repositories. Then, the product is manufactured and delivered to customers and service providers. Due to a limited success of the product, it is decided to stop the production. At this stage, all relevant product data and metadata is collected and archived.

Although new products are not manufactured anymore, many physical products are still in operation and need to be operated. This product lifecycle phase might last for a
long time period according to individual service contracts. During this time period it is likely that ontology evolution (both schema and instance evolution) will occur because technologies and knowledge progress, i.e., knowledge representations have to be changed over time. The instance update evolution scenario is taken from the early innovation management phase of a product lifecycle whereas the schema evolution is illustrated by an update of a product classification ontology.

### 6.1.2 Ontology Creation

This section describes the applied ontologies from innovation management and product design. In the following, namespaces prefixes are not displayed and it is assumed that the default namespace is @prefix : <http://www.fernuni-hagen.de/ontologies/>.

#### 6.1.2.1 Ideation Ontology

Innovation is the process of taking an original idea and converting it into a sellable product. Ideas for future products are generated by executing an innovation process which starts when one or more idea authors generate an idea document that captures the list of ideas of the innovation. Innovation management software allows maintaining the idea semantics and visualization. Although nearly all of the ideas are not realized, they are still important company intellectual property and therefore they are archived. An idea contains among title, descriptive text, visual illustrations, and creation date also the business category which is a semantic annotation that includes concepts like...
Beauty Beverage Appliances, Shaving/Grooming, Kitchen Appliances, Sleep, and Television. Later on, the business category allows to search for ideas that the user is interested in.

```prolog
:Ideas = {
    :BusinessCategory a rdfs:Class .
    :Idea a rdfs:Class .
    :hasBusinessCategory a rdf:Property ;
      rdfs:domain :Idea ;
      rdfs:range :BusinessCategory .
    :Television a :BusinessCategory .
} .
```

Figure 6.2: A Simple Product Ideation Ontology

The ontology displayed in figure 6.2 defines the classes BusinessCategory, Idea, and a property named hasBusinessCategory that connects an idea with a business category. In addition, the metadata instance business category Television is defined as part of the domain ontology.

### 6.1.2.2 Product Ontology

As described in previous chapters, product classification standards like UNSPSC and eClass exist in the engineering domain which ensure a common understanding and categorization of products and which define basic technical product properties. This section introduces a simple product ontology which is displayed in figure 6.3.

```prolog
:Products = {
    :Product a rdfs:Class .
    :material a rdf:Property ;
      rdfs:domain :PCBConnector ;
      rdfs:range rdfs:Literal .
    :termination a rdf:Property ;
      rdfs:domain :PCBConnector ;
      rdfs:range rdfs:Literal .
} .
```

Figure 6.3: A Simple Product Part Classification Ontology
The simple product ontology has a root class named `Product` and one subclass `PCBConnector`. `PCBConnector` has two properties named `material` and `termination`. The property `termination` indicates if the connector is available in vertical or horizontal termination for perpendicular and parallel mounting.

### 6.1.2.3 Annotation Ontology

Figure 2.4 displayed a product data model in which annotations connect product data and metadata. It is required that a product data model part can be referenced by using an unique identifier. For the identification of the metadata, an URI can be used that consists of the namespace and the name of the ontology element. Annotations themselves are described by an ontology which is displayed in figure 6.4.

```rdfs
:Annotations = {
  :Annotation a rdfs:Class .

  :hasMetadataURI a rdf:Property ;
  rdfs:domain :Annotation ;
  rdfs:range xsd:anyURI .

  :hasDataID a rdf:Property ;
  rdfs:domain :Annotation ;
  rdfs:range rdfs:Literal .
} .
```

Figure 6.4: Annotation Ontology

First, a class `Annotation` is defined which has several properties. The property named `hasMetadataURI` stores the reference to the metadata and the property `hasDataID` stores the reference to the data. In addition, the ontology has two properties that are not shown in figure 6.4. These properties include the property `hasAuthor` which holds a reference to the creator of the annotation and the property `hasDate` stores the creation date of the annotation.

### 6.1.3 Capturing and Annotation

In the described evaluation scenario, ideation and product design metadata is captured and annotated. While the ideation metadata is annotated to a word processor document, the product design metadata is annotated to a product part.
6.1.3.1 Innovation Management Metadata

The PLM system manages documents created during different product lifecycle phases. The documents that are created during innovation management are also part of the product lifecycle data. An idea has to be described in prose in a word processor document which can be identified in a content management system with the unique identifier fileId992349. To attach meaning to that word document, an annotation is created and stored. The annotation a1234 displayed in figure 6.5 will connect the document with a metadata instance ThreeDTVIdeaInThe70s. This annotation express the meaning that the file describes a television related innovation.

```
:a1234 a :Annotation ;
  :hasDataID fileId992349 ;
  :hasMetadataURI :ThreeDTVIdeaInThe70s .
```

Figure 6.5: Annotation of Ideation Metadata

Figure 6.6 displays metadata instance definitions conforming to the ontology defined in figure 6.2. The definition includes a 3D TV related idea instance from the 1970s named ThreeDTVIdeaInThe70s. This idea instance also contains the predefined business category Television which is the only appropriate business category available at that time.

```
:ThreeDTVIdeaInThe70s a :Idea ;
  :hasBusinessCategory :Television .
```

Figure 6.6: Ideation Metadata Instances created in the 1970s

The annotated idea document is then passed to a collaborative ideation review team who rates the ideas.

6.1.3.2 Product Design Metadata

Some months later, the idea has passed the review and the realization of the product has begun. An engineer will begin to design the television with special design software which allows to annotate parts of the design with metadata. For example, a printed circuit board (PCB) is used as part of the design. The required PCB has to be made of polycarbonate and has to contain a vertical contact which is relevant for manufacturing. These requirements are expressed as metadata that conforms to the product ontology that is displayed in figure 6.3.
Evaluating Metadata Preservation

Figure 6.7: A PCB Connector Instance

Figure 6.7 displays the metadata `pcbConnector0815` which expresses a PCB connector with vertical termination and polycarbonate material. This metadata instance is then annotated to the product part by means of an annotation (figure 6.8).

Figure 6.8: Annotation of Product Part Metadata

The annotation will annotate the product design part that has the identifier named `aras1234` with the `pcbConnector0815` metadata.

6.1.4 Archival

After some additional months, the product is manufactured and brought to market. Unfortunately, after some years it is detected that the customer demand is not as high as previously thought and it is decided to stop the production of the product. Therefore, all relevant product data and metadata has to be archived. The relevant metadata include the ideation and product ontology (figure 6.2 and 6.3) and their instances (figure 6.6 and 6.7) as well as the annotation ontology (figure 6.4) and instances (figure 6.5 and 6.8).

This metadata is collected by the workflow described in figure 5.7. The collected data and metadata is then aggregated into a SIP described by OAI-ORE [108] based packaging information ready for ingestion into a OAIS-based archive. The SIP generation will include a normalization of data and metadata. The data normalization transfers a proprietary product data model into a standard product data model (e.g., PLM/XML [30]). The metadata which conforms to an external schema can be semantically normalized into metadata conforming to an archive local schema which makes preservation more controllable. In addition, metadata can also be syntactically normalized (e.g., N3 [83]).
6.1 Evaluation Scenario

6.1.5 Evolution

This section provides examples of schema and instance evolution of the metadata which has been used to annotated the product lifecycle data.

6.1.5.1 Instance Evolution

According to section 5.1.3, the metadata schema as well as the metadata instances of the ideation and annotation ontology were archived and deleted from the active repository. Some decades later, the business category Television has evolved into several categories due to technology innovations (figure 6.9). These new categories include the new classes ThreeDTV, PlasmaTV, LCDDTV, and LEDDTV. The ontology engineer decided to delete the business category Television.

![Figure 6.9: Business Categories After Ontology Evolution](image)

This new categorization is expressed as a new ontology version. The contemporary application software will use this new ontology version to query the active repository and the long-term archive in parallel. To prevent semantic obsolescence of archived ideas that contain the business category Television, the EVO tool will not delete the class Television physically. It will rather mark the class as deprecated and the ontology engineer will define an alignment between the newly introduced business categories and the previously defined category Television.

To express the relationship between ontology elements of different versions, a dedicated alignment vocabulary [42] can be used. Figure 6.10 displays an example of such a alignment representation between ThreeDTV and Television.
6 Evaluating Metadata Preservation

How this machine readable alignment can be used to discover archived metadata in later metadata preservation lifecycle phases is described below.

6.1.5.2 Schema Evolution

Figure 6.7 displayed a metadata instance which conformed to the ontology displayed in figure 6.3 which was archived. To continue the application scenario, the following ontology evolution occurs.

Property Refinement After archiving, the semantics of the material property is refined (figure 6.11).

It became important to differentiate between contact material and housing material of the PCB. In order to clearly specify the semantics, two new properties are defined. The first property is named housingMaterial and the second is named contactMaterial. The first property defines the housing material of the connector (e.g., polybutylenterephthalat, polycarbonat) while the latter defines the material of the contact material (e.g., copper, brass). The previously defined property material is marked as deprecated.

Successful harmonization of metadata requires metadata engineering tools that trace, publish, and communicate ontology updates in both a human and machine readable way. Update semantics need to describe the compatibility and transformation rules.
6.1 Evaluation Scenario

of the committed ontology updates. The described property refinement is therefore captured as operational metadata update package. Figure 6.12 displays an update package using the Changeset vocabulary [27]. The Changeset vocabulary enables to describe the difference between versions of ontologies by describing additions and removals. A change set include the time and reason for the change as well as statements to be added and removed from a graph.

@prefix products: <http://www.fernuni-hagen.de/ontologies/products/> .

:cs1234 a cs:ChangeSet ;
  cs:createdDate "2011-11-07"^^xsd:date ;
  cs:creatorName "Ontology Engineer" ;
  cs:changeReason "Property Refinement" ;
  cs:preceedingChangeSet :previousCSId ;
  cs:subjectOfChange products:PCBConnector ;
  cs:removal [  
    rdf:subject products:material ;
    rdf:predicate rdf:type ;
    rdf:object rdf:Property .
  ] ;
  cs:addition [  
    rdf:subject products:contactMaterial ;
    rdf:predicate rdf:type ;
    rdf:object rdf:Property .
  ] ;
  cs:addition [  
    rdf:subject products:housingMaterial ;
    rdf:predicate rdf:type ;
    rdf:object rdf:Property .
  ] .

Figure 6.12: Metadata Update Package modelled with the Changeset Vocabulary

In order to conform to the new schema, the properties of the archived metadata have to be translated to the newly modelled semantics by moving the previously defined values for the material either to the housingMaterial or contactMaterial property. While it is difficult to find out the update semantics for a dependent application of an ontology, it is easy for an ontology engineer of such an ontology update. The author of the ontology update or the person that applies the update can define a rule, that moves the material value to the correct property slot. If the update author is not able to clearly specify a transformation rule, then the update of the semantic has to be annotated to
Evaluating Metadata Preservation

The local archive curator will inspect the changes and will determine, if the application of updates requires human intervention or human knowledge to solve the semantic conflict. For example, figure 6.13 displays a such a rule.

```
WITH <http://www.fernuni-hagen.de/ontologies/products>
DELETE { ?s :material ?m }
INSERT { ?s :housingMaterial ?m }
WHERE {
  ?s a :PCBConnector .
  ?s :material ?m .
  FILTER ( ?m = 'Polycarbonate' )
}
```

Figure 6.13: Conditional Metadata Update Package

The rule moves the values of the `material` property to the `housingMaterial` property if the material is polycarbonate.

6.1.5.3 Update Monitoring and Processing

Ontology engineering tools like EVO will publish an ontology update feed. Metadata harmonization functionality executed within a long-term archive will monitor the published update feeds of the archived metadata ontologies. The harmonization functionality will determine whether or not the updates influence the semantics which have been archived or if only non-archived classes and properties are affected by the ontology updates. If archived metadata is affected by the updates and metadata should be migrated, the updates are downloaded, archived and applied. To continue the scenario, the changeset for the instance and schema evolution would be downloaded and archived for future processing.

6.1.6 Exploration

To continue the cognitive walkthrough of the evaluation scenario, it is assumed that some years after the archival and evolution took place, access to archived product data is necessary because it needs to be discovered, explored, and reused.
6.1 Evaluation Scenario

6.1.6.1 Access for Display and Tracing

Several of the televisions have caught fire. Since it is assumed that the fires are caused by a malfunction of an electrical component, an accident investigator needs to check whether the manufacturer has considered existing regulations. In order to understand and trace the product data, it is necessary to restore the complete product design. Therefore, the archived data and metadata has to be accessed from the archive without altering. In addition to the product metadata, all ontologies and referenced ontologies must have been archived. After restoring, the data and metadata can be displayed either with multivalent technologies (emulation) or the application software and operating systems that is necessary to read the product design has also been archived (system preservation) and can be used to display.

6.1.6.2 Query Mediation for Discovery

**Innovation Management**  Twenty years after archival of the television data and metadata, an engineer has a 3D TV related product innovation idea and he remembers that 3D TVs were already envisioned in the 1970s. The engineer explores the active idea repository as well as the long-term archive in parallel because the engineer does not want to reinvent the wheel. The same idea was probably already rejected for some reason or the engineer wants to get inspirations by studying similar ideas. The engineer uses the semantic exploration tool to search for ideas (figure 5.11). The list of business categories can be selected from a drop down box. Since the ontology has evolved, the Television business category is not available any more and only ThreeDTV is available. The engineer marks a checkbox to indicate that the search should also be executed in the archive. By doing so, the previously defined alignments between the business categories can be exploited so that archived ideas conforming to Television category are also be part of the result set.

**Product Design**  A service company needs to purchase a replacement for the broken PCB connector based on the archived requirements. Ontologies are used to search for spare parts. If archived product designs are included during this search, current ontology versions are used that provide means to enter search criteria (e.g., housingMaterial). As the current ontology versions have evolved and are different from those that were archived, the ontologies need to be temporarily mediated. Mediation maps old ontology concepts to new ones so that archived ontologies are suitable for querying.
6.1.7 Reuse

The ontology updates that occurred after archival are not relevant for displaying the archived product data because the complete environment is restored. However, as described above, for discovery based on a contemporary ontology those updates need to be processed by the metadata harmonization functionality. The harmonization is also required when the archived metadata has to be embedded in a current system environment.

6.1.7.1 Transformation for Embedding

Although archiving means to delete product designs from the production environment and daily usage, it is not uncommon that archived product designs are reused (e.g., product variation). Product data reuse means to embed the complete archived product design into the current system environment. The product design also include digital knowledge that is necessary to fully understand the design. This knowledge was captured using a specific domain ontologies. Unfortunately, after several years these ontologies might be obsolete in the current system environment or other domain ontologies are used.

Especially archived semantics has to be adjusted to new real world which means that metadata instances have to be translated into the current world. If the product metadata has to be reused after several years of archiving, it has to be transformed into the representation language of a current ontology version or into another domain ontology. During archive access, it can be specified to which ontology and to which ontology version the metadata should conform. The archival functionality will be able to transform the metadata by exploiting alignments definitions which were created by ontology engineers assisted by ontology engineering tools.

6.2 Evaluation Results

A demonstration and evaluation event was carried out as part of the SHAMAN project activities [124]. The participants of that event were practitioner of an industrial project partner who produces consumer lifestyle products including TV sets. During that event, the metadata preservation tools and the application usage scenarios were presented to the participants. After presenting tools and scenarios, several evaluation questions regarding usefulness and relevance were posed and the participants were asked to critically rate these questions according the perceived importance. The importance value range from very important, moderate important, not important to not known. The perceived ratings
and the corresponding questions which are relevant for this thesis are displayed in table 6.1.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Very Important</th>
<th>Moderate Important</th>
<th>Not Important</th>
<th>Not Known</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalization of PLM data into open standards</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Relevance of automatic and manual capturing of metadata for archiving in design processes</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Practical relevance of terminology evolution</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Applicability of metadata schema mappings for representing terminology evolution</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Benefits of accessing the archive via semantic search using different independent vocabularies</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.1: Importance of Functionality

The overall importance across all rated functionality is displayed in figure 6.14.

After analyzing the ratings made during the event and the distribution of the ratings, the following facts become evident:

- all of the rated functionality is rated important (very and moderate) by at least 66 percent of the participants.
• apart from the functionality application of schema mappings, all other functionalities are rated very important by at least 33 percent of the participants.

• the perceived importance for semantic search is equally rated as important and not important by some participants.

• the most frequent rating for the all functionalities is moderate important.

• the functionality capturing of metadata annotations for archiving is rated as very important by 66 percent of the participants.

In addition to these results, some relevant points were done by the participants during discussion:

**Application Scenario Occurred in Real-Life**  One participant mentioned that the application scenario regarding 3D TV innovations occurred in real-life. When engineers in an innovation laboratory searched for 3D related research and did not know that in 1970s 3D was not called 3D but stereoscopic display they were not able to find relevant documents.

**Importance of Automatic Metadata Capturing**  In addition to the fact that capturing metadata is rated as very important, the participants also indicated that automatic metadata generation is highly desirable in daily work activities of engineers in order to save time.

As a result of the discussions and the analysis of the functionality ratings, it can be stated that the technologies for harmonization of metadata is a move in the right direction. But the harmonization of metadata must be seen as a second step. The first step during metadata preservation is the capturing and annotation of metadata. This finding is a confirmation for the approach of the thesis since it considers metadata preservation to span over the whole metadata lifecycle including capturing. Therefore, the thesis covers the demand of practitioner. However, the requirements of automatic annotation and techniques to implement these requirements have to be studied further. Having deployed these capturing solutions, a large amount of metadata will be created and archived. This fact will lead to a rising demand for the discovery and hence the harmonization of metadata.
7 Conclusion and Outlook

This chapter summarizes the results of the thesis, reviews the research questions, and presents a number of ideas for future research directions.

7.1 Summary

The long-term preservation of metadata which is generated in the design and engineering domain along the product lifecycle is a complex challenge. As an initial step, this thesis took a closer look at this challenge by first studying the state-of-the-art with regard to the long-term preservation of product lifecycle knowledge. The thesis then provided an overview of the features and the scope of the state-of-the-art and based on these findings, relevant and important research gaps were identified. The thesis continued by tackling one of the missing features, i.e., the issue of preserving ontology-based metadata over long time periods. As a conceptual model for this knowledge preservation, the metadata preservation lifecycle was described which not only captures the knowledge, it also manages it even over long time periods in which ontology evolution occurs.

The first phase of metadata preservation is the annotation of metadata to large amounts of heterogeneous digital product data. The annotated metadata express the knowledge of engineers and customers. Its capturing circumvent the loss of important company’s intellectual property. This thesis demonstrated how references to metadata are annotated automatically and manually to various parts of the product data. The annotated metadata conform to various ontologies which represents different domains of the product lifecycle and thus guarantees the discovery, traceability as well as the reuse of the product data in the future.

Several legal and economic requirements demand that this product data and metadata has to be archived together for a long time period. During the ingest process, product data and metadata are extracted and processed separately. The thesis revealed in its analysis of state-of-the-art and related work that the archival of these annotated product data is not sufficient to guarantee its long-term understandability since technology innovations are frequent and terminologies are relatively short-lived compared to some product lifecycles. The terminology which is able to interpret the metadata will become
obsolete due to semantic heterogeneity caused by ontology evolution. Such knowledge obsolescence will have impacts on long-term digital preservation systems whose intention it is to provide access to archived information for the indefinite future. To overcome the loss of understandability of archived product data, the harmonization of metadata is required which also supports the last reuse phase of the metadata preservation lifecycle.

Since dedicated metadata harmonization for archived metadata was not the focus of the related work, this thesis focused on defining missing OAIS-based functionality for the harmonization of metadata. This harmonization functionality requires supporting ontology engineering system functionality which tracks the history of ontologies via domain knowledge provenance graphs. Updates to these graphs are published as operational metadata update packages which are retrieved by the metadata harmonization functionality. After retrieval, the update packages can be stored or executed immediately for metadata migration purposes. If a migration is executed, metadata conform to a new version of the same domain ontology or to another domain ontology. The metadata harmonization also supports the archive consumer during access where it is necessary to query the archived data with an ontology-based metadata schema which may have evolved. Metadata harmonization also includes special functionality for the exploitation of metadata update packages during the temporarily mediation of queries which seek archive contents.

After modeling missing metadata harmonization functionality and required ontology engineering functionality, the thesis continued by describing the implementation of experimental tools along the metadata preservation lifecycle. The implementation demonstrated the usefulness of a product lifecycle which is aware of the generated knowledge and the long-term preservation requirements that respects knowledge evolution. Finally, this thesis evaluated the implementation of the preservation and knowledge aware product lifecycle by specifying several domain ontologies which define the meaning of metadata of different product lifecycle phases. The evaluation scenario continued with the evolution of ontologies which result in the loss of archived metadata. Despite this evolution, the evaluation demonstrated that metadata harmonization keeps archived metadata interpretable.

### 7.2 Contributions

After modeling, implementing, and evaluating the proposed preservation aware product lifecycle, answers to the research questions which were stated at the beginning of the thesis in section 1.3 can be provided.
7.2 Contributions

Semantic Annotation of Product Data

How can product data be annotated with a variety of domain metadata to enable its discovery, traceability, and understanding in the future?

The thesis demonstrated how a commercial PLM tool with a proprietary product data model can be enhanced with functionality for creating, editing, storing, and visualizing annotations of different parts of product data. The annotations which conform to a special annotation ontology allow the separate management of data and metadata. Semantic annotations include references to unique identifiers of the product data and metadata. As the annotated metadata has to enable precise human and machine knowledge exchange and search, it conforms to domain ontologies which attach semantics (meaning) to product data. Since the annotated product lifecycle metadata goes beyond standard preservation metadata, the domain ontology-based metadata can be used for semantic exploration in later product lifecycle phases.

Product Lifecycle Knowledge Representation

How can product lifecycle knowledge be represented for human and machine processing?

Human knowledge is applied during all PLM phases. While the knowledge has to be captured manually in some situations, it can sometimes even be extracted from data automatically. In both cases, the knowledge is captured as ontology-based metadata and then annotated to the different product parts and documents. The thesis pointed out that successful knowledge representation of different PLM phases does not require that the number of necessary domain ontologies is fixed or foreseeable. Hence, the thesis illustrated the usage of different domain ontologies for different product lifecycle phases for the annotation of product data. During implementation of PLM knowledge capturing, the Resource Description Framework was chosen for ontology representation due to its simplicity, wide deployment, and extensive software framework support.

Archival of Annotated Product Data Models

How can product lifecycle data and metadata be aggregated and ingested into long-term archives?
After analyzing the state-of-the-art of the archival of product lifecycle data, it became evident that product data which conform to a proprietary model is normalized into vendor neutral data models before archival. Although the state-of-the-art considers knowledge as an important asset, the necessary preservation requirements of annotations and metadata which are relevant for interpretability of product data were not observed. Therefore, the thesis described how a commercial PLM tool can be enhanced with archival functionality that processes annotated product data models before archive ingestion. Such functionality will first collect all relevant data and metadata. After collecting, serializing, and normalizing relevant data and metadata, the data collection is aggregated into a container which can be ingested into a long-term archive. References between data and metadata are explicitly expressed and the metadata is separated from the data so that ingest functionality can process data and metadata independently.

**Integration of Long-term Archives into PLM Processes**

*How can a long-term archive be integrated as permanently accessible repository and knowledge base into the product lifecycle tool chain?*

Product data and metadata which have been ingested into long-term archives is removed from active product data repositories. Nonetheless, the reuse of archived product data has to be possible. This thesis gave several examples for reusing the captured and annotated knowledge and it demonstrated how this knowledge reuse delivers additional value in PLM processes. Therefore, ingest, search, and access service interfaces were defined which are executed on top of the archival system and which can be accessed tools along the product lifecycle. These interfaces integrate the long-term archive as permanently accessible knowledge base into daily PLM processes. Actors of all product lifecycle phases are able to access archived product data by using ontology-based schemas.

**Archived Metadata is Threatened by Semantic Obsolescence**

*How is the traceability of annotated product lifecycle data be threatened by ontology evolution?*

Captured metadata conforms not only to a specific domain ontology. It also conforms to a specific version of the domain ontology. This metadata is a part of the data collection
which will be archived for later access. After archival, changes in domain conceptualization are reflected by the evolution of ontologies which were used for product data annotation. This thesis provided an overview of different use cases for ontology evolution. The use cases also include non-backward compatible updates which lead to a better understanding of the issues which archived metadata is exposed to. The use cases demonstrated that archived product metadata is in danger of becoming unusable or even lost while the products are in operation for several decades.

Dedicated Metadata Harmonization Ensures Knowledge Reuse

How can product lifecycle metadata be harmonized with OAIS archive functionality?

After presenting the threats that archived metadata is exposed to during updates in domain conceptualization, the thesis addressed one of the core questions as it developed methods to keep archived metadata up-to-date with contemporary ontologies. Special functionality of metadata harmonization in OAIS-based long-term archives was modeled, implemented, and evaluated. This functionality is able to prevent semantic obsolescence of archived product metadata, by providing the necessary harmonization functionality which also interact with ontology engineering tools.

Metadata Harmonization Requires Traced Ontology Engineering

How can ontology engineering functionality support the long-term harmonization of archived metadata?

The thesis draw the conclusion that successful metadata harmonization is impossible without special ontology engineering functionality which allows to query and access metadata updates. This functionality includes domain knowledge provenance graphs which record the history of ontologies and allow to generate and publish operational metadata update packages. This functionality was implemented in a special ontology engineering tool deployable in a distributed system environment. The tool has several service interfaces which allow to query the history of ontology elements and to access update packages for exploitation in archival functionality.
7.3 Future Work

After having dealt with the long-term preservation of product lifecycle metadata, several future challenges still remain. Some of the open challenges which are not directly connected with the thesis topic were already identified in section 3.1.3.3. Additional directions for future research areas and investigations of long-term preservation of product metadata are derived in the following section. The list is provided along the phases of the metadata preservation lifecycle.

Product Metadata Capturing

Automatic Extraction of Metadata. During the evaluation it became evident that the annotation of various document types which are created during the different collaborative product lifecycle phases is important for future reuse. Since manual metadata capturing is time consuming and error-prone, techniques have to be developed which allow automatic extraction of metadata right at data creation-time. The extraction has to be tailored to file formats used in product lifecycle processes.

Product Data and Metadata Archival

Transformation to Standard Product Data Representations. The investigation of the state-of-the-art in the realm of product lifecycle data preservation revealed that proprietary annotated product data models are transformed to vendor-neutral standard product data models. Since several standard product data model exists, it has to evaluated which of them is suitable for the representation of knowledge enriched product data generated during the whole product lifecycle.

Annotation Management. Although the product data model and the semantic annotations are transformed to and stored in a suitable representation, they must be regarded as one coherent component that has to be preserved. For example, if archived product data or metadata is migrated, the semantic relationship between the data and the metadata must be asserted by inspecting and modifying relevant annotations.

Syntactic Metadata Normalization. As metadata conforms to ontology versions, it has to be investigated which language representation is suitable for long-term archival with regard to expressiveness and compactness.
7.3 Future Work

**Ontology-based Metadata Exploration**

**Semantic Exploration Techniques.** Since an archive consumer often lacks the knowledge about the contents which has been archived, ontology schemas which have been used for data annotation help to understand the archived contents. To support the exploration aim of the archive consumer, different ontology visualization techniques (e.g., 3D) have to be evaluated.

**Metadata Harmonization**

**Implement Metadata Migration.** While query mediation has been implemented as part of the metadata harmonization functionality, it will be necessary to implement metadata migration functionality. In addition to migrating domain ontology-based metadata, also the migration of annotations has to be kept in mind if the annotation schema evolves. Also, an ontology engineer may provide rules for undoing the execution of a metadata packages as well as rules to validate if the migration has been executed successfully.

**Metadata Update Processing.** The update management functionality might also include functionality for change visualization which provides a quick overview of the ontology update for better understanding. In addition, some functionality might execute the update in test mode to query change effects of the ontology update.

**Harmonize Preservation Metadata.** During ingest, preservation metadata based on complex XML schema definitions is generated and archived. This preservation metadata is also threatened if schema definitions evolve. Techniques for schema agnostic transformations have to be found in order to allow the migration of metadata conforming to different schema languages.

**Ontology Engineering**

**Metadata Update Ontology.** The metadata update packages used in the thesis are expressed as SPARQL/Update [122] queries. To express the semantics of an update more explicitly, a suitable metadata update ontology has to be found. The ontology engineer should be able to classify the updates according to its severity regarding backward compatibility. If the ontology engineer wants to create a new release, he should be able to select relevant updates for publishing. This update ontology has also support a large variety of complex ontology update patterns.

**Pushing Metadata Update Packages.** According to the system model presented in this thesis, operational metadata update packages are pulled by interested applications.
To manage the updates even more automatically, metadata update packages could also be pushed to interested and dependent applications. These applications then could archive the update packages for later use or they could be executed immediately in case of backward compatible updates.

**Alignment Maintenance.** During ontology evolution, special consideration has to be taken for alignments. Because different conceptualizations of a domain exists, alignments are defined by humans or are automatically generated by matching algorithms. Since vendors of commercial product ontologies are not willing to maintain alignments to other product ontologies and since the alignment specification requires a large amount of intellectual effort, the maintainance of alignments under schema evolution has to be examined as well.
Bibliography


[40] ETIM Deutschland e.V.: ETIM Klassifikationssystem. Internal Document,

[41] ETIM Deutschland e.V.: ETIM Klassifikationssystem. http://www.etim.de,


[132] Stojanovic, L.: Methods and Tools for Ontology Evolution, Karlsruhe University, Germany, Diss., 2004


[139] **Van Rees, R.:** Clarity in the usage of the terms ontology, taxonomy and classification. In: *Proceedings of the 20th International Conference on Construction IT*, 2003


[142] **Vogel, T.:** *Wissensbasiertes und prozessorientiertes Innovationsmanagement WPIM*, Fernuniversität, Germany, Diss., 2011

[143] **W3C OWL Working Group:** Web Ontology Language Reference, W3C Recommendation. (2004), February


[145] **Walcker, C.:** Erzeugung und Visualisierung von semantischen Annotationen im Product Lifecycle Management. 2011


