Knowledge Representation and Decision Support for Managing Product Obsolescence

Liyu Zheng

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
In
Industrial & Systems Engineering

Committee Chair: Janis P. Terpenny
Committee Member: Subhash C. Sarin
Committee Member: Robert H. Sturges
Committee Member: Peter A. Sandborn

December 9, 2011
Blacksburg, VA

Keywords: Obsolescence, Knowledge Representation, Ontology, Decision Support, Design Refresh

Knowledge Representation and Decision Support
for Managing Product Obsolescence

Liyu Zheng

ABSTRACT

Fast moving technologies have caused high-tech components to have shortened life cycles, rendering them obsolete quickly. Technology obsolescence creates significant problems for product sectors that use components that are only available for a short period of time for manufacture and maintenance of long field-life systems. Technology obsolescence can make design changes of systems prohibitively expensive, and results in high life cycle costs of systems. While the impact and pervasiveness of obsolescence problems are growing, existing tools and solutions are lacking the needed information and knowledge to do much more than focus on reactively managing obsolescence. Current methods and tools are limited by data conflicts and data inexplicitness, incompleteness, and inconsistency.

In response to the drawbacks of current tools, comprehensive knowledge representation that allows information sharing, reuse, and collaboration on obsolescence issues across different organizations is required. Further, decision making tools that can support proactive and strategic obsolescence management are needed. The purpose of this research is to establish an ontology-based knowledge representation scheme for information sharing, reuse, and collaboration on obsolescence issues, and develop decision making models to support proactive and strategic management for overall cost savings in managing obsolescence.

Three primary aspects of this research are investigated. First, ontologies for obsolescence knowledge representation are developed in a systematic way with the use of UML diagrams. The generality of the developed ontology is demonstrated with distinct examples. Diminishing Manufacturing Sources and Material Shortages (DMSMS) obsolescence provides the basis for this study. Second, an ontology-based hybrid approach for integrating heterogeneous data resources in existing obsolescence management tools is proposed. Third, decision support models are developed and formalized, and include the obsolescence forecasting method for proactively managing obsolescence, and the mathematical models to determine the optimal design refresh plan to minimize the product life cycle cost for strategic obsolescence management. Finally, the design of the obsolescence management information system is provided along with a system evaluation methodology.

Ultimately, the research contributes to the field of knowledge representation as well as design for managing product obsolescence.

Acknowledgements

First and foremost, I would like to express thanks to my advisor, Dr. Janis P. Terpenny for her utmost guidance, support, encouragement, and patience during my journey towards a Ph.D.. Dr. Terpenny helped me tremendously from the first day I joined her research group. She taught me much more than her extensive knowledge, but also enthusiasm and positive attitude towards work and life. I will be forever grateful, thankful and honored for all the valuable learning and advice she has given me.

I would like to thank my dissertation committee members, Dr. Peter Sandborn, Dr. Subhash Sarin, and Dr. Robert Sturges for their time and energy in serving on my committee. I am honored to have had the chance of working with Dr. Sandborn from the University of Maryland in our collaborative project. His utmost support, useful suggestions and comments helped to shape the dissertation work. To Dr. Sarin and Dr. Sturges, I am very grateful for their probing questions and validation of my research in order to make the dissertation better.

I would like to thank my friends in the SMART lab as well as in the NSF e-Design Center, including Dr. Richard Goff, Dr. Nicholas Polys, Lynette Wilcox, Nihal Orfi, Yanfeng Li, William (Mike) Butler and Ni Hu, for the study and happy times together. I would like to thank a former student of the SMART lab, Dr. Xiaomeng Chang, for her nice work in the ontology area. I learned a lot from her project. I would like to give my appreciation to my coworker Raymond Nelson III for all the help, suggestions and discussions he shared with me during our collaborative work.

I am inspired by and thankful to all friends around the world, United States, and China. Their continual support and friendship have been the great motivation to pursue my dream. I am grateful to my former advisor and teachers in China for their guidance that well prepared me for studying in the United States and will serve me well for life.

This dissertation is dedicated to my parents, Nanxing Zheng and Jingfang Zhu, and all the other family members. As your single daughter, I know how hard it is for you to be apart from me. Thank you for consistently encouraging me to be brave, independent, and optimistic. I appreciate your support and love throughout all these years.

Table of Contents

List of Figures

List of Tables

Knowledge Representation and Decision Support for Managing Product Obsolescence

Chapter 1 Introduction

1.1 Background

1.1.1 Present Situation in Product Obsolescence Management

Fast moving technologies have caused high-tech components to have shortened life cycles, rendering them obsolete quickly. Some estimate that over 2000 electronic components pass into obsolescence every month (Prophet, 2002). Technology obsolescence creates significant problems for product sectors that use components that are only available for a short period of time for manufacture and maintenance of long field-life systems. These complex systems, such as military and aircraft avionics, are often manufactured for many years and maintained for decades. Obsolescence often emerges before systems are even fielded and recurs throughout the support life. For example, in the surface ship sonar system, over 70% of the parts are obsolete before the system is installed (NSWC Crane; Singh, Sandborn, 2006). Since product sectors of long field-life systems have no control over critical components when they become obsolete, obsolescence of components makes design changes of systems prohibitively expensive. Ultimately, the ongoing challenges of obsolescence result in high overall costs for maintaining systems. It is anticipated that the costs of this problem will continue to increase in the future with the increased prevalence of obsolescence as products and systems become ever more technologically sophisticated.

With growing impact and pervasiveness of the problem, many tools have been developed and are being used for managing product obsolescence. Most current tools are based on a database and characterized by maintaining the database with the information of obsolete components. CAPSXpert database consists of technical data on over 2.7 million semiconductors. PARTSXpert database provides summaries of the obsolescence status of components based on a stoplight system that depends on the number of manufacturers available for the part.

The most prominent effort of obsolescence management is the Diminishing Manufacturing Sources and Material Shortages (DMSMS) Knowledge Sharing Portal (KSP) of the Department of Defense (DoD). The KSP provides a forum for government programs, and industry and supplier representatives to identify and solve obsolescence associated problems. At the heart of the KSP is the Shared Data Warehouse, which is a web-based system to identify commonality among different organizations including obsolete parts or commercial off-the-shelf (COTS) parts.

There are also many other obsolescence management tools and solutions like the Component Obsolescence and Reuse Tool (CORT) of Raytheon, COMET™ of IHS, Q-Star™ of QinetiQ, IGG's obsolescence monitoring service, and ARINC's obsolescence management program. They can report current obsolescence status of parts, forecast obsolescence risk, and enable the identification of possible alternative parts.

However, current obsolescence management tools are predominately populated by a set of data and service providers that are neither integrated with each other, complete, consistent in the information provided, or even contain consistent definitions of the quantities tracked and achieved. It is unclear if the data available is a complete description of the state of a part and data conflicts are more common than data agreement (Terpenny, Sandborn, Rai, 2009). Existing obsolescence solutions have focused on reactively managing obsolescence at the component level, which resolve the problem after it has occurred.

In order to manage product obsolescence with lower overall costs in the long term, obsolescence solutions must move beyond reactive management at the component level to proactive and strategic management at the product or enterprise level. Knowledge representation that allows information sharing, reuse, and collaboration on obsolescence issues across different organizations needs to be established. Based on the knowledge representation scheme, decision making tools that support proactive and strategic obsolescence management need to be developed.

1.1.2 Motivation

One of the most promising ways to represent domain knowledge is to develop standardized ontologies that domain experts can use to share and annotate information in their fields. Ontology is the standard formalized and explicit explanation of the terms in the domain and relations among them (Gruber, 1993). Ontology defines a common vocabulary for sharing information in a domain. Ontologies have become common on the World-Wide Web (WWW). For example, Yahoo and Amazon.com use ontologies to develop taxonomies for categorizing web sites (Yahoo) and products for sale (Amazon).

There have been a variety of interpretations and implementations of knowledge representation schemes based on ontology. Consider the Common Knowledge Aided Design System (KADS) as an example. The design process of assembly systems has been modeled using three levels of knowledge: task knowledge, inference knowledge, and domain knowledge. The task knowledge level defines the reasoning tasks required to achieve a specific goal. The inference knowledge level defines what inferences are needed to fulfill the reasoning tasks. The domain knowledge level defines all the specific concepts needed by the inferences. The unified ontology for the assembly domain supports all of the decisions during the design of assembly systems (Lohse, Valtchanov, Ratchev, Onori, Barata, 2005).

The current challenge for obsolescence management is the lack of knowledge representation. The obsolescence related data distribute in heterogeneous resources without unified information formats. This hampers proactive and strategic obsolescence management.

Knowledge representation in the domain of obsolescence is expected to support various obsolescence resolution activities (reactive, proactive, and strategic) across the entire group. Similar to the Knowledge Aided Design System (KADS), the domain knowledge level needs to construct a common vocabulary for sharing obsolescence information. It allows obsolescence parts that are closely related in resolution to be grouped together. This will provide the means for managing obsolescence in a systematic way as it is important to distinguish obsolescence part data and obsolescence knowledge. The task

and inference knowledge level needs to define the resolution ontology for resolution activities. This ontology will support the capture and retention of the intent and justification of resolution activities of obsolescence. And the most cost-effective solution consistent with support requirements will be selected from many resolution alternatives.

There are three types of resolutions to DMSMS obsolescence: reactive, proactive and strategic management. For solving DMSMS obsolescence with lower overall cost in the long term, decision support models for strategic obsolescence management needs to be developed based on the knowledge representation scheme. Design refresh planning is one of the effective ways of realizing strategic obsolescence management.

Planning is necessary for organizations to systematically execute activities to achieve a desired goal on some scale by creating and maintaining a plan. A plan can help management to clarify and focus their business development and prospects, and provide a considered and logical framework within which a business can develop and pursue strategies over the following several years. Production planning is a typical planning activity existing in manufacturing organizations. Aggregate production planning is concerned with the determination of production, inventory, and work force level to meet fluctuating demand requirements over a planning horizon. The planning horizon is often divided into periods. Demand can be obtained from forecasting. For example, consider lot-size production planning models where set-up to initiate the production of an item is considered. Lot-size models are used to determine the relative frequency of set-ups to minimize the sum of variable production costs, the inventory holding costs and the set-up costs for all items over the planning horizon of a number of periods, within the inventory balance constraints, the resource constraints and other constraints. They are mixed integer linear program models. Similar to lot-size models for production planning, the objective of design refresh planning for strategic obsolescence management is to determine the optimal plan of design refreshes taken during the support life of the system including when to do design refresh and what obsolete/non-obsolete system components should be replaced at a specific design refresh to minimize the total life cycle cost of the system. The planning horizon is the life cycle of the system, which is also divided into periods. At the beginning of each period, a decision whether design refresh is taken needs to be

determined. At each design refresh, costs include system change costs, testing costs, requalification costs and any other system-level costs that need to be considered. Such costs can be treated as set-up costs to initiate a design refresh. Hence, how to build design refresh planning models can be learned from the production planning models. Mixed integer linear programming techniques are suitable and can also be considered in design refresh planning.

Obsolescence knowledge representation and strategic obsolescence management provide a basis for the obsolescence management information system development, since the three fundamental components of the information system are data management, decision model management, and the user interface.

1.2 Research Purpose and Objectives

The purpose of this research is to establish an ontology-based knowledge representation scheme for information sharing, reuse, and collaboration on obsolescence issues across different organizations, and to develop decision models to support proactive and strategic management for overall cost savings in managing obsolescence. Diminishing Manufacturing Sources and Material Shortages (DMSMS) obsolescence is selected as the context for this study. The ontology-based knowledge representation for obsolescence issues is intended to:

- Capture the domain knowledge in product obsolescence;
- Define a set of data and their structure and relations for obsolescence information sharing;
- Support evolution, updating and maintenance of the obsolescence knowledge base.

Decision models are intended to:

- Make obsolescence forecasting for proactively managing DMSMS obsolescence;
- Determine the optimal design refresh plan to minimize the total life cost of the system for strategically managing DMSMS obsolescence.

Utilizing advantages of ontology in knowledge capture, storage, sharing, and retrieval, and optimization approaches for decision making, this research will make contributions to the field of knowledge representation as well as decision support for managing product obsolescence. Ultimately, a collaborative environment will be created to integrate different organizations suffering from obsolescence and the cost of managing obsolescence will be significantly reduced.

Based on the research purpose and objectives, primary research questions and research sub-questions are defined in Section 1.3.

1.3 Research Questions

Primary research questions, supported by sub-questions, provide the basis and outline for the proposed work.

1.3.1 Primary Research Questions

How can an ontology-based knowledge representation scheme and decision support models help different organizations suffering from obsolescence to capture, manage, and reuse obsolescence knowledge, to make better decisions in proactively and strategically managing obsolescence, and to solve the significant limitations that exist in current obsolescence management tools, where information required to make decision may not be available, may lack consistency, and may not be expressed in a generalized manner, and obsolescence is always solved in a reactive way after it has already occurred?

The topic will be studied in the context of DMSMS obsolescence in this research and lay the foundation for a generalized and comprehensive knowledge representation scheme for information sharing and optimization models for decision support.

1.3.2 Research Sub-questions

1. Questions on Knowledge Representation

Q1. How should an obsolescence ontology be developed? What tool and methodology are used to support the ontology development process and help in the evolution, updating and maintenance of the ontology?

Q2. What concepts and properties related to obsolescence need to be structured in the ontologies? Is the developed obsolescence ontology generalizable, i.e., suitable to a wide range of application?

2. Questions on Information Integration

Q3. What technology can be applied for solving semantic heterogeneity for web-based implementation?

Q4. What approach can be used for integrating information for distributed heterogeneous data sources? How can this approach be applied?

3. Questions on Decision Support

Q5. Considering that forecasting results are the important inputs for proactive and strategic obsolescence management, what obsolescence forecasting method can be developed based on the ontological knowledge representation scheme?

Q6. Considering that strategic obsolescence management is a promising way of significantly reducing overall cost related to obsolescence in the long term, what method is used to realize strategic obsolescence management? How can this method be applied?

4. Questions on System Development and Evaluation

Q7. How can the obsolescence management information system be developed and evaluated?

Q8. Which ontology editor (such as Protégé, OilEd, OntoBuilder, etc) is best suited and how is the obsolescence ontology built in the chosen ontology editor?

1.4 Research Approaches and Methods

Three primary aspects of this research will be investigated. First, a knowledge representation scheme in the domain of obsolescence is established with the utilization of ontology. The UML (Unified Modeling Language) tool is used for capturing and visualizing the design of the obsolescence ontology, and helps in its updating and maintenance. The obsolescence ontology is developed in a systematic way with the combination of top-down and bottom-up approaches. Two distinct examples are presented to demonstrate the generality of the developed ontology. Second, the Semantic Web technology is introduced for the purpose of information integration. Ontologies are constructed with a Semantic Web language OWL (Web Ontology Language), which build the obsolescence knowledge base. The ontology-based hybrid approach is proposed for integrating heterogeneous data resources in existing obsolescence management tools. Third, decision support models are developed for proactive and strategic obsolescence management. The obsolescence forecasting method based on the product life cycle is represented with SWRL (Semantic Web Rule Language) rules. Mathematical models are developed to determine the optimal design refresh plan to minimize the overall obsolescence management cost, with and without the consideration of the uncertainty of the obsolescence dates. The application of these models is demonstrated with an industrial example.

The obsolescence ontology is built in the ontology editor Protégé. Preliminary work on the obsolescence management information system development and evaluation is also discussed.

1.5 Research Contributions

It is anticipated that this research will make significant contributions to the fields of knowledge representation and decision support for managing product obsolescence:

- A knowledge representation scheme established to facilitate information sharing and collaboration for obsolescence management;

- Mitigation efforts made on data conflict between existing obsolescence management tools and across different organizations;
- Proactive and strategic obsolescence management resolution methodology and tools for overall cost savings for the long term and avoidance and minimization of future obsolescence issues.

1.6 Outline of Dissertation

As shown in Figure 1-1, following the first introductory chapter, the second chapter includes a review of literature related to product obsolescence, knowledge representation based on ontology, information integration, decision support for obsolescence management, and system development and evaluation. Chapter 3-Chapter 6 present the research methods and results. Chapter 3 presents the method and results of establishing the obsolescence knowledge representation scheme and associated ontology. Chapter 4 presents the method and results of integrating heterogeneous obsolescence data information. Chapter 5 presents the method and results of obsolescence forecasting and design refresh planning for proactive and strategic obsolescence management. Chapter 6 discusses preliminary work on the obsolescence management system development and evaluation utilizing the results from Chapter 3-5. Contributions and future work are summarized in Chapter 7.

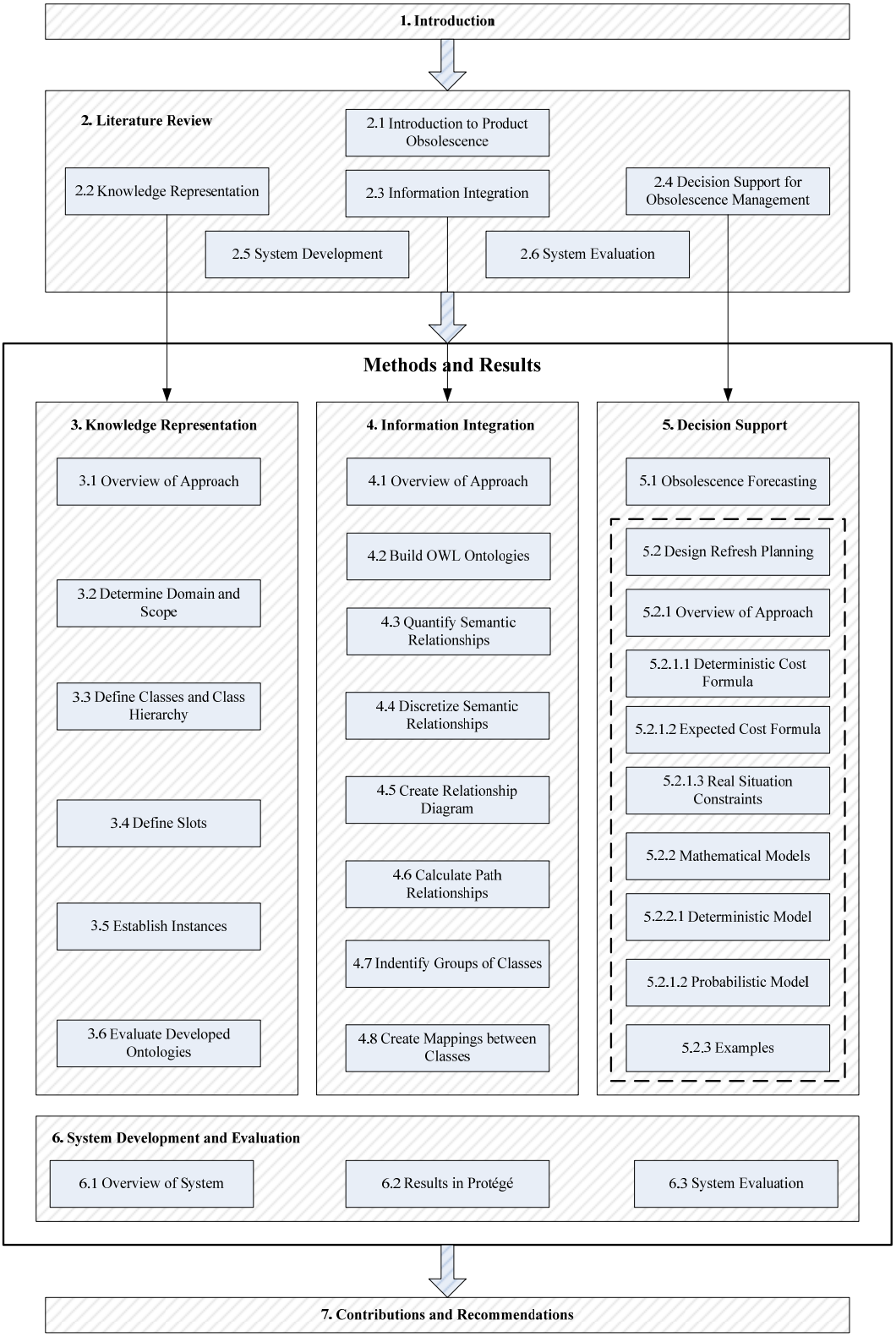


Figure 1-1: Pictorial overview of dissertation

Chapter 2 Literature Review

This chapter begins with an introduction to the problem of product obsolescence. Then a review of work that has been done to solve the obsolescence problem is provided. This is then followed by sections that provide background and review of methods and tools that relate to the approach and methodology of this work. Topics covered include knowledge representation, information integration, decision support, system development and evaluation. These methods are then discussed in the context of the obsolescence problem: reactive, proactive, and strategic obsolescence management. The chapter concludes with a summary and highlight of the needs for the proposed work as evidenced by the literature.

2.1 Introduction to Product Obsolescence

In this section, the prevalence of product obsolescence and its serious consequences are introduced first (Section 2.1.1). Then types of obsolescence are identified and explained in detail. Diminishing Manufacturing Sources and Material Shortages (DMSMS) obsolescence is intended to be the focus of this research (Section 2.1.2). Finally, the status quo of DMSMS management tools is analyzed and causes of common root failure for each is indicated (Section 2.1.3).

2.1.1 Prevalence of Product Obsolescence

Products evolve to accommodate competitive market pressures, rapid rates of technology change, constant improvements in the performance and functionality of products, and the transient and multi-dimensional nature of customer needs. Most high-volume consumer oriented products must adapt with the newest materials, parts, and processes in order to prevent loss of market share to competitors. Examples are prevalent in many high-volume consumer oriented electronics, e.g., personal computers (*example 1*), mobile phones (*example 2*), video and audio players (*example 3*), etc. Figure 2-1 shows the rapid rate of change of computer CPU development.

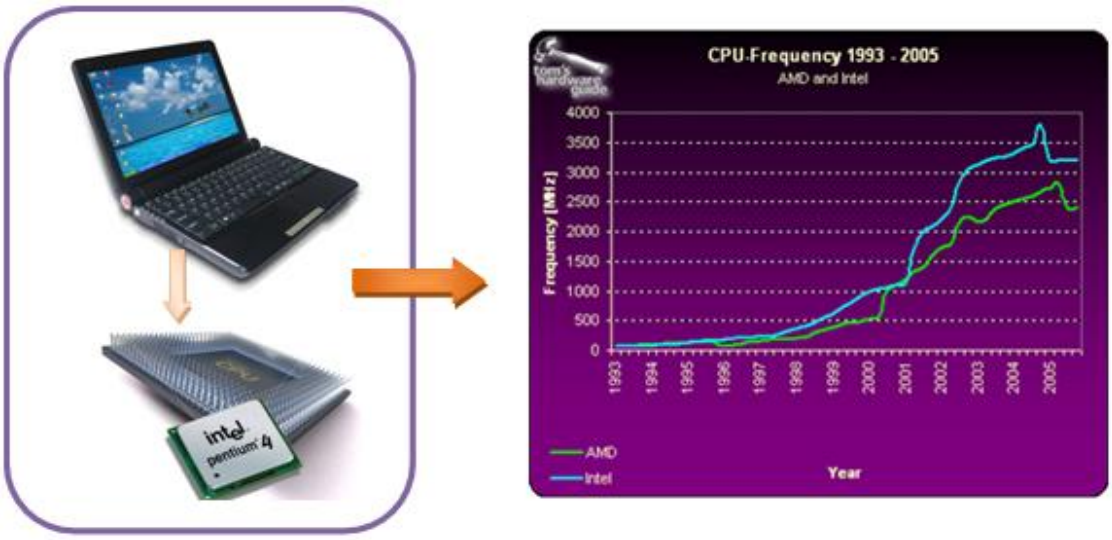


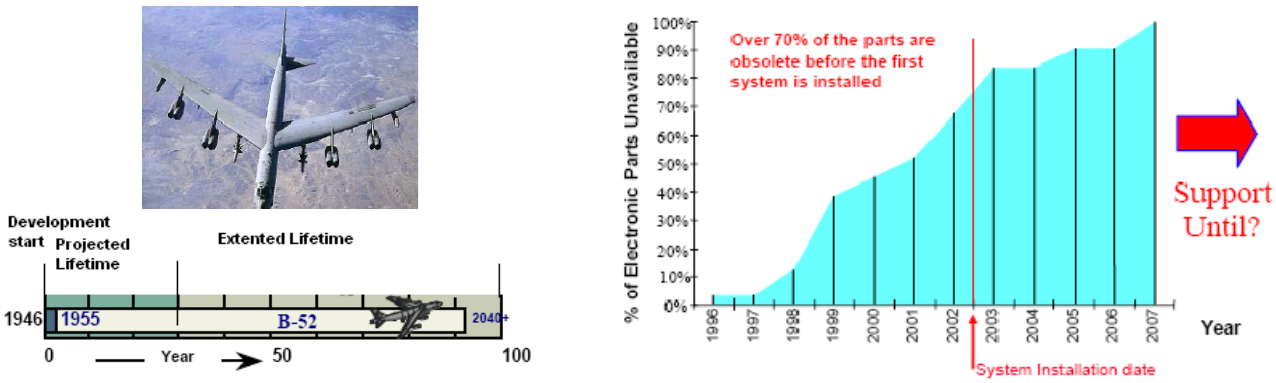
Figure 2-1: Rapid rate of change of computer CPU development

(http://www.tomshardware.com/reviews/mother-cpu-charts-2005,1175.html)

While adding functionality and value, the fast moving technologies also make products obsolete quickly. Technology obsolescence is a significant problem facing many long field-life sustainment-dominated[2-1] systems. Typical examples are low-volume products such as those utilized by military and aircraft avionics. These complex systems are often produced over many years and maintained for decades. Consider the B-52, a bomber which was introduced in 1955 and is expected to be in service until 2040 (*example 4*, Figure 2-2(a)). This is over 80 years of service life! For such long-field life systems, obsolescence problems often emerge before the systems are even fielded and recur throughout the support life. For example, in the surface ship sonar system, over 70% of the parts are obsolete before the first system is installed (*example 5*, Figure 2-2(b)). According to one estimate, over 2000 electronic components pass into obsolescence every month (Prophet, 2002). Obsolescence of components is more serious in military systems as manufacturers have stopped producing low-volume components for the military and have shifted production to the high-volume components market. The military share of the microcircuit market shrank from 17% (1975) and 7% (1985) to now, less

[2-1]Sustainment refers to all activities necessary to keep an existing system operational, continue to manufacture and field versions of the system that satisfy the original requirements, or manufacture and field revised versions of the system that satisfy evolving requirements (Singh, Sandborn, 2006).

than 0.1% (2002) (although decrease in percentage does not mean decrease in actual number if total number base increases). The problem gets worse when one considers military systems are expected to remain in service for a much longer time than commercial systems. Military aircraft are an average of 22.5 years old (US Air Force, 2003) compared to 11 years for commercial aircraft (Bureau of Transportation Statistics, 2006). Since military systems have no control over critical parts that become obsolete, obsolescence of components makes design changes of systems prohibitively expensive (due to re-qualification and re-certification requirements).



(a) B-52 bomber (b) Surface ship sonar system (NSWC Crane)

Figure 2-2: Impact of obsolescence on long field-life systems

The issue of obsolescence goes beyond aging military systems. The problem is increasing in scope and severity for more and more industries facing the faster pace of technology changes than product life cycles, especially in systems with a long field-life (Barahona, Bermon, Gunluk, Hood, 2005). Examples are diverse and include media broadcasting (Blumenfeld, 2007), heating ventilation air conditioning (HVAC) (Goggins, 2007), publishing (Charlesworth, 2007), semiconductor manufacturing (Mutschler, 2008), and electronics manufacturing (Carbone, 2006). For example, obsolescence impacts long life medical instruments where support requirements for healthcare systems can extend beyond 10 years (Hoorickx, 2008) (*example 6*). Obsolescence also affects everyday users of technology. For example, users of personal computers are forced to add memory or upgrade to a complete new computer for adapting to a newer version of software which comes into the market with new features (*example 7*). Obsolescence results in a high life

cycle cost for the system. The cost will continue to increase in the future with the increased prevalence of obsolescence.

2.1.2 Definition of Product Obsolescence

Based on our prior work, four types of obsolescence have been identified (Terpenny, Sandborn, Rai, 2009):

1. **Functionality Improvement Dominated Obsolescence (FIDO)**: Manufacturers cannot maintain market share unless products evolve in order to keep up with competition and customer expectations. (Manufacturers are forced to make changes to products caused by market pressures.)
2. **Logistical Obsolescence**: Loss of the ability to procure the parts, materials, manufacturing or software necessary to manufacturing and/or support a product.
3. **Functional Obsolescence**: The product or subsystem still operates as intended and can still be manufactured and supported, but the specific requirements for the product have changed, and as a result, have made the product's current function, performance or reliability obsolete.
4. **Technological Obsolescence**: More technologically advanced components have become available. (This may mean that inventory still exists or can be obtained for older parts that are used to manufacture and support the product, but it becomes a technological obsolescence problem when suppliers of older parts no longer support them.)

Examples of FIDO obsolescence are *example 1-example 3* mentioned in Section 2.1.1. While this type of obsolescence is not the focus of this research, the following is a brief introduction to the FIDO problem. For managing FIDO obsolescence, governing principles and proven, teachable guidelines are needed that prevent obsolescence. These are best addressed in the early conceptual design stage (Terpenny, Sandborn, Rai, 2009). In this preliminary work related to FIDO obsolescence management, a "piggybacking" strategy, that enables renewed functionality of a technologically obsolete product through the integration or add-on of secondary devices or components is applied (Rai, Terpenny, 2008). An example of "piggybacking" is the MobiBLUs' MP3 cassette player. This

product combines digital storage of MP3 and high quality sound of cassette players, helping to lengthen the life, or sustain cassette players (http://www.dansdata.com/dah220.htm). In this work, a set of formal principles is derived from the results of an empirical study of different products for guiding the design of "piggyback" products.

The last three types of obsolescence (logistical, functional and technological) are referred to as **Diminishing Manufacturing Sources and Material Shortages (DMSMS) obsolescence**: the loss or impending loss of original manufacturers of items or suppliers of items or raw materials (Department of Defense). Examples of DMSMS obsolescence are *example 4-example 7* mentioned in Section 2.1.1. Although it is not strictly limited to electronic systems, much of the effort regarding DMSMS deals with electronic components that have a relatively short lifetime. This is an important and costly problem. For example, Defense Supply Centre Columbus (DSCC) deals with procurement and quality assurance of over 2.2 million spare parts. On an average 10,000 parts become obsolete every year because of discontinuance of production by manufacturers. It is reported that 84% of the discontinued items are electronic components with the rest being mechanical and passive devices (Tomczykowski, 2003). DMSMS is the problem domain of this research and will be discussed throughout the document.

2.1.3 Status Quo of Obsolescence Management Tools

With growing impact and pervasiveness of the problem, many tools have been developed and are being used for managing DMSMS obsolescence. Most current tools are based on a database and characterized by maintaining the database with the information of obsolete components.

CAPSXPert database (bought by Partminer from Information Handling Services, Incorporated around June 2000) consists of technical data for over 2.7 million semiconductors, including 772,505 ICs, 127,465 optoelectronic semiconductors, and over a million discrete semiconductors. CARSXpert provides the technical specifications data, including part number, manufacturer, supply voltage, manufacturing process technology, and whether the component meets the guidelines for military use requiring high reliability.

CAPSXpert also provides access to manufacturer product datasheets (http://www.partminer.com/main).

PARTSXpert database (maintained by Manufacturing Technology, Incorporated) provides summaries of the obsolescence status based on a stoplight system that depends on the number of manufacturers available for the part. Red indicates the component is out of production. Yellow indicates only one manufacturer exists or none exists but current needs are met by on-hand stock or a pending "life of type" (LOT) buy. Green indicates two or more manufacturers exist, with no known plans to terminate production (http://www.totalpartsplus.com).

The most prominent effort of DMSMS obsolescence management is the DMSMS Knowledge Sharing Portal (KSP) of the Department of Defense (DoD). The KSP provides a forum for program representatives including government program, industry and supplier representatives to identify and solve obsolescence associated problems. At the heart of the KSP is the Shared Data Warehouse, which is a web-based system to identify commonality among different organizations including obsolete parts or commercial off-the-shelf (COTS) parts. The KSP facilitates the identification of components that can no longer be procured and consolidates the demand for and inventory of obsolete parts across the DoD enterprise. The KSP then allows various stakeholders (various programs requiring obsolete parts) to work together to identify and implement common resolutions (http://www.dmsms.org).

There are also many other DMSMS obsolescence management tools and solutions like the Component Obsolescence and Reuse Tool (CORT) of Raytheon (Drake, 2003), Component Obsolescence Management and Engineering Tool (COMET™) of IHS (http://www.tactrac.com/comet/login.jsp), Q-Star™ of QinetiQ (http://www.qtec.us/Default.aspx?p=DynamicModule&pageid=248016&ssid=104770&vnf=1), IGG's obsolescence monitoring service (http://www.igg.co.uk/), and ARINC's obsolescence management program (http://www.arinc.com/products/dmsms/index.html) . These tools and solutions have similar functions and they can report current obsolescence

status of parts, forecast obsolescence risk, and enable the identification of possible alternative parts.

However, current tools and solutions for DMSMS obsolescence management are predominately populated by a set of data and service providers that are neither integrated with each other, complete, consistent in the information provided, or even contain consistent definitions of the quantities tracked and archived. It is unclear if the data available is a complete description of the state of a part and data conflicts are more common than data agreements (Terpenny, Sandborn, Rai, 2009). The common root failure is the lack of knowledge representation models for obsolescence management. This hampers the introduction of supply chain knowledge like consolidation of demand and inventory in the application of these tools and solutions. Hence existing work on DMSMS obsolescence has focused on reactively managing obsolescence at the component level, which means minimizing the cost of resolving the problem after it has occurred. Reactive approaches ignore long term solution paths to avoid future DMSMS issues. In order to solve DMSMS obsolescence with lower overall cost, DMSMS solutions must move beyond reactive management at the component level to proactive and strategic management at the product or enterprise level. Knowledge representation that allows information sharing, reuse, and collaboration on obsolescence issues across different organizations provides the basis of proactive and strategic management for DMSMS obsolescence. To establish a comprehensive knowledge representation scheme for DMSMS obsolescence, ontology, a backbone information model, will be required (Sandborn, Jung, Wong, Becker, 2007).

Ontology is an explicit formal specification of the terms and their relations for sharing information in a domain (Noy, McGuiness, 2001). It is expected to help represent, manage and utilize knowledge in the obsolescence management system, consider different criteria and relations in decision making, and enhance the quality and efficiency of product obsolescence management.

In this research, ontology will be used to represent the domain knowledge of DMSMS obsolescence and integrate obsolescence data in obsolescence management tools with

different formats for sharing information and collaborating on DMSMS obsolescence issues. A decision making methodology will be developed and take advantage of a knowledge representation scheme to support proactive and strategic management of DMSMS obsolescence. It is expected to significantly reduce the cost in managing DMSMS obsolescence.

In the remainder of chapter 2, knowledge about ontology representation, information integration, and decision support is provided. Ontology, UML and Model Driven Architecture for obsolescence knowledge representation are introduced in Section 2.2. Concepts about Semantic Web and OWL, and approaches for integrating heterogeneous data information are presented in Section 2.3. In Section 2.4, three types of obsolescence resolutions (reactive, proactive, strategic) are described, and decision making model realization techniques are discussed. Finally, basic knowledge about information system development and evaluation is introduced in Section 2.5 and 2.6.

2.2 Knowledge Representation

While the idea of ontology emerged in applied artificial intelligence some time ago as a means for sharing knowledge (Gruber, 1993), its adoption into practice has been slow. Software engineering tools and methodologies are needed to support or partially automate the development process and help in the evolution, updating and maintenance of ontologies throughout their development life cycles. UML (Unified Modeling Language), a well-known software modeling language has been proposed for ontology development by some researchers (Cranefield, 2001). However, UML, as a software engineering approach, does not completely support the representation of ontology concepts. To enable the use of UML for ontology development, the MDA (Model Driven Architecture) of OMG (Object Management Group) can be applied.

In this section, basic concepts of OWL and UML are introduced first (Section 2.2.1 and Section 2.2.2). MDA for ontology development is then presented in Section 2.2.3.

2.2.1 Ontology and Its Development

An ontology defines a common vocabulary for sharing information in the domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them. Some of the reasons to develop ontologies are: (1) to share common understanding of the structure of information among people or software agents (Musen,1992), (2) to enable reuse of domain knowledge, (3) to separate domain knowledge from operational knowledge (McGuinness, Wright, 1998), and (4) to analyze domain knowledge (McGuinness, Fikes, Rice, Wilder, 2000).

In practice, developing an ontology includes defining **classes** in the ontology, arranging classes in a taxonomic hierarchy (classes, subclasses, and super classes), defining **slots** and specifying allowed values for these slots, creating **instances** of classes and inputting/editing values of their slots (Noy, McGuinness, 2001). There are seven steps in ontology development, including: (1) Determine the domain and scope of the ontology, (2) Consider reusing existing ontologies, (3) Enumerate important terms in the ontology, (4) Define the classes and the class hierarchy, (5) Define the properties of classes' slots, (6) Define the facets of the slots, and (7) Create instances (Noy, McGuinness, 2001).

A number of ontology representation languages are available due to the efforts of AI specialists and the World Wide Web Consortium (W3C). Most of the later ones were developed to support ontology representation on the Semantic Web (introduced in the Section 2.3.1), and hence they are also referred to as "Semantic Web languages", "Web-based ontology languages" and "ontology markup languages" (Gómez-Pérez, Corcho, 2002). Current Web-based ontology languages are all based on XML. XML (eXtensible Markup Language) enables the specification and markup of computer-readable documents. It looks very much like HTML in that special sequences of characters and tags are used to mark up the document content, and that XML data is stored as ordinary text. Unlike HTML, however, XML can be used to annotate documents of arbitrary structure, and there is no fixed tag vocabulary. Widely used Web-based ontology languages include:

- RDF (Resource Definition Framework), developed by the W3C as a semantic-network-based language to describe Web resources. RDF represents data about "things on the Web" (resources) is that of **O-A-V** triplets and semantic networks. A resource description in RDF is a list of statements (triplets), each expressed in terms of a Web resource (an **O**bject), one of its properties (**A**ttributes), and the **V**alue of the property (Manola, Miller, 2004).
- RDF Schema, also developed by the W3C, is an extension of RDF with frame-based primitives. The combination of both RDF and RDF Schema is known as RDF(S) (Brickley, Guha, 2004).
- DAML+OIL, having evolved from two DAML-ONT and OIL, both of which were heavily influenced by RDF(S). DAML-ONT was part of the DAML (DARPA Agent Markup Language) initiative, aimed at supporting the development of the Semantic Web. As an ontology language, it covered the capturing of definitions of terms-classes, subclasses, and their properties, as well as their restrictions and individual object descriptions (Hendler, McGuinness, 2000). OIL (Ontology Inference Layer) is based on description logics and includes frame-based representation primitives (Fensel, van Harmelen, Horrocks, McGuinness, Patel-Schneider, 2001).
- OWL (Web Ontology Language), a successor to DAML+OIL. Like its predecessors, the OWL vocabulary is built on top of the RDF(S) vocabulary, including a set of XML elements and attributes, with well-defined meanings. These are used to describe domain terms and their relationships in ontology (Smith, Welty, McGuinness, 2004). OWL is currently the most popular ontology representation language.

No ontology for product obsolescence exists in the literature, but many ontologies have been developed in the related area of product design and development. Among these ontologies, some focus on product information described with attributes and relationships. Vegetii et al. (2005) proposed PRoduct ONTOlogy (PRONTO), which defines concepts relations among them and axioms to be applied in the complex product modeling domain. It considers products with hybrid structures and concepts related to variant management. Li et al. (2008) established a product ontology model for product information organization to make organization and search of product information more efficient.

Some ontologies focus on the integration of product design and manufacturing activities. Kim et al. (2006) proposed a collaborative assembly design framework that offers a shared conceptualization of assembly design modeling. It encapsulates assembly knowledge about a product and has the flexibility to represent assembly constraints and spatial relationship among features, explicitly in a computer interpretable manner. However, Kim's assembly design ontology focuses on the geometric relations among parts and lacks other information such as manufacturing process and cost. Based on an assembly design ontology, Chang et al. (2008, 2009) developed a Design for Manufacturing (DFM) ontology in order to facilitate manufacturing process design with the consideration of cost, feasibility, available, and so on. They also proposed an ontology-based approach for knowledge management working along with the graphical modeling tool Integrated DEFinition (IDEF) to support designers at the conceptual design stage of product (Chang, Sahin, Terpenny, 2008). It helps designers to generate design concepts flexibly, fast and easily. There are also many other ontologies related to product design and development such as the product requirement design ontology defined by Jiang et al. (2006). Product requirement design is the key to success for a new product in the market. Requirement ontology provides compulsory design consideration and evaluation conceptual criterion for conceptual design. Defined ontologies in the domain of product design can be extended and reused for product obsolescence. To construct ontologies related to product obsolescence, some preliminary work focuses on a taxonomy and defining the evaluation criteria for DMSMS tools, databases and services (Sandborn, Jung, Wong, Becket, 2007). This work provides taxonomy and evaluation criteria, further described in Chapter 6.

2.2.2 UML

The UML (Unified Modeling Language) has been used to specify, visualize, modify, construct, and document the artifacts of an object-oriented software-intensive system (FOLDOC, 2001). UML is a result of best practices in engineering modeling and has been successfully proven in modeling many big and complex systems.

UML has features for emphasizing specific views of a model by using graphical representations of models, namely UML diagrams. In this way, models can be abstracted;

otherwise, complex systems may not be able to be analyzed and solved. UML defines the following diagrams for various views of models:

- Use Case Diagram
- Class Diagram
- Behavior Diagrams including Statechart Diagram and Activity Diagram
- Interaction Diagrams including Sequence Diagram and Collaboration Diagram
- Implementation Diagrams including Component Diagram and Deployment Diagram

These diagrams provide developers with various perspectives of the system under study or development. A model which captures a whole system and is graphically represented by diagrams just integrates all these perspectives into one common entity, comprising the union of all modeled details of that system.

UML combines techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, across different implementation technologies (Mishra, 1997). UML models can be automatically transformed to other representations by means of transformation languages. UML can also be extensible with two mechanisms for customization: profiles and stereotypes.

2.2.3 MDA for Ontology Development

The MDA (Model Driven Architecture) of the OMG (Object Management Group) is based on a few complementary standards and a four-layer metamodeling architecture (Figure 2-3) (Gašević, Djurić, Devedzić, 2006). These complementary standards are:

- the Meta-Object Facility (MOF);
- the Unified Modeling Language (UML);
- the XML Metadata Interchange (XMI).

The four –layer metamodeling architecture include:

- M3 layer: topmost level of the architecture, metametamodels, i.e., the MOF (Meta-Object Facility). It is an abstract, self-defined language and framework for specifying, constructing, and managing technologically independent metamodels. It is a basis for defining any modeling language, e.g., UML or the MOF itself.
- M2 layer: metametals, standard and custom (user-defined), e.g., UML defined with metametamodel belonging to M3 layer.
- M1 layer: models of the real world, e.g., UML models represented by concepts from a metamodel belonging to the M2 layer.
- M0 layer: things from the real world, modeled in the M1 layer.

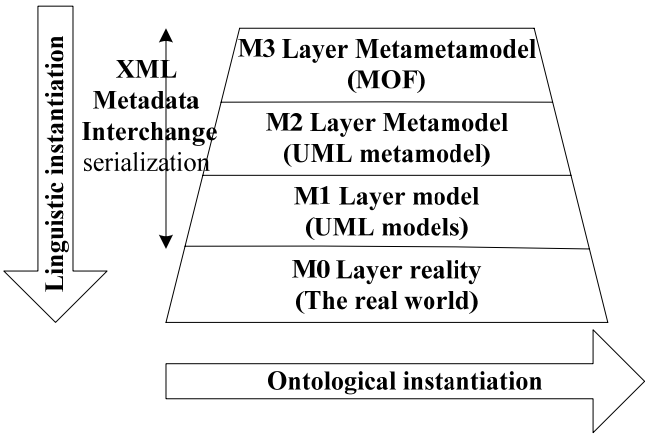


Figure 2-3: Model Driven Architecture (MDA)

To enable UML to support ontology development, the MDA for ontology development is proposed by OMG. The following components are defined in the MDA for ontology development for bridging the gap between ontology and UML (Figure 2-4) (OMG ODM RFP, 2003):

- ODM (Ontology Definition Metamodel): designed to include the common concepts of ontologies.
- Ontology UML Profile: a UML Profile that supports UML notation for ontology definition.

- Two-way mapping between OWL and the ODM, the ODM and other metamodels, the ODM and the Ontology UML Profile and from the Ontology UML Profile to other UML profiles (Gašević, Djurić, Devedzić, Damjanović, 2004).

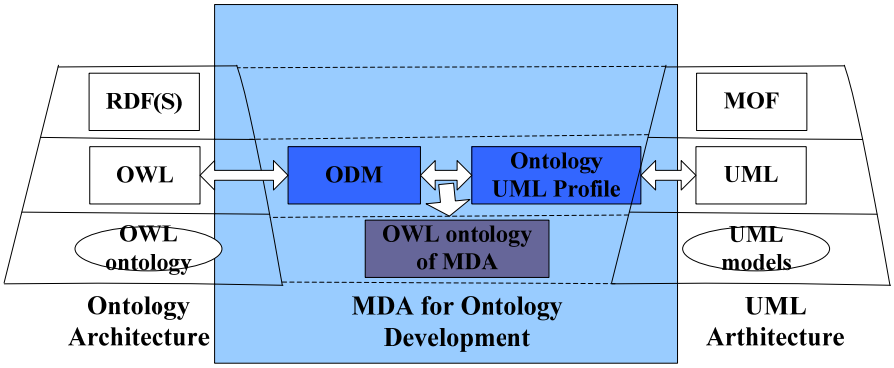


Figure 2-4: MDA for ontology development

MDA for ontology development provides a theoretic basis for ontology development with UML to facilitate evolution, updating and maintenance of built ontologies.

2.3 Information Integration

Data conflict in the DMSMS obsolescence management tools discussed in Section 2.1.3 can be classified as the problem of data heterogeneity. Heterogeneity problems are already well known within the distributed information systems community (Kim, Seo, 1991; Kashyap, Sheth, 1997). In general, heterogeneity problems can be divided into three categories:

- Syntax (data format heterogeneity): For instance, the format of a component's obsolescence date in Tool A is January, 1, 2011; however in Tool B it is 01/01/2011 (mm/dd/yyyy).
- Structure (homonyms, synonyms or different attributes in database tables): For instance, a component's original equipment manufacturer (OEM) information and introduction date are in the same data table of Tool A; However, they are in the separate data tables of Tool B.

- Semantics (intended meaning of terms in a special context or application): For instance, Tool A uses components while Tool B uses parts to represent the items that form the product.

This research will focus on the problem of semantic heterogeneity, because sophisticated solutions to syntactic heterogeneity and semantic heterogeneity have been developed (Stuckenschmidt, van Harmelen, 2005). Semantic Web technologies in ontology integration and collaboration play a key role in integrating, retrieving and utilizing distributed heterogeneous data and are applied to solve semantic heterogeneity.

In this section, basic concepts of Semantic Web are introduced first (Section 2.3.1). OWL, as a Semantic Web language, will be used to build ontologies for information integration. Then some important knowledge of OWL is introduced (Section 2.3.2). Finally, three types of ontology-based information integration approaches are described: single-ontology, multiple-ontology and hybrid approaches (Section 2.3.3).

2.3.1 Semantic Web

The main purpose of the Semantic Web is driving the evolution of the current Web by reaching its full use potential, thus allowing users to find, share, and combine information more easily. The current Web is an immense, practically unlimited source of information and knowledge that everyone can use, but it is not smart enough to easily integrate all of those numerous pieces of information from the Web that users really need. Integration for semantic heterogeneity, at a high, user-oriented level, is desirable in nearly all uses of the Web. Today, most of the Web information is represented in natural-language; however, computers cannot understand and interpret its meaning. Humans themselves can process only a tiny fraction of information available on the Web, and would benefit enormously if they could turn to machines for help in processing and analyzing the Web contents (Noy, McGuinness, 2001). Unfortunately, the Web was built for human consumption, not for machine consumption—although everything on the Web is machine-readable, it is not machine-understandable (Lassila, 1998). The Semantic Web is needed to express information in a precise, machine-interpretable form, ready for agents to process, share, and reuse it, as well as to understand what the terms describing the data mean. That

would enable Web-based application to interoperate both the syntactic and the semantic level (Gašević, Djurić, Devedzić, 2006).

The Semantic Web is a "web of data" that enables machines to understand the semantics, or meaning, of information on the World Wide Web. It extends the network of hyperlinked human readable web pages by inserting machine-readable metadata about pages and how they are related to each other, enabling automated agents to access the Web more intelligently and perform tasks on behalf of users (Berners-Lee, Hendler, Lassila, 2001).

The Semantic Web comprises the standards and tools of XML, XML Schema, RDF, RDF Schema and OWL that are organized in the Semantic Web Stack (Figure 2-5). Most components in the Semantic Web Stack have been described in Section 2.2.1 except URI. URI (Uniform Resource Identifier) identifies resources on the web and uses a global naming convention.

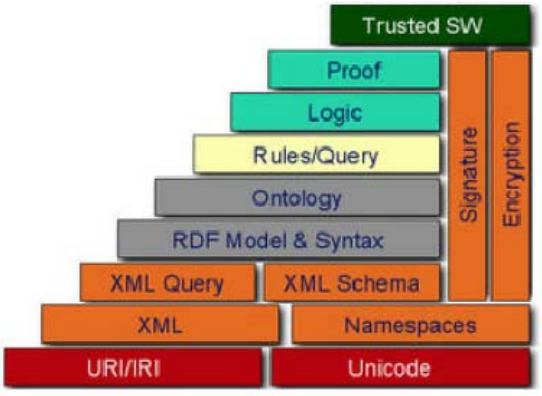


Figure 2-5: Semantic Web Stack
(http://commons.wikimedia.org/wiki/File:W3c_semantic_web_stack.jpg)

2.3.2 OWL

OWL (Web Ontology Language) is built upon W3C Standards XML, RDF and RDFS, and extends them to express class properties (Brickley, Guha, 2004; Lassila and Swick, 1999). It incorporates lessons learned from the design and application of DAML +OIL and is a revision of the DAML +OIL (Horrocks, 2002). XML version of an OWL file

example is shown in Figure 2-6. OWL, as a Semantic Web language, will be applied to build ontologies for the purpose of information integration.

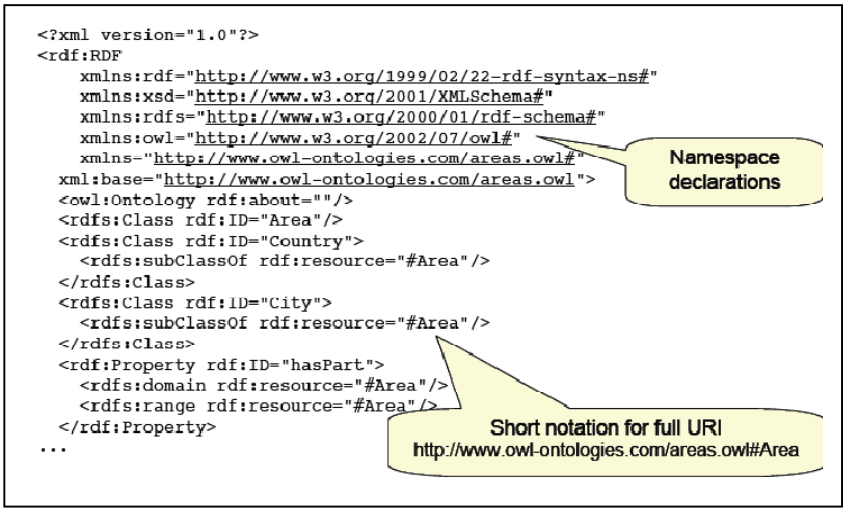


Figure 2-6: XML version of OWL file example

OWL is powerful in logic description. It can not only create a class and assign a name as most ontology languages do, but also build up anonymous classes from logic descriptions. Restrictions like existential restrictions and universal restrictions and logic operators are often used to describe classes:

- Existential restrictions, denoted by 'some', describe classes of instances that participate in a least one relationship along a specified property to a specified class.
- Universal restrictions, denoted by 'only', describe classes of instances that for a given property only have relationships along this property to a specified class.
- Logic operators like 'and' and 'or' cement simple class descriptions together to build up complex class descriptions (Horridge, 2009).

Consider the following example, based on descriptions of pizza. The class example described with restrictions is like the VegetarianPizza class of Figure 2-7. Vegetarian pizza is a kind of pizza which has cheese topping or vegetable topping or both, but cannot have other toppings besides these two. The expression of VegetarianPizza class satisfies the necessary and sufficient condition. OWL applies the necessary and sufficient condition to define classes so that any instance that satisfies the condition can be inferred

to be a member of the class that the condition is on. OWL also has other ways of describing classes including intersection, union, complement and enumeration. And complex class descriptions can be built up on the combinations of simple classes.

hasTopping **some** CheeseTopping
hasTopping **some** VegetableTopping
hasTopping **only** (CheeseTopping **or** VegetableTopping)

Figure 2-7: Description of class VegetarianPizza

2.3.3 Ontology-based Information Integration Approaches

Ontology provides an infrastructure for transforming the Web of information and data into the Web of knowledge—the Semantic Web. It plays an important role in information integration. Ontology-based integration approaches utilize ontology for the explicit description of the information-source semantics. But there are different ways of how to employ the ontologies. In general, three different directions can be identified: single-ontology approaches, multiple-ontology approaches and hybrid approaches, as shown in Figure 2-8 (Stuckenschmidt, van Harmelen, 2005).

In single-ontology approaches, a global ontology is used in order to provide a shared vocabulary for the specification of the semantics (Figure 2-8(a)). In multiple-ontology approaches, each part of the data is described by its local ontology (Figure 2-8(b)). To overcome the drawbacks of the single- or multiple-ontology approaches, hybrid approaches have been developed. In hybrid approaches, the semantics of each source is described by its own ontology. That is similar to multiple-ontology approaches. But in order to make the source ontologies comparable to each other they are built upon one global shared vocabulary (Figure 2-8(c)) (Stuckenschmidt, van Harmelen, 2005).

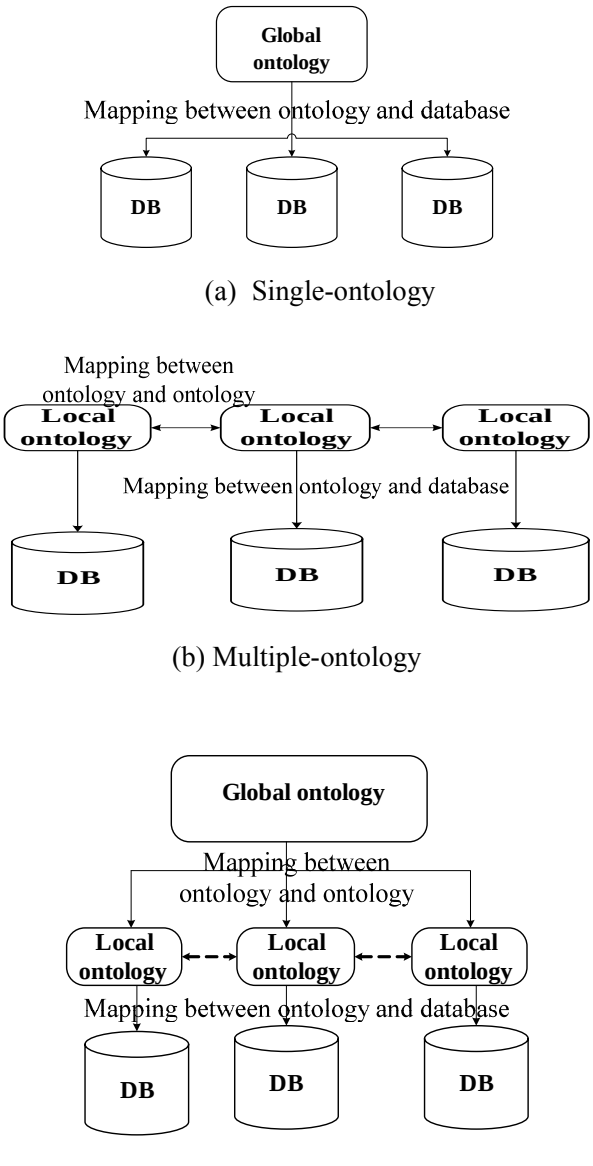


Figure 2-8: Three possible ways of ontology-based integration approaches

Three ontology-based integration approaches are compared in Table 2-1. The advantage of hybrid approaches is clear. Hybrid approaches can easily add new sources with the need of modification in the mappings or in the shared vocabulary (the global ontology). It also supports the acquisition and evolution of ontologies. The use of a shared vocabulary (the global ontology) makes the source ontologies comparable and avoids the disadvantages of multiple-ontology approaches. Hybrid approaches will be utilized in this

research in order to integrate heterogeneous data from existing tools discussed in Section 2.1.3 for managing DMSMS obsolescence related information.

Table 2-1: Comparison of ontology-based integration approaches (Stuckenschmidt, van Harmelen, 2005; Chang, 2008)

	Single-ontology approaches	**Multiple-ontology approaches**	**Hybrid approaches**
Implementation effort	Straight forward (shared data do not need to be rebuilt)	Costly (need some repeated work)	Reasonable
Semantic heterogeneity	Similar views of a domain	Supports heterogeneous views	Supports heterogeneous views
Adding/removing sources	Need for some adaption in the global ontology	Providing a new source ontology; relating to all other local ontologies	Providing a new local ontology and relating to the shared vocabulary
Comparing multiple ontologies	----	Difficult because of the lack of a common vocabulary	Simple because ontologies use a common vocabulary

2.4 Decision Support for Obsolescence Management

While DMSMS obsolescence knowledge is represented and integrated by using ontology, a decision support tool is also needed to support decision making in the process of DMSMS obsolescence management. The topic of decision support reveals a very broad area related to Decision Theory, Decision Analysis, Operations Research, Management Science, and Artificial Intelligence. For DMSMS obsolescence in particular, decision support needs to take into consideration three different types of obsolescence resolution strategies including: reactive, proactive and strategic management (Singh, Sandborn, 2006; Terpenny, Sandborn, Rai, 2009; Nelson III, Sandborn, 2010).

In this section, three types of DMSMS obsolescence resolutions are explained in detail. First, design refresh planning, an important way of realizing proactive and strategic obsolescence management is described in Section 2.4.1. Then an overview of models for decision making is provided. Mathematical programming is intended to be used to develop design fresh planning models for obsolescence management (Section 2.4.2).

2.4.1 DMSMS Obsolescence Resolutions

There are three types of resolutions to DMSMS obsolescence: reactive, proactive and strategic management. They are explained in detail in the following subsections. Traditionally, efforts to mitigate the effects of DMSMS have been reactive; that is, the effects are addressed only when they are observed. This reactive approach to DMSMS solutions leads to decisions that put a premium on faster solution paths with attractive short-term gains in order to avoid system inoperability, while ignoring the long-term solution paths that would lead to generic families of solutions or larger-scale solutions with the capacity of avoiding future DMSMS issues. In order to solve DMSMS issues with lower overall cost, DMSMS solutions must change from reactive to proactive and strategic (Livingston, 2000).

2.4.1.1 Reactive DMSMS Management

Reactive DMSMS management is concerned with determining an appropriate, immediate resolution to the problem of DMSMS obsolescence after it has occurred. Common reactive DMSMS management approaches include (Stogdill, 1999):

- Lifetime or last time buys: buying and storing enough parts to meet the system's forecasted lifetime requirements or requirements until a redesign is possible.
- Part substitution: using a different part with identical or similar form, fit, and function.
- Redesign: upgrading the system to make use of newer parts.
- Aftermarket sources: third parties that continue to provide the part after its manufacturer has obsoleted it.
- Emulation: using parts with identical form, fit, and function that are fabricated using newer technologies.
- Reclaim: using parts salvaged from other products.
- Uprating: using parts outside of their manufacturer specified environmental range (Wright, Humphrey, McCluskey, 1997). Uprating is used when the commercial part substitutes for the military-specification one.

2.4.1.2 Proactive DMSMS Management

Proactive DMSMS management identifies and manages DMSMS obsolescence before they actually happen. The realization of proactive management requires an ability to forecast obsolescence. The obsolescence dates of components obtained by forecasting are important inputs to proactive DMSMS obsolescence. Most DMSMS obsolescence forecasting is based on the development of models for the product life cycle. Proposed methods of life cycle forecasting include:

- Ordinal scale based methods: the life cycle stage of the product is determined from a combination of technological attributes (Henke, Lai, 1997; Josias, Terpenny, 2004).
- Product sales curve based methods: the life cycle curve of a product is obtained by fitting the sales data (Solomon, Sandborn, Pecht, 2000; Pecht, Solomon, Sandborn, Wilkinson, Das, 2002; Sandborn, Mauro, Knox, 2007).
- Leading indicator methods: a leading indicator product can be further identified in each life cycle pattern of products that provides advanced indication of changes in demand trends of products (Meixell, Wu, 2001).

Life cycle curves provide a basis for developing forecasts and planning for technology insertion and product update opportunities, and hence play an important role in proactive DMSMS management. The life cycle refers to the period from the product's launch into the market until its final withdrawal. The product normally passes through six stages: introduction, growth, maturity, saturation, decline, phase-out. Each stage is associated with several characteristics such as sales, price, usage and so on. Figure 2-9 presents a summary of the characteristics associated with each life cycle stage. Not surprisingly, not all products conform to the six life cycle stages. Some may undergo a false start and die out, and others may be revitalized after a niche market. The life cycle of these products may only consist of some of the six life cycle stages. The representation of obsolescence forecasting method used in this work is based on the product life cycle curve, similar to the one proposed by Solomon et al. (2000). The method defines life cycle stages and the zone of obsolescence for the products with the Gaussian distribution life cycle curve, and forecasts the life cycle stages and years to obsolescence by modeling the life cycle curve

based on the sales data. It captures market trends and makes good forecasting for obsolescence of products in the practical application.

Table 2-2: Characteristics of product life cycle stages (Livingston, 2000)

	Stage 1 Introduction	**Stage 2 Growth**	**Stage 3 Maturity**	**Stage 4 Saturation**	**Stage 5 Decline**	**Stage 6 Phase-Out**
Sales	Slow increase	Increasing rapidly	Stable	Leveling out	Decreasing	Lifetime buys offered
Price	Highest	Declining	Stable	Stable	Rising	High
Usage	Low	Increasing	Stable	Stable	Decreasing	Decreasing
Part modification	Frequent	Major	Periodic changes	Few	Few or none	None
Competitors	Few	High	Stable number	Decline begins	Declining	Declining
Manufacturer Profit	Low	Increasing	Stable	Stable	Reasonable for survivors	Reasonable for survivors (aftermarket)

2.4.1.3 Strategic DMSMS Management

Strategic DMSMS management tends to enable strategic planning, life cycle optimization, and long-term support for obsolescence susceptive systems using DMSMS data, logistics management inputs, technology forecasting and business trend. It covers reactive and proactive management. Two types of strategic planning approaches that exist include Material Risk Indices (MRI) and design refresh planning.

A Material Risk Indices (MRI) approach analyzes a product's bill of materials and scores a supplier-specific part within the context of the enterprise using the part. MRI are used to combine the risk prediction from obsolescence forecasting with organization-specific usage and supply chain knowledge in order to estimate the magnitude of sustainment dollars put at risk within a customer's organization by the part's obsolescence (Singh, Sandborn, 2006).

Design refresh[2-2] indicates an action of making changes to the system is driven by technology obsolescence. It is used to differentiate from redesign, which is driven by

[2-2] Design refresh is different from redesign. Design refresh is use as a reference to system changes that "Have to be done" in order for the system functionality to remain useable. Redesign

improvements in manufacturing, equipment or technology. Design refresh is technical terminology in the domain of obsolescence. Without design refresh, long field-life sustainment-dominated systems will be no longer producible and sustainable due to technology obsolescence. The aim of design refresh planning is to determine the optimal plan of design refreshes taken during the support life of the system including when to do design refresh and what obsolete/non-obsolete system components should be replaced at a specific design refresh. The optimal plan achieves the objective of minimizing life cycle cost of the system.

With the application of design refresh planning, strategic management for DMSMS obsolescence has such a timeline: Several design refreshes take place at some time points and divide the life cycle of the system into several time periods. During each time period between two design refreshes, when component obsolescence occurs, a lifetime buy of the component or short-term mitigation approaches (e.g., stock, last time buy, aftermarket source, etc) are applied on a component-specific basis until the next design refresh. When a design refresh is encountered, long-term mitigation solutions (e.g., substitute part, emulation, upgrade of similar part, etc) are applied until the end of the system life or possibly until some future design refresh. Because these long-time mitigation solutions may result in design change, requalification may be required (Figure 2-9).

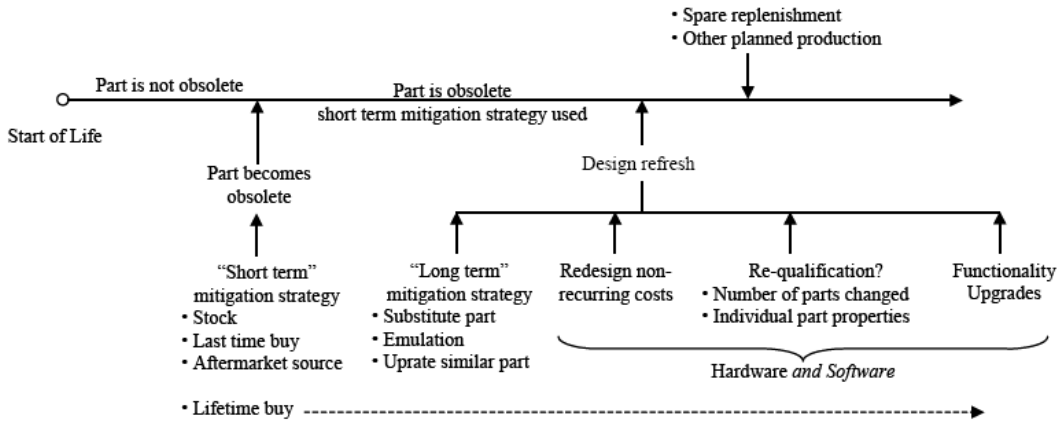


Figure 2-9: Design refresh planning analysis timeline (Singh, Sandborn, 2006)

is a term used to identify the "Want to be done" system changes, which include both the new technologies to accommodate system functional growth and new technologies to replace and improve the existing functionality of the system (Sandborn, Herald, Houston, Singh, 2003).

To determine the optimal design refresh planning, a model called MOCA (Mitigation of Obsolescence Cost Analysis) has been proposed by Singh et al. (2006), which is also one of a few existing solutions to strategic DMSMS management. The MOCA model uses a simulation approach. Inputs of the model are the bill of materials (BOM) information of the system including components' procurement costs and forecasted obsolescence dates, and the production plan. The model generates different candidate design refresh plans according to the production plan. The life cycle cost of the system for each design refresh plan is calculated through simulation, and the plan with the lowest life cycle cost is picked as the optimal design refresh plan. The output of the model is the optimal design refresh plan with the minimum life cycle cost, including time and content of each refresh in the plan. Although MOCA is a strategic tool to support DMSMS management, it does not solve the problem of how to integrate with other information systems.

2.4.2 Models for Decision Making

Quantitative models which support decision making are classified by the method or purpose for which they are applied. Operations Research models are typically grouped as Mathematical programming (or Optimization), Simulation, and Decision analysis models (Mitra, 1988). Others group models into two categories: Mathematical programming including Linear and Non-linear models, and Probabilistic including Queuing models, Markov chains, Simulation, and Decision analysis (Hillier, Lieberman, 1990). Decision models can also be classified into the following groups (Koutsoukis, Mitra, 2003):

- Mathematical programming: decision variables, constraints, objective function.
- Network flow: nodes, arcs, conservation of flow, node/arc attributes.
- Decision analysis: decisions, states of nature, outcomes, utility, main criteria, trees.
- Project management: activities with time and other attributes, precedence, critical path.
- Monte Carlo: computations with random variables, trials, outcome distributions.
- Inventory: stock, demands, replenishment.
- Queuing systems: arrivals, service, measures of congestion.

- Markov chains: state space, transition matrix, transient and long-term behavior, kinds of states.
- Static games: players, strategies, Nash equilibrium, stability.

The above model classification is a hybrid approach that captures simultaneously the underlying mathematical properties of the models, the corresponding solution methods, and application characteristics.

Decision making models are not always quantitative. Intuition, judgment, and heuristics are also applied in highly unstructured decision making circumstances.

2.4.2.1 Mathematical Programming

Mathematical programming (or Optimization) models represent fixed relations (equalities and inequalities) involving a number of linear and non-linear functions. The sets of equality and inequality restrictions are commonly referred to as constraints, and typically represent quantitative or logical restrictions of the real life problem. The solution of the set of equations is normally an extremum of an objective function, typically representing minimum cost or maximum profit. For example, a business problem considered is to find the optimal levels for performing a collection of i activities that require j resources, with minimum cost. Each activity i requires an amount a of resource j, leading to a_{ij}. There is a finite amount b of resource j available, leading to b_j. Each activity will be performed at the unknown level x per resource, leading to x_{ij}. There is a requirement for d amount of activity i leading to d_i. Each activity costs the company c per activity i per resource j leading to c_{ij}. The business requirement is to identify the activity levels x_{ij} for which the activity requirements are satisfied without exceeding the available resources, at minimum cost. The model formulation is as follows:

$$Minimize\ Cost = \sum_{ij} c_{ij}\, x_{ij}$$

$$Subject\ to: \sum_{j} x_{ij} \geq d_i$$

$$\sum_i a_{ij} x_{ij} \leq b_j$$

$$x_{ij} \geq 0, \forall\, i, j, i = 1 \cdots I, and\ j = 1 \cdots J.$$

Although profit and cost are typically used as objective functions, their optimum values are rarely the real object of optimization. That is, during the repeated modeling and solving cycles, analysts and decision makers seek to understand the behavior of the system modeled which leads to minimum cost or maximum profit. In this sense, profit and cost are, more often than not, used to indicate the optimization direction, rather than set a benchmark for decision making (Koutsoukis, Mitra, 2003).

2.5 System Development

A decision support system is a computer-based information system that supports decision making activities. It normally includes a knowledge-based system. A properly designed decision support system is an interactive software-based system intended to help decision makers compile useful information from a combination of raw data, documents, personal knowledge, or business models to identify and solve problems and make decisions. Three fundamental components of the system architecture are (Power, 2002; Koutsoukis, Mitra, 2003):

- the database (or knowledge base)
- the model(i.e., Decision models)
- the user interface

Obsolescence management system is a kind of decision support system.

2.5.1 Ontology Development Tool

An ontology development tool for knowledge representation and decision support for managing DMSMS obsolescence problem needs to be determined. From an ontology editor survey results (Denny, 2004), 94 ontology editors are compared in terms of 16 characteristics, such as release data, base language, web support, import/export format

and other characteristics. Among these editors, Protégé has more advantages than others. Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats including RDF(s), OWL, XML and Schema (http://protege.stanford.edu/). Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications. Many components providing interfaces to other knowledge-based tools such as Jess, Argenon, OIL and PAL Constraint, have been implemented and integrated in Protégé.

Protégé has a MOF compatible metamodel and is interoperability with MDA-based systems like UML. Its UML back end allows the exchange of UML models and ontologies between UML tools and Protégé based on MDA (Model Driven Architecture) for obsolescence domain ontology development discussed in Section 2.2.

Protégé supports two main ways of modeling ontologies via the Protégé-Frames and Protégé-OWL editors. The Protégé-Frame editor provides a full-fledged user interface and knowledge server to support users in constructing and storing frame-based domain ontologies, customizing data entry forms, and entering instance data. The Protégé-OWL editor enables users to build ontologies for the Semantic Web, in particular in the W3C's OWL (Web Ontology language). Therefore the Protégé-Frames can be used to generate instances to constitute obsolescence knowledge base and the Protégé-OWL can support OWL for obsolescence related information integration discussed in Section 2.3.

Protégé's SWRLTab is a development environment for working with SWRL rules. It provides a set of libraries that can be used in rules, including libraries to interoperate with XML documents, and spreadsheets, and libraries with mathematical, string, PDFS, and temporal operators. A SWRL-based OWL query language called SQWRL is also provided. Basic knowledge about SWRL rules and SQWRL queries are introduced in the next subsection. The SWRLTab currently requires the Jess rule engine to execute SWRL

rules or SQWRL queries. It has a plug-in called SWRLJessBridge that supports the execution of rules or queries using the Jess rule engine.

2.5.2 Ontology Rule and Query

SWRL (Semantic Web Rule Language) is a proposed Semantic Web rules-language, combining sublanguages of the OWL Web Ontology Language (OWL DL and Lite) with those of the Rule Markup Language (Unary/Binary Datalog). The SWRL rule is of the form of an implication between an antecedent and a consequent:

antecedent → consequent

Whenever the conditions specified in the antecedent hold (are true), then the conditions specified in the consequent must also hold (be true). An antecedent and a consequent are both conjunctions of atoms. Atoms are basic units that SWRL rules consist of. They can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is a class, P is a slot, and x, y are either variables, individuals or data values (W3C, 2004). For example, a SWRL rule representing that the brother of parents is uncle is:

parent(?x,?y) ∧ brother (?y, ?z)→uncle(?x,?z)

A more complex SWRL rule representing that a person whose age is greater that 17 is an adult is:

Person(?x) ∧ hasAge(?x,?age) ∧swrlb:greaterThan(?age,17) →Adult(?x)

Where *swrlb:greaterThan* is called a built-in for SWRL. It is built with a modular approach and can realize certain function. Examples of important built-ins for SWRL include:

- Built-ins for comparisons: *swrlb:equal, swrlb:lessThan, swrlb:greatThan,* etc.
- Math built-ins: *swrlb:add, swrlb:subtract, swrlb:multiply, swrlb:divide,* etc.
- Built-ins for Boolean values: *swrlb:booleanNot,* etc.
- Built-ins for strings: *swrlb:stringLength,* etc.

- Built-ins for date, time and duration: *swrlb:yearMonthDuration,* etc.
- Built-ins for URIs: *swrlb:resolveURI,* etc.
- Built-ins for lists: *swrlb:listConcat,* etc.

(Names of these built-ins indicate their functions.)

SQWRL (Semantic Query-enhanced Web Rule Language) is built on the SWRL rule language. SQWRL takes a standard SWRL rule antecedent and effectively treats it as a pattern specification for a query. It replaces the rule consequent with a retrieval specification. SQWRL uses SWRL's built-in facility as an extension point. Using built-ins, it defines a set of operators that can be used to construct retrieval specifications (O'Connor, Das, 2009). For example, people who are adults can be selected with SQWRL query in the second SWRL rule example:

Adult(?x) →sqwrl:select(?x)

It can also be merged with the second SWRL rule to be one SQWRL query:

Person(?x) ∧ hasAge(?x,?age) ∧swrlb:greaterThan(?age,17) → sqwrl:select(?x)

Examples of important built-ins for SQWRL query also include *sqwrl:orderby, sqwrl:count, sqwrl:avg, sqwrl:min, sqwrl:max, sqwrl:sum.* (Names of these built-ins indicate their functions.)

2.6 System Evaluation

System evaluation is an important part of system development. Evaluation[2-3] of the knowledge-based system (KBS) is concerned with the quality of the advice and decisions the system provides, the correctness of the reasoning techniques used, the quality of the human-computer interface, and the system efficiency. As the need to evaluate the performance and quality of knowledge-based systems has emerged, a large number of approaches and methods have been proposed, which have been summarized in the survey (anonymous, 1992). They can be classified into some categories (Guida, Mauri, 1993):

[2-3] Evaluation is different from assessment. Assessment of the system concerns about benefit and utility analysis, cost effectiveness, user acceptance, organizational impact, etc.

- **Knowledge-base oriented approaches**

A knowledge base is checked if it is *consistent*, *complete*, and *concise*. A knowledge base is:

  - *Consistent* if there is no way to derive a contraction from valid input data. Consistency checking aims at identifying such problems as conflicting knowledge, cyclic inference chains, ambiguities, etc;
  - *Complete* if it can cope with all possible situations that can arise in its domain. Completeness checking mainly aims at identifying problems derived from missing knowledge;
  - *Concise* if it does not contain any unnecessary or useless piece of knowledge.

These methods are derived from formal logic. They are based on a systematic check of the knowledge base through syntactic inspection and manipulation of rules and frames, interpreted as logic expression. Such checking methods are, in general, only capable of discovering potential problems in a knowledge base, but the mapping between knowledge base symbols and entities in the real world is not known to the checking procedure.

- **Approaches based on evaluation criteria**

The evaluation criteria are *validity*, *usability*, *reliability*, *effectiveness*:

  - *Validity* concerning correctness, quality, accuracy, and completeness of responses (e.g., solutions, decisions, advice, etc);
  - *Usability* concerning quality of human-computer interaction, understandability, explanation facilities;
  - *Reliability* including hardware and software reliability, robustness, sensitivity;
  - *Effectiveness* including cost-effectiveness (costs and benefits involved in solving a task), efficiency (both in terms of response time and use of computer resource), development time and cost, maintainability and extensibility.

These methods are derived from the experimental use of a knowledge-based system in controlled situations. They involve running a system (a prototype or the final target system) on selections of test cases, and assessing the results. This task is straight-forward if there is a reference standard against which system outputs can be compared. If not,

agreement methods can be used to measure the correspondence between the system problem solving behavior and the one of human experts.

- **Approaches based on evaluation methods**

These methods are based on empirical evaluation either qualitative or quantitative or both. Several methods are grounded on a concept of the system life cycle. Some of them are also based on evaluation criteria, while others include the adoption of knowledge-based checking methods.

Existing approaches and methods for knowledge-based system evaluation although proposing interesting ideas and methods, lack a sound foundation and are affected by several practical limitations (e.g., partial checking, weak criteria, only empirical principles, etc). In view of these problems, Guida et al. proposed an approach which is based on a certain sound foundation and has effective procedures to apply the methodology to real cases (Guida, Mauri, 1993). In this approach, evaluation is based on the procedure of knowledge-based system development and via verifying the actual system behavior versus specification (verification). The specification is defined according to requirements of users of the system. The structure of the approach is shown in Figure 2-10.

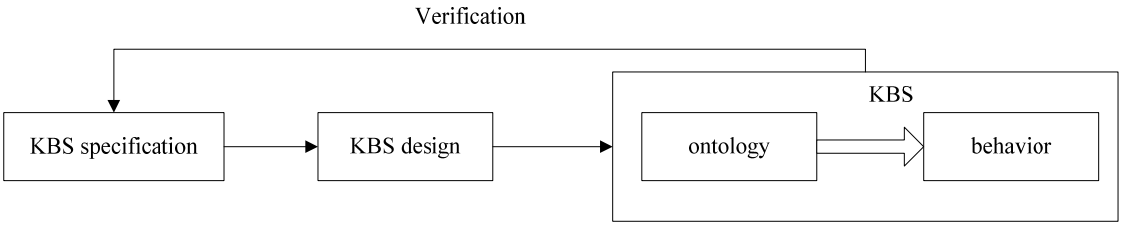


Figure 2-10: Structure of approach proposed by Guida et al.

The central task of knowledge-based system evaluation is to verify the actual system behavior versus specification. A set of criteria related to performance and quality need to be defined. In the domain of DMSMS obsolescence, Sandborn et al. have proposed a taxonomy for DMSMS tools data and services. The taxonomy can be expanded into a set of evaluation criteria useful to assess the DMSMS management tools. The taxonomy definitions for DMSMS tools, database, and services are as follows (Sandborn, Jung, Wong, Becker, 2007):

- Aggregation/collaboration environments: the environment with which the tools and databases that comprise the DMSMS management solutions reside
- Part data management: the tools and databases that collect and manage data that supports the analysis of DMSMS issues
- Part list monitoring: part status tracking and inventory tracking for lists of parts or individual bills of materials treated independently from the application context
- Platform/system analysis and management: the use of DMSMS data combined with platform/system descriptive and life cycle information to enable platform/system lifecycle management across multiple enterprise
- Strategic planning: the use of DMSMS data, logistics management inputs, and technology/business forecasting/trending to enable strategic planning, life cycle optimization, and long-term business case development and support

2.7 Conclusion

In this chapter, a comprehensive literature review of the product obsolescence problem and the methodologies and technologies for designing and evaluating a knowledge-based decision support system is provided. The serious consequence of DMSMS obsolescence and status quo of obsolescence management which lacks the knowledge representation scheme and only focuses on reactive management are stated. The potential approach and methodology that associates the background statement is provided. Ontology will be utilized to represent and manage obsolescence domain knowledge. Proactive and strategic obsolescence management strategies such as obsolescence forecasting and design refresh planning will be applied to support decision making. The details of the approach and methodology will be discussed in the following chapters.

Chapter 3 Knowledge Representation

This chapter provides the method and resultant obsolescence ontology developed for establishing a knowledge representation scheme for obsolescence information sharing and reuse. An overview of the method is presented in Section 3.1. The detail of each step in the method is introduced in Sections 3.2-3.6. The application of the method is demonstrated with two distinct examples. The result from applying each step of the method is also put forward in each section together with the method.

3.1 Overview of Approach

As described previously, current obsolescence management tools lack knowledge representation models that allow information sharing, reuse, and collaboration on obsolescence issues across different organizations. To establish a comprehensive knowledge representation scheme for managing obsolescence, ontology, as an explicit formal specification for sharing domain knowledge, is applied in this research. The key to successful ontology development is to start with a good design. UML (Unified Modeling Language) is utilized for obsolescence ontology development. Utilizing UML not only supports the capture and the visualization of the design during the ontology development phase, but also is used in updating the ontology in the maintenance phase.

For ontology development, the general methodology has been developed by Noy and McGuinness (2001). The steps of this methodology have been introduced in Section 2.2.1. Meanwhile, how to develop obsolescence ontology is also a question related to knowledge engineering. Terpenny et al. (2000) developed a methodology that can assist the discovery, classification, and capture of design knowledge (Terpenny, Strong, Wang, 2000). The process of obsolescence ontology development can make use of this general methodology, but has unique features. The process of ontology development for obsolescence knowledge representation put forward in this research is shown in Figure 3-1. The details of steps in the process are introduced in the sections that follow.

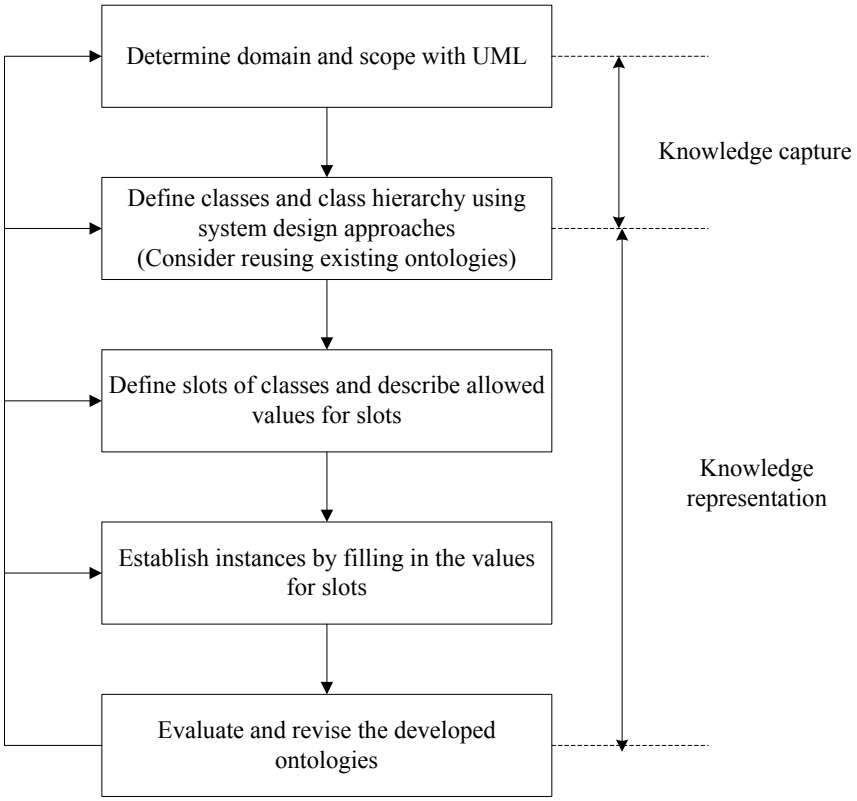


Figure 3-1: Flowchart of ontology development for obsolescence knowledge representation

3.2 Determine Domain and Scope

Determining the domain and scope of the ontology for obsolescence management is a process of knowledge discovery and capture. UML, a well-known software modeling language, is utilized for obsolescence ontology development based on a Model Driven Architecture (MDA). This was shown in previously in Figure 2-4, and introduced in Section 2.2.3. UML for obsolescence ontology development starts from the design of use cases. Use cases can identify a goal of the system. Then classes which support use cases can be enumerated. Use Case Diagrams and Class Diagrams are effective mechanisms for capturing the functional requirements of the system and discovering knowledge in the domain. They also support design, coding, testing, and validation activities during the system development.

Utilizing UML for obsolescence ontology development is best illustrated through an example. Figure 3-2 shows a use case diagram of DMSMS obsolescence management in the product production and maintenance process. Product managers inside the company

send orders for parts to part vendors. Part vendors outside the company normally fulfill orders if parts are in production. Once the part is out of production, part vendors send the notice of discontinuance. Part managers inside the company receive the notice of discontinuance and select obsolescence resolutions to solve the problem. This is referred to as reactive DMSMS obsolescence management (previously described in Section 2.4.1.1). To reduce the influence of obsolescence as much as possible, the company has technicians to do part procurement life cycle forecasting. Strategic managers can select parts which are not likely to become obsolete in the near future according to the forecasting of part procurement life cycles. This is referred to as proactive DMSMS obsolescence management (previously described in Section 2.4.1.2). There are also designers who do design refresh planning using the forecasting information. Then strategic managers can schedule design refreshes. If necessary, systems with some changes made by design refreshes will be re-qualified by strategic managers and product managers together. This is referred to as strategic DMSMS obsolescence management (previously described in Section 2.4.1.3). As expected, inventories of parts need to be monitored continuously and activities of different types of management are restricted by the budget of the company.

A class diagram which can support the above use cases of DMSMS obsolescence management in the product production and maintenance process is developed, as shown in Figure 3-3. Among these classes, Product, BOM, Part, Source, Inventory are related to product information, and Obsolescence forecasting, Obsolescence resolution, Design refresh, Proactive management, Obsolescence mitigation, Budget, Qualifying, Standard are related to obsolescence management. The former supports part status and inventory tracking, alternative and substitute part identification, notice collection, archiving and alerting, part procurement logistics. The later supports obsolescence risk and date forecasting, mitigation planning and management including mitigation approach identification, lifetime buy quantity determination, design refresh costing, re-engineering and emulation (Sandborn, Jung, Wong, Becker, 2007). Knowledge about obsolescence management along with UML diagrams provides the foundation for ontology development in following steps.

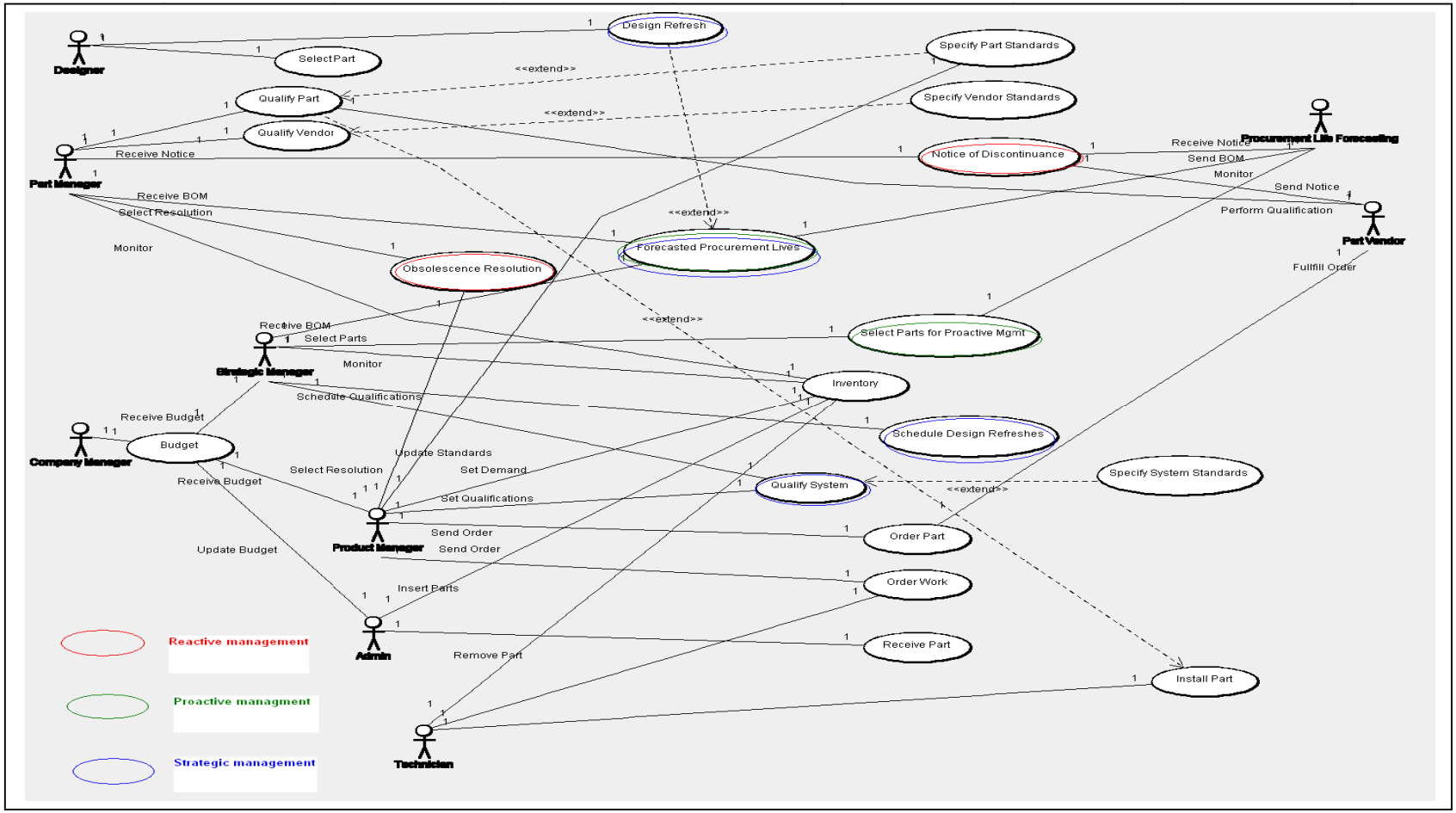


Figure 3-2: Use case diagram of DMSMS obsolescence management in product production and maintenance process

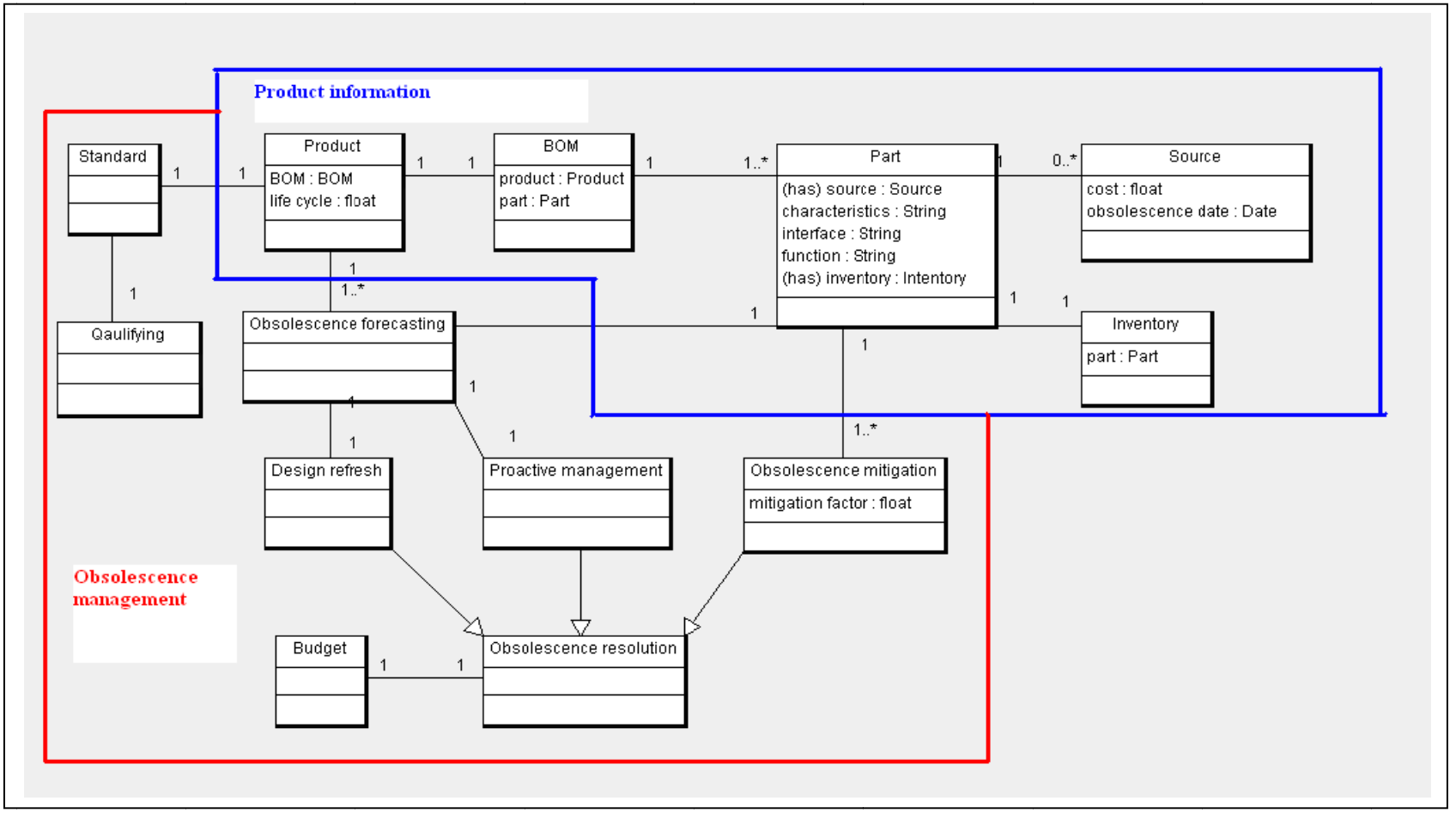


Figure 3-3: Class diagram supporting use cases of DMSMS obsolescence management

3.3 Define Classes and Class Hierarchy

The formal classes and the class hierarchy for knowledge representation and reuse in the domain of obsolescence can be generated from terms appearing in the UML diagrams. These classes need to represent concepts with some meanings in the real world. Overlap among classes should be as little as possible. There are three possible approaches in developing a class hierarchy: **top-down**, **bottom-up** and a **combination** approach (Uschold, Gruninger, 1996). A top-down development process starts with the definition of the most general class and subsequent specialization of the classes. A bottom-up development process starts with definition of the most specific classes with subsequent grouping of these classes into more general concepts. A combination development process is a combination of the top-down and bottom-up approaches (Noy, McGuiness, 2001). Terpenny has indicated the advantages and disadvantages of the top-down and bottom-up approaches from the view of synthesis, integration, knowledge capture, and reuse: the top-down approach satisfies the functional requirements, but may not lead to realizable solutions (i.e., real-world embodiments). While for the bottom-up approach, realizable solutions are implicit, but embodiments which meet functional requirements may not be assured (Terpenny, 1997). Terpenny also presented an integrated conceptual modeling system that can support the blended (combination) method, and demonstrated the utility of the system with an example design problem. The obsolescence class hierarchy is developed in a similar way by combining the top-down and bottom-up approaches together, as shown in Figure 3-4. A top-down approach is considered first. For the problem of obsolescence, the most frequently asked questions are: “What objects easily incur obsolescence?”, “What is the cause of obsolescence?”, “How to manage obsolescence?”, etc. And subclasses Type, Cause, Object, Management, Constraint are generated from the class Obsolescence according to these questions. In the opposite direction, there has already been a pool of classes from the UML diagrams in the previous step. Among these classes, the classes Product and Part can be subclasses of the class Object; the classes Reactive, Proactive, Strategic can be subclasses of the class

Management[3-1]; etc. This is a bottom-up process. Classes developed in opposite direction finally coverage by using the blended (combination) method.

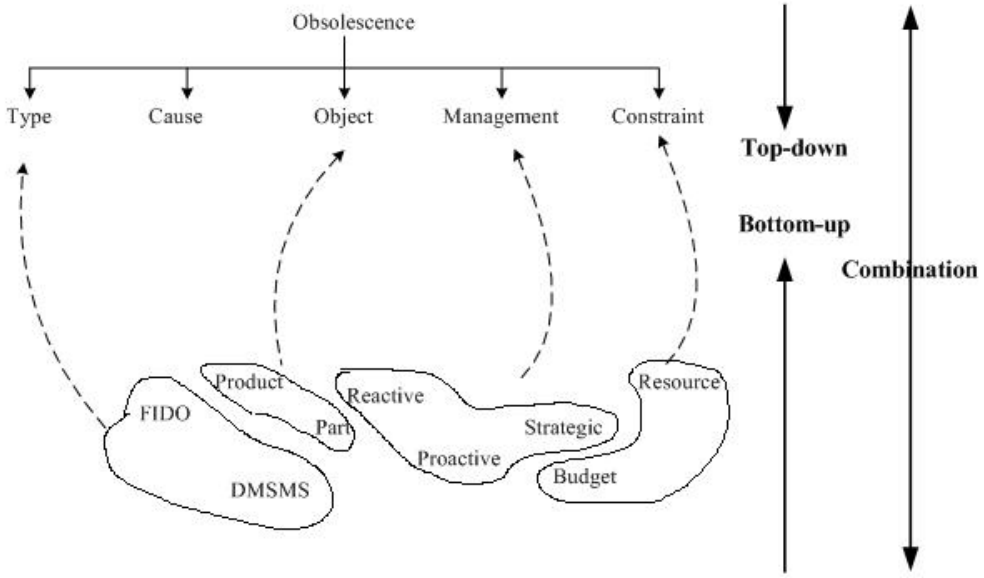


Figure 3-4: Process of obsolescence class hierarchy development

Objects affected by obsolescence can be quite varied and not limited to only products. For example, skill obsolescence is a kind of obsolescence related to workers in a job.

This research focuses on product obsolescence. The class hierarchy developed for product obsolescence is shown in Figure 3-5. Meanings of most classes have been introduced in the chapter of literature review. But relationships between Part, Assembly, and Component still need to be clarified. Parts here represent elements which build up a product. Part includes assembly and component. A component is a basic functional element that is not further divided, and an assembly consists of component[3-2]. Classes Component and Assembly can reuse the existing ontologies. See reference for other such

[3-1] The terms of Reactive management and Strategic management replace the terms of Obsolescence mitigation and Design refresh in the Class diagram supporting use cases of DMSMS obsolescence management for the purpose of formalization.

[3-2] The classification of a product to be a component or an assembly is determined based on the relationship between the product and its user. For example, for a computer manufacturer, a motherboard may be treated as one of components to assemble a computer. However, for a motherboard manufacturer, it is an assembly which is built with microprocessors, memories, driver controllers and so on.

example ontologies (the University of Massachusetts Amherst Center for e-Design). This is referred to as **ontology reuse**, an important way of ontology development.

It is important to note that there is no single correct definition of a class hierarchy. For example, System can also be used instead of Product. System and Product are equivalent in this research. Forecasting can be considered to be the subclass of Proactive management, because proactive management requires forecasting. If an obsolescence forecasting method is based on a product sales curve, class Product life cycle needs to be defined. This is referred to as **ontology extension**. For the developed class hierarchy, this research works on 'design refresh' strategic management restricted by 'temporal' constraint with input from 'product sales curve based' forecasting for 'DMSMS' type of 'product' obsolescence with 'involuntary' cause (e.g., loss of supply chain, loss of tech support, etc). Related classes have been marked with ellipses in Figure 3-5.

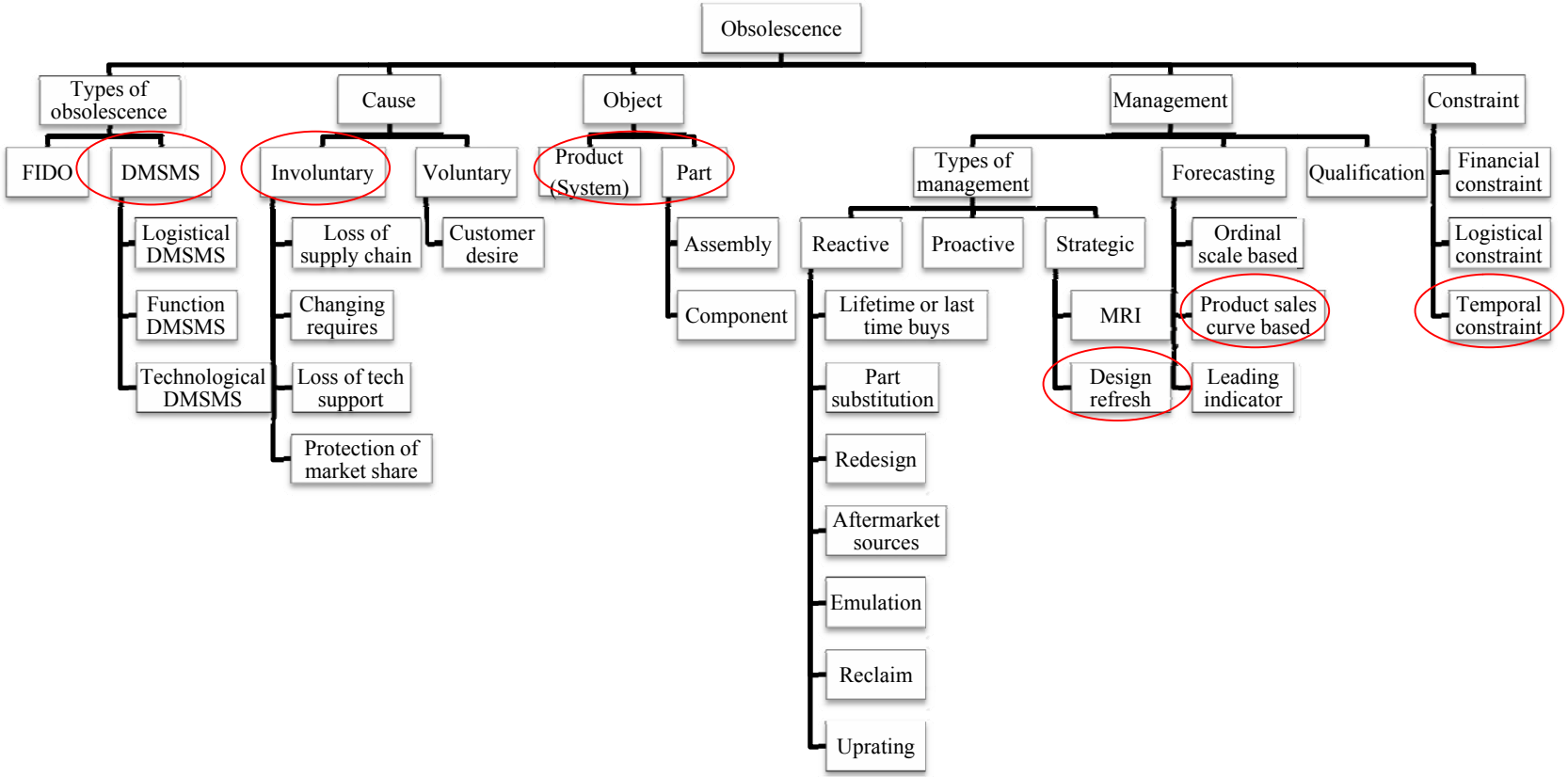


Figure 3-5: Class hierarchy for product obsolescence

3.4 Define Slots

Slots define the properties of classes. Some slots have data type value (e.g., int, float, Boolean, string, date, time, etc). Others have value which is another class. If a slot value is of type class, this slot represents a relationship which links two classes together. Important slots of classes and their value type related to product obsolescence are listed in Table 3-1. Obsolescence_date is an important slot of the class Object because when the object will go obsolete is important information input for obsolescence management. Cost is an important slot of the class Management because the objective of obsolescence management is to minimize the cost. Product and Part are the subclasses of Object, so they can have slot obsolescence_date which is inherited from the class Object. Product has a slot has_part whose value is class Part. The slot has_part link Product and Part together, which represents what parts exist in the product. This information can be obtained from the BOM (bill of materials) of the product. More information about the slot has_part will be introduced in the next chapter.

Important slots of the class Part include source, inventory, characteristics, interface, and function. Sources represent organizations where products come from and include original manufacturers, distributers, brokers, and so on. According to the definition of obsolescence, the part is obsolete if it can no longer be procured from original manufacturers. Inventory can give the signal of obsolescence. When obsolescence occurs (original manufacturers are lost), long field-life product sectors often must resort to obtaining parts from brokers and other aftermarket sources that may or may not be approved by the original manufacturer. Characteristics indicate dimensional and material properties of an object that are generally measurable, such as form and fit. Characteristics, interface, and functions are important slots for seeking the substitution of the obsolete part. Only the substitutions with identical or similar slots can be used.

More slots can be defined in the same way if they are necessary.

Table 3-1: Important slots of classes related to product obsolescence

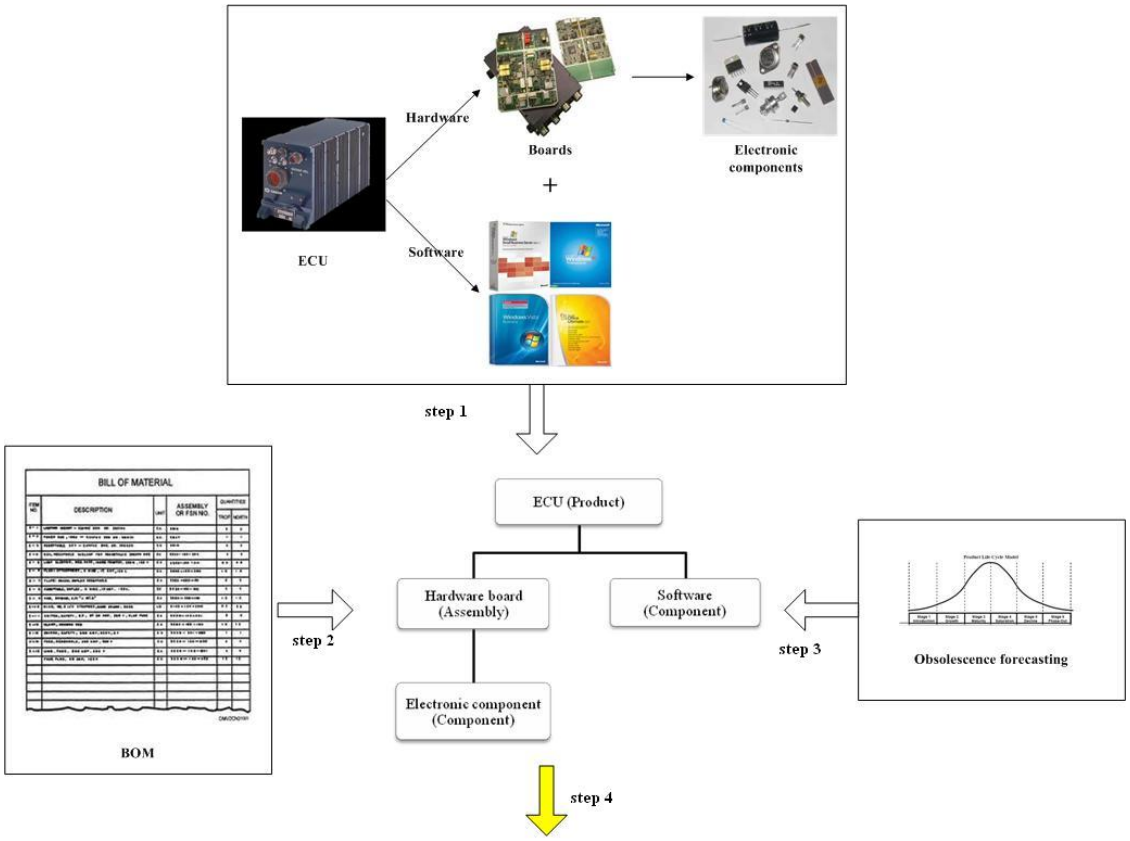
Class	Slot	Value type
Object	obsolescence_date	date
Management	cost	float
Product	has_part	class Part
Part	source	string
	inventory	float
	characteristics	string
	interface	string
	function	string

3.5 Establish Instances

Instances of classes can be created by specifying values of slots for the application. The example used for demonstration of the application of the developed ontology framework for product obsolescence is an electronic engine control unit (ECU) produced by a major automation and control company. The ECU is a device that controls the engine for aircrafts to produce a determined amount of thrust by monitoring engine parameters. The background information of the ECU is introduced in Appendix A. The process of establishing ontology instances for the ECU is shown in Figure 3-6. In the first step, a two-level product structure is identified from the configuration of the ECU. The ECU consists of three hardware boards (named I/O, CPU, EMI) populated with hundreds of electronic components (e.g., microprocessor, memory, passive components, etc), and several software systems or applications (e.g., operating system, control application software, data management software, etc) operating the hardware. Parts in level-1 of product structure include hardware boards and software. Parts in level-2 are electronic components which boards consist of. The ECU is an instance of the class Product, boards are instances of the class Assembly, and software and electronic components are instances of the class Component. In the second step, detail information about parts can be obtained from the bill of materials for filling in the slots of classes (e.g., source, inventory, characteristics, interface, function, etc). In the third step, parts may be obsolete during the life cycle of the ECU because of loss of supply or tech support. Life time and obsolescence dates of parts are predicted by using some methods like obsolescence forecasting. Finally, instances of classes can be created. The typical example is the class Part which may have millions of instances for managing part information. Table 3-2

shows some instances of the class Part. They are similar to entities which appear in the rows of the database table. This makes the ontology model very useful in standardizing the format of a database.

Obsolescence management can be realized based on the knowledge representation for the ECU. The ECU consists of hardware and software parts. Management for hardware and software obsolescence is restricted by temporal constraints, and is different. Hardware can continuously be used in the system even when obsolete, while software is not allowed to be used in the system when it is obsolete because the system may be at some security risk if it continues being operated with obsolete software. Obsolete hardware does not need to be replaced immediately until the next design refresh, but obsolete software has to be updated or replaced as soon as possible. Strategically managing hardware and software obsolescence in the ECU, applying the design refresh planning, will be introduced in detail in Chapter 5.



ID	name	quantity	source	cost($)	obs date	life time
part_4	Microcircuit	3	Vishay	2.9	2004.4	10
part_5	Microcircuit	1	Microchip technology	27.98	2004.05	10
part_51	Operating system	1	Microsoft	10.9	2006.8	10

Figure 3-6: Process of establishing ontology instances for ECU

Table 3-2: Some instances of class Part

ID	name	quantity	source	cost($)	obs date	life time
part_4	Microcircuit	3	Vishay	2.9	2004.4	10
part_5	Microcircuit	1	Microchip technology	27.98	2004.05	10
part_51	Operating system	1	Microsoft	10.9	2006.8	10

3.6 Evaluate Developed Ontologies

In this section, the generality of the developed obsolescence ontology is evaluated for its ability to represent multiple types of obsolescence problems. The problem of skill obsolescence is used for this evaluation. Skill obsolescence occurs with the depreciation (loss) of workers' skill because of various developments in society. Some examples

include the loss of manual skills due to ageing, and the introduction of computer in an office environment decreasing the value of basic typing skill of workers. Skill obsolescence can create a high risk of unemployment for workers and present problems to operations of companies due to the lack of workers with required skills. Skill obsolescence is a very different form of obsolescence than product obsolescence. The 'object' vulnerable to obsolescence in skill obsolescence is the worker.

The class hierarchy for skill obsolescence can be readily generated in a little time by using the developed obsolescence ontology framework. It is shown in Figure 3-7. Two main types of skill obsolescence have been distinguished: technical and economic skill obsolescence. Technical skill obsolescence is due to changes which originate in workers. The loss of manual skills due to ageing is an example of this. Economic skill obsolescence is caused by changes in the job or work environment (Van Loo, De Grip, De Steur, 2001). The introduction of computer in an office environment decreasing the value of basic typing skills of workers is an example of this. Types of skill obsolescence can be further subdivided. See Table 3-3 for several types of skill obsolescence and their descriptions. Different types of skill obsolescence are caused by different risk factors. For example, in the case of wear, taxing labor conditions are important risk factors. Job-specific skill obsolescence occurs when job content changes due to various developments like organizational developments and new technologies. Market-specific skill obsolescence occurs when employment in some occupation shrinks, forcing workers to move to another occupation. In this process, they may lose part of their human capital. In addition, workers may be forced to change firms, leading to company-specific skill obsolescence (Van Loo, De Grip, De Steur, 2001). These causes of skill obsolescence have been identified in the class hierarchy in Figure 3-7. The objects suffered from skill obsolescence are organizations and workers, where organizations consist of workers. Skill obsolescence management includes keeping mobility and functional flexibility, and training. Mobility enables employees to stay employable by expanding their experience base. Training helps workers to update their skills. And function flexibility means workers can perform tasks that are not part of their jobs (Hyatt, 1995; Bishop, 1997; Rajan, 1996; Thijssen, 1997, Van Loo et al., 2001). In addition, forecasting is needed in

skill obsolescence management. Qualifying new skills is required while management is restricted by financial, logistical and temporal constraints. This part of knowledge is shown in the class hierarchy of Figure 3-7.

Table 3-3: Types of skill obsolescence (Neumann et al., 1995; De Grip et al., 1997; Van Loo et al., 2001)

Type		Description
Technical	Wear	Natural ageing process, illness, or injury
	Atrophy	Lack or insufficient use of skills
Economic	Job-specific	New skill requirements for the job due to developments in society
	Market-specific	Shrinking employment in occupation or developments in economic sector
	Company-specific	External mobility

Slots can be defined for classes. For example, slots of the class Worker include job category, salary, date of demission, work life time. Instances can be created by specifying values of slots. Some instances of the class Worker for an IT company are shown in Figure 3-4. These are similar to the instances of the class Part, previously described in the product obsolescence example. For the case of the skill obsolescence ontology, it can be a useful tool to manage the personnel database of an organization.

Table 3-4: Some instances of class Worker

ID	name	gender	job category	cost($)	date of demission	work life time
103	Hanes Mary	M	architect design	72,000	2004.5	10
201	Foster Linda	F	software testing	38,000	2012.3	15
404	Gao Anthony	M	software development	54,000	2006.8	10

3.7 Conclusion

In this chapter, a systematic method for developing an ontology for domain knowledge capture and representation is presented. The developed obsolescence ontology can also be used in the hybrid approach for integrating heterogeneous obsolescence information in the next chapter. This method can be utilized for establishing an obsolescence knowledge scheme, and also be extended to develop ontologies for other domain problems.

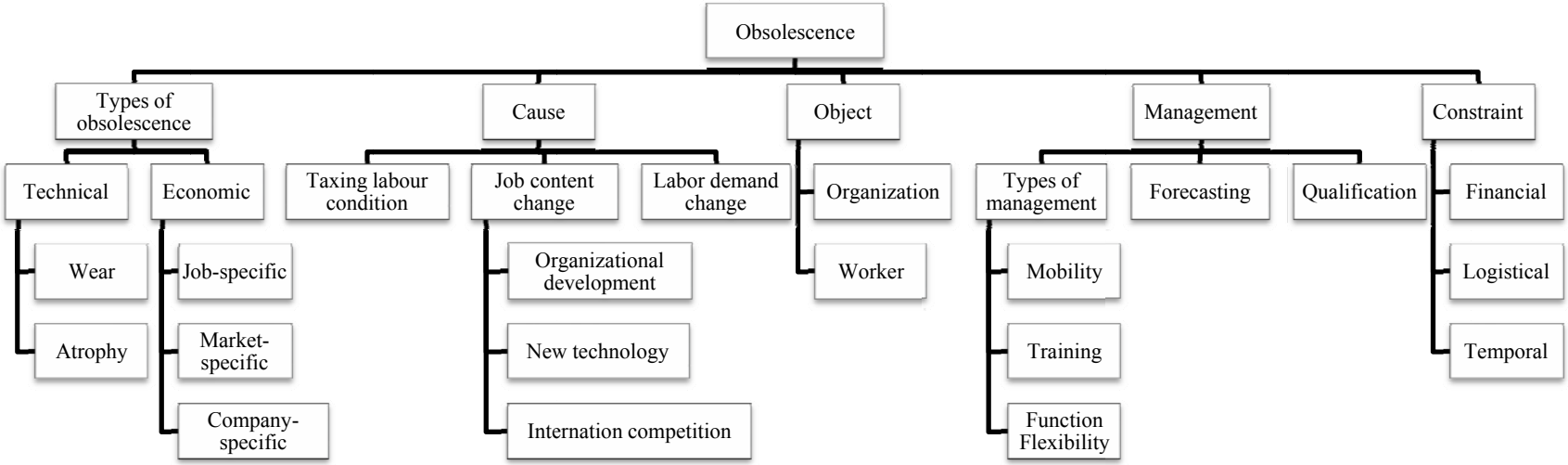


Figure 3-7: Class hierarchy for skill obsolescence

Chapter 4 Information Integration

This chapter provides the method and results of applying the ontology-based hybrid approach for integrating heterogeneous data sources. The overview of the procedure for applying the approach is presented in Section 4.1. The detail of each step in the procedure is described in Sections 4.2-4.8. Three data sources (from a relational database, an object-oriented database, and a file) are provided as examples of heterogeneous data sources. The results of incorporating these varied data sources into the obsolescence knowledge representation scheme using the proposed approach is presented at the end of the chapter.

4.1 Overview of Approach

Semantic Web is currently the most effective way of solving the problem of semantic heterogeneity existing in web information, as introduced in Section 2.3.1. OWL (Web Ontology Language) is a popular Semantic Web language that is used in the development of obsolescence ontology for the purpose of information integration. Using OWL can eliminate semantic heterogeneity in the new information system. But OWL alone is not sufficient to deal with existing heterogeneous data sources. To integrate information from existing heterogeneous data sources, three types of ontology-based information integration approaches (single-ontology approaches, multiple-ontology approaches, hybrid approaches) were compared in Section 2.3.3, and hybrid approaches were selected. With hybrid approaches, local ontologies are used to describe each data source, and the global ontology is in charge of local ontologies (Figure 2-8(c)). To apply hybrid approaches, local ontologies and the global ontology are built with OWL for Semantic Web integration. Relationships between the global ontology and local ontologies are quantified and identified. Bergamaschi et al. have proposed a semantic approach to information integration which takes into account semantic conflicts and contradictions, caused by the lack of a common shared ontology. In the method, Description Logic and clustering techniques are exploited. Description Logic is used to reason about the validity of terminological relationships in the Thesaurus and to infer new relationships out of a set of explicit ones in an automated way. Clustering techniques allow the automated identification of schema classes in different sources schemas that are candidates to be

unified in the global schema (Bergamaschi, Castano, De Capitani di Vimercati, Montanari, Vincini, 1998). Based on the Bergamaschi et al.'s method, the procedure of applying hybrid approaches for information integration has been developed, as shown in Figure 4-1. Clustering techniques are also used to identify relationships between classes. The details of steps in the procedure will be introduced in the sections that follow.

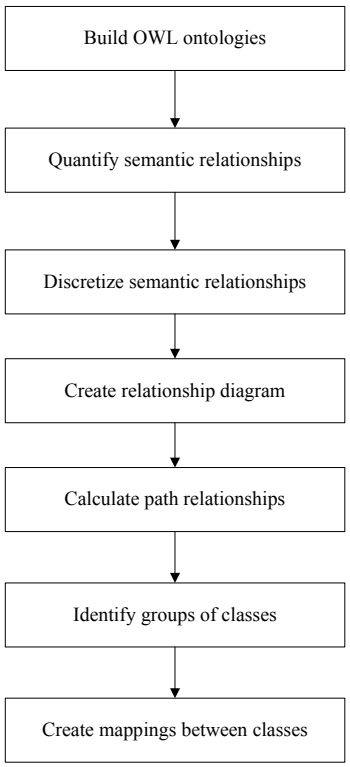
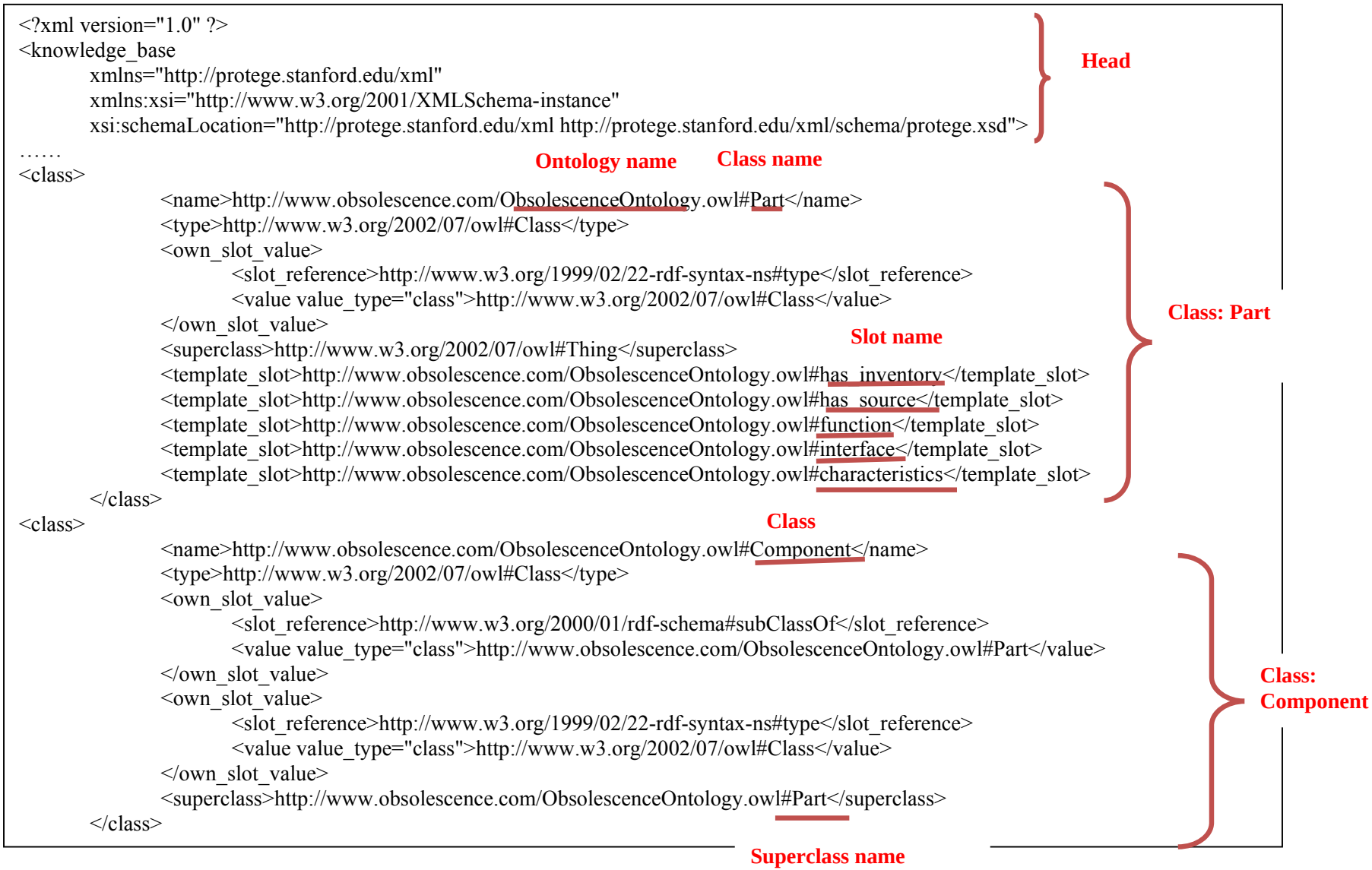


Figure 4-1: Procedure of applying hybrid approaches for information integration

4.2 Build OWL Ontologies

The global ontology and local ontologies for Semantic Web integration are built with OWL. XML (eXtensible Markup Language) is well suited for representing the obsolescence ontology. XML enables the specification and markup of computer-readable documents. Figure 4-2 shows a portion of XML of the obsolescence ontology built using OWL. In this case, the class Part and its subclass Component are defined. Additional information such as the file name and the path name appearing in the form of the URL is also included.


<?xml version="1.0" ?>
<knowledge_base
xmlns="http://protege.stanford.edu/xml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://protege.stanford.edu/xml http://protege.stanford.edu/xml/schema/protege.xsd">
Head
……
Ontology name
Class name
<class>
<name>http://www.obsolescence.com/ObsolescenceOntology.owl#Part</name>
<type>http://www.w3.org/2002/07/owl#Class</type>
<own_slot_value>
<slot_reference>http://www.w3.org/1999/02/22-rdf-syntax-ns#type</slot_reference>
<value value_type="class">http://www.w3.org/2002/07/owl#Class</value>
</own_slot_value>
Slot name
<superclass>http://www.w3.org/2002/07/owl#Thing</superclass>
<template_slot>http://www.obsolescence.com/ObsolescenceOntology.owl#has_inventory</template_slot>
<template_slot>http://www.obsolescence.com/ObsolescenceOntology.owl#has_source</template_slot>
<template_slot>http://www.obsolescence.com/ObsolescenceOntology.owl#function</template_slot>
<template_slot>http://www.obsolescence.com/ObsolescenceOntology.owl#interface</template_slot>
<template_slot>http://www.obsolescence.com/ObsolescenceOntology.owl#characteristics</template_slot>
Class: Part
</class>
<class>
Class
<name>http://www.obsolescence.com/ObsolescenceOntology.owl#Component</name>
<type>http://www.w3.org/2002/07/owl#Class</type>
<own_slot_value>
<slot_reference>http://www.w3.org/2000/01/rdf-schema#subClassOf</slot_reference>
<value value_type="class">http://www.obsolescence.com/ObsolescenceOntology.owl#Part</value>
</own_slot_value>
<own_slot_value>
<slot_reference>http://www.w3.org/1999/02/22-rdf-syntax-ns#type</slot_reference>
<value value_type="class">http://www.w3.org/2002/07/owl#Class</value>
</own_slot_value>
<superclass>http://www.obsolescence.com/ObsolescenceOntology.owl#Part</superclass>
Class: Component
</class>
Superclass name

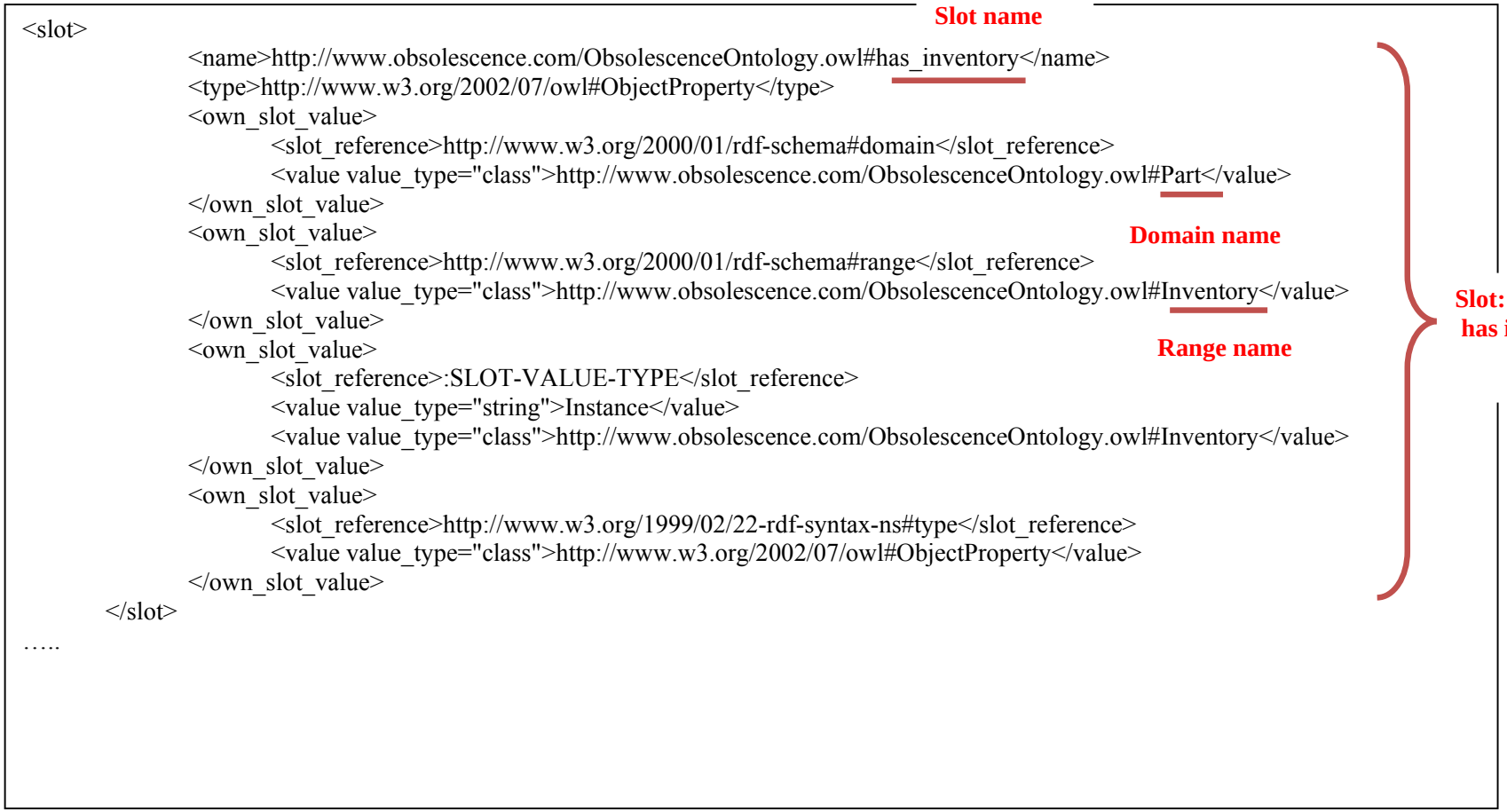


Figure 4-2: XML version of obsolescence ontology built with OWL

OWL is a powerful tool for logic description. It makes use of restrictions and logic operators in the logic description. These were previously introduced in Section 2.3.2. Two examples of logic description are provided here: BOM of product and product life cycle. They are both needed for decision support functions, described more fully in Chapter 5. BOM of product is used in design refresh planning, and product life cycle is used in obsolescence forecasting.

- **BOM of product**

BOM information of product is described with a closure axiom. A closure axiom on a slot consists of a universal restriction that acts along the slot to say that it can *only* be filled by the specified fillers. The restriction has a filler that is the *union* of the fillers that occur in the existential restrictions for the slot (Horridge, 2009). In Figure 4-3, the BOM of Product A shows it consists of Part a, Part b and Part c. The closure axiom makes the Product A 'have and only have' those parts in the BOM. It includes the parts in the BOM with existential restrictions 'some', and excludes the parts not in the BOM with universal restrictions 'only'.

Basic BOM information of Product A

Product	**Part**
Product A	Part a
	Part b
	Part c

⟺

Logic description for Product

has part **some** Part a
has part **some** Part b
has part **some** Part c

has part **only** (Part a **or** Part b **or** Part c)

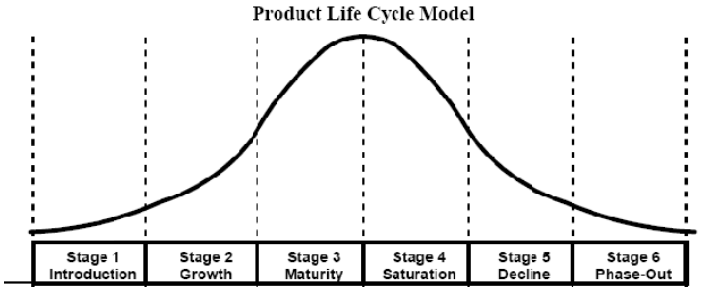
Figure 4-3: Logic description for BOM of product

- **Product life cycle**

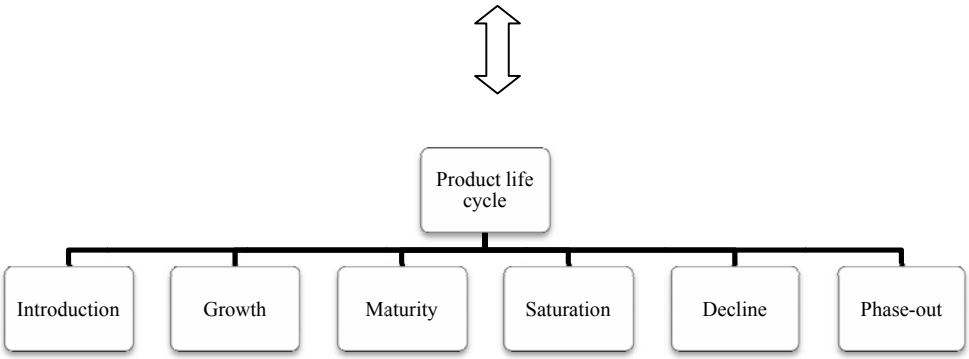
The most common product life cycle model includes six stages: introduction, growth, maturity, saturation, decline and phase-out, as shown in Figure 4-4(a). Each stage is associated with several characteristics such as sales, price, usage and so on.

The class Product life cycle is defined with main slots including start, end, and characteristics. Start and end describe start time and end time of the product life cycle period. And characteristics describe characteristics associated with the life cycle, which

are described in Table 2-2. The class Product life cycle has six subclasses representing six stages of the life cycle. These subclasses inherit the slots from the class Product life cycle. The start time of one stage class equals to the end time of its previous stage class if there is no special indication, except the start time of the introduction class and the end time of the phase-out class. Classes of the product life cycle ontology are shown in Figure 4-4(b).



(a)



Main slots of class Product life cycle

Class	Slot	Value type
Product life cycle	start	date
	end	date
	characteristic	string

(b)

Figure 4-4: Product life cycle ontology

The covering axiom is applied to the product life cycle ontology. A covering axiom consists of two parts: the class that is being covered, and the classes that form the

covering. It specifies a class is covered by its subclasses, which means that a member of the class must be a member of at least one of its subclasses. If subclasses are disjoint, a member of the class must be a member of the only one of its subclasses (Horridge, 2009). The covering axiom for the product life cycle ontology can be illustrated with set theory as follows:

$$\begin{gathered} B_i \subseteq A\ (i = 1,2,3,4,5,6) \\ A \subseteq (B_1 \cup B_2 \cup B_3 \cup B_4 \cup B_5 \cup B_6) \\ B_i \cap B_j = \emptyset\ (i \neq j; i = 1,2,3,4,5,6; j = 1,2,3,4,5,6) \end{gathered} \tag{4-1}$$

Where A represents the class Product life cycle; $B_{i/j}$ represents the subclass i/j of the class Product life cycle. (B_1 represents introduction, B_2 represents growth, B_3 represents maturity, B_4 represents saturation, B_5 represents decline, and B_6 represents phase-out.)

The product life cycle ontology is applied with the covering axiom, and its subclasses are disjoint, so a member of the product life cycle class must be a member of either introduction, or growth, or maturity, or saturation, or decline, or phase-out, in accordance with the fact that a product can only be in one stage of the life cycle at a time. The logic description with restrictions and logic operators for the class Product life cycle is shown in Figure 4-5.

Equivalent class of product life cycle:
Introduction **or** Growth **or** Maturity **or** Saturation **or** Decline **or** Phase_Out

Figure 4-5: Logic description for class Product life cycle

Not all products conform to the six life cycle stages. Some may undergo a false start and die out, and others may be revitalized after a niche market. Their product life cycle classes just are special cases, which only have part of subclasses representing some of the six life cycle stages.

Classes of the global obsolescence ontology have been developed based on the use cases of DMSMS obsolescence management and shown in Figure 3-3. Examples of local data sources are provided to illustrate the procedure of applying hybrid approaches for

information integration: the first source (S_1) is part of a relational database[4-1] owned by large electronic component distributor (e.g., mouser electronics) which contains information about parts and their suppliers; The second source (S_2) is part of an object-oriented database[4-2] used by a company for managing DMSMS obsolescence; The third source (S_3) is part of a file system which stores the amplifier product information of a small amplifier product supplier. Three examples of local data sources are shown in Figure 4-6.

In the process of defining local ontologies for each data source, the structure of each data source is translated into OWL, which is similar to the file shown in Figure 4-2. And the information needed by the global ontology such as the source name and type is added. Let $S = \{S_0, S_1, S_2, \cdots S_N\}$ be a set of ontologies, and S_0 is the global ontology. Each ontology S_i is a collection of classes $S_i = \{C_{1i}, C_{2i}, \cdots, C_{Mi}\}$, and a class $C_{ji} \in S_i$ is characterized by set of slots $A(C_{ji})$ ($a_t \in A(C_{ji})$).

[4-1] Relational databases are consisted of multiple tables which are related to each other, information can be retrieving and update from a combination of tables with the Structured Query Language (SQL). Relational databases have been extremely successful in the market and there are many commercial products (e.g., DB2, Informix, Oracle, and Sybase).

[4-2] Relational databases are not suitable for applications with complex data structures or new data types for larger, unstructured objects such as CAD/CAM, Geographic, multimedia, imaging and graphs, so object-oriented databases appeared in the early 80's. There is no specification existed for object-oriented databases. But if a database is considered to be a object-oriented database, it must have mandatory features of both object-oriented system (e.g., complex objects, types & classes, object identity, class hierarchy, encapsulation, extensibility, overriding & overloading and computational completeness) and DBMS (e.g., persistence, storage management, concurrency, recovery, and querying).

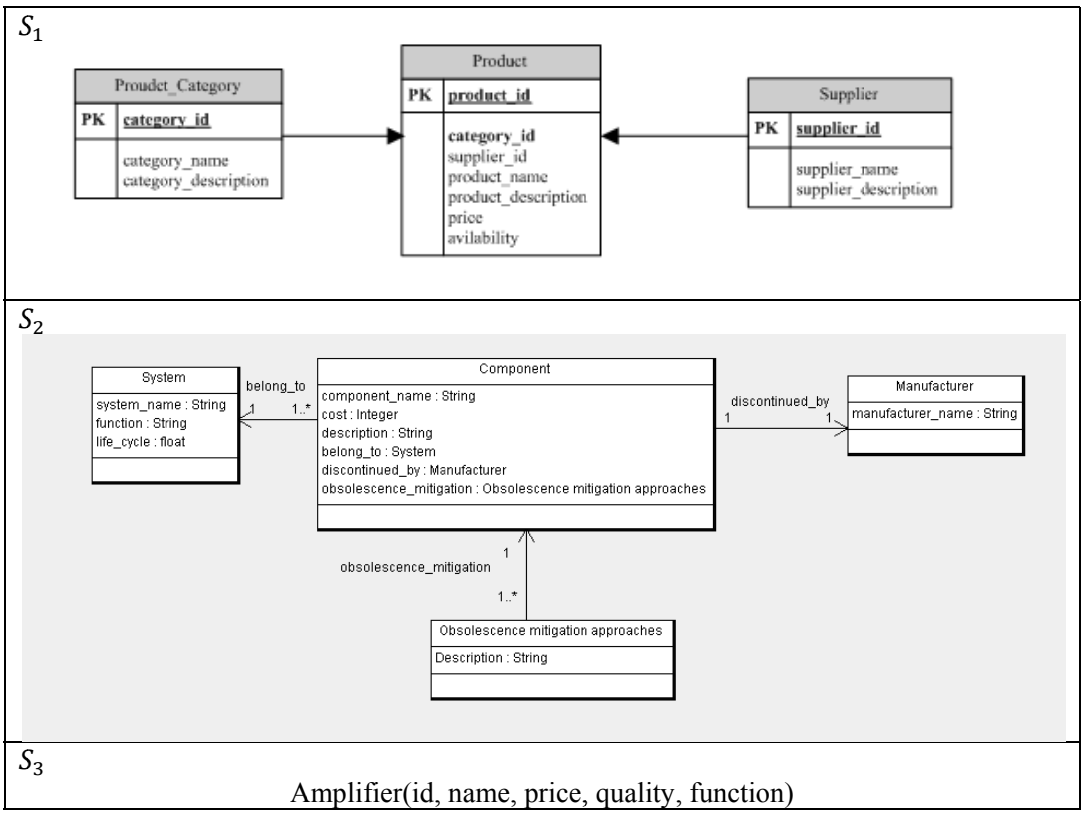


Figure 4-6: Examples of local data sources

4.3 Quantify Semantic Relationships

Relationships between classes in ontologies are important for integration of heterogeneous sources. Two types of relationships between classes are defined: semantic relationship and path relationship. Semantic relationship indicates the relationship of semantic meaning between classes. Path relationship indicates the relationship between classes through a path in the relationship diagram.

Semantic relationships between classes can be identified by comparing their slots. Slots of classes convey the most information about classes. δ is used to denote the semantic relationship between two classes. Its value can be calculated using the following equation:

$$\delta(C_{ji}, C_{hk}) = \frac{2|\{(a_t,a_q)|a_t=a_q,a_t\in A(C_{ji}),a_q\in A(C_{hk})\}|}{|A(C_{ji})|+|A(C_{hk})|} F \quad (4\text{-}2)$$

Where $a_t = a_q$ means that slots a_t and a_q indicate the same thing. F is an adjustment factor allowing domain experts to adjust the value of δ according to their experience. $\delta \in [0,1]$. The greater δ means the closer semantic relationship between two classes. For example, the value of the relationship between the class Part in the global ontology (S_0) and the class Product in the local ontology (S_1) (Figure 4-7), and the value of the relationship between the Product in the global ontology (S_0) and the class Product in the local ontology (S_1) can be calculated according to the equation (4-2):

$$\delta(\text{Part}(S_0),\text{Product}(S_1)) = \frac{2\times7}{7+8} \times 1 = 0.93 \ (F = 1)$$

$$\delta(\text{Product}(S_0),\text{Product}(S_1)) = 0$$

Product(S_1) has closer relationship with Part(S_0) than with Product(S_0). That is in accordance with the fact. Parts which build up products (systems) are named Product(S_1) in the first local data source (S_1) because this source is owned by a part vendor, and this part vendor called what they sell product. Product(S_1) in the first local data source (S_1) actually indicates Part(S_0), not Product(S_0) in the global ontology. Therefore, the relationship between classes should be determined from the semantic structure aspect of classes, not just class names.

Part (S_0)		Product (S_1)
name	=	product_id
source	=	supplier_id
characteristics	=	product_characteristics
interface	=	product_interface
function	=	product_function
inventory	=	availability
cost	=	price
		Category_id

Figure 4-7: Equivalent slots between classes

4.4 Discretize Semantic Relationships

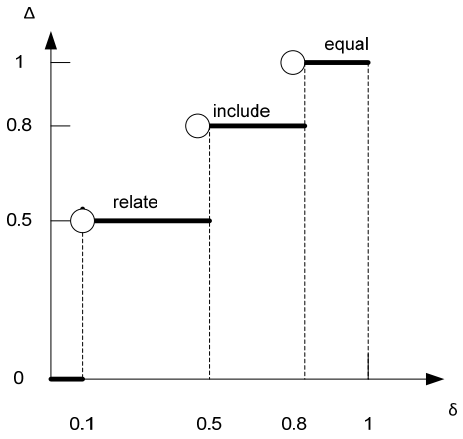
Based on the value of δ, three types of relationships between classes are defined:

- **Equal** ($0.8 < \delta \leq 1$) (Class A equals to class B, $A = B$), one class equals to another class if these two classes represent the same domain object, and can be interchangeably used without changing meaning.
- **Include** ($0.5 < \delta \leq 0.8$) (Class A includes class B, $A \supset B$), class A include class B if all members of class B are also members of class A.
- **Relate** ($0.1 < \delta \leq 0.5$) (Class A relates to class B, $A \sim B$), it can be any kind of relationship except equivalence and inclusion. For example, in the general ontology, the class Part is related to the class Source, because parts are provided by sources.

There is no relationship between two classes if $0 \leq \delta \leq 0.1$.

Discrete variable Δ is used to represent the type of relationship between classes. Let $\Delta_{=} = 1$, $\Delta_{\supset} = 0.8$, $\Delta_{\sim} = 0.5$. The function $f: \delta \longrightarrow \Delta$ is as follows (The detail of the discretization function is shown in Figure 4-8):

$$\Delta = \begin{cases} \Delta_{=} = 1 & if\ 0.8 < \delta \leq 1 \\ \Delta_{\supset} = 0.8 & if 0.5 < \delta \leq 0.8 \\ \Delta_{\sim} = 0.5 & if\ 0.1 < \delta \leq 0.5 \\ 0 & if\ 0 \leq \delta \leq 0.1 \end{cases} \tag{4-3}$$



$\boldsymbol{\delta}$	[0,0.1]	(0.1,0.5]	(0.5,0.8]	(0.8,1]
Δ	no relation	$\Delta_{\sim}(0.5)$	$\Delta_{\supset}(0.8)$	$\Delta_{=}(1)$
relationship	no relation	relate	include	equal

Figure 4-8: Function $\boldsymbol{f}$: $\boldsymbol{\delta} \longrightarrow \boldsymbol{\Delta}$

4.5 Create Relationship Diagram

Each pair of classes in the global ontology and local ontologies needs to be compared to determine their semantic relationship by calculating δ and discretizing it. If there is total number of K classes in the set of ontologies S, the number of comparisons is $\binom{K}{2} = \frac{K(K-1)}{2}$. After each pair of classes is compared, the relationship diagram of all classes in the global ontology and local ontologies can be created, as shown in Figure 4-9. Classes are connected through one of the three relationships: equal, include, and relate.

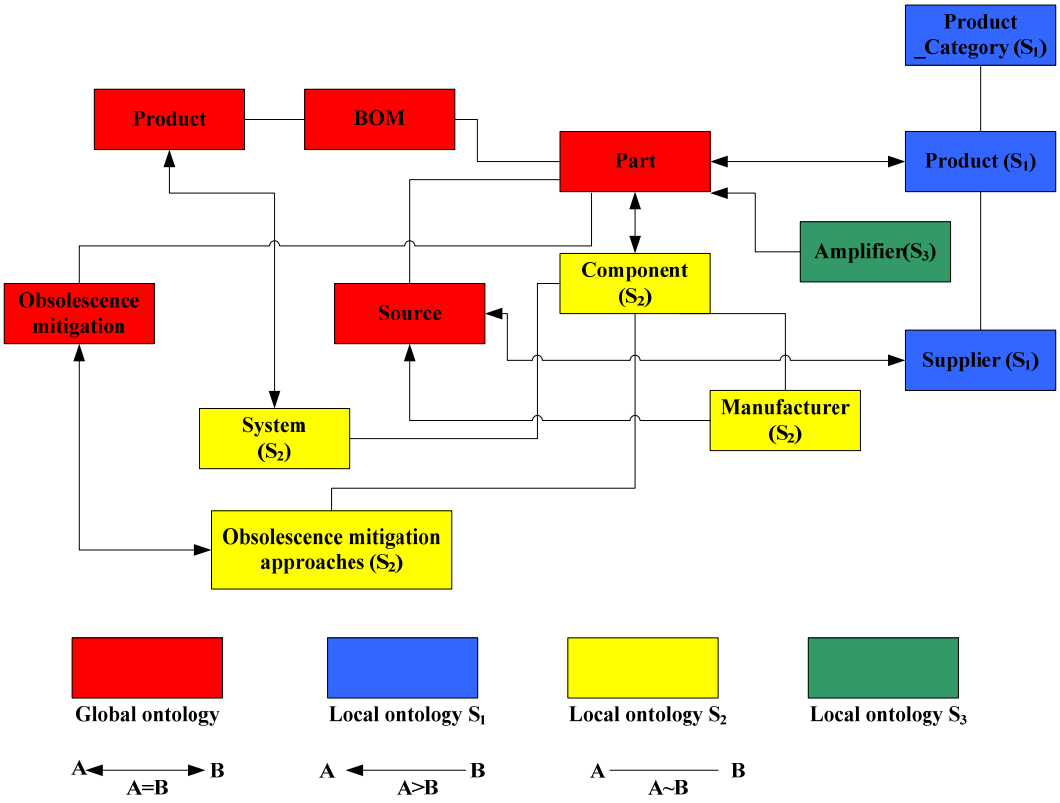


Figure 4-9: Relationship diagram of classes in global and local ontologies

4.6 Calculate Path Relationships

From the relationship diagram, more potential relationships between classes can be detected, which are path relationships. For example, $\delta(\text{Product}(S_0),\text{Product}(S_1)) = 0$ shows there is no semantic relationship between the class Product(S_0) and the class Product(S_1). However, it is not difficult to find that there is a path existing between these two classes in the relationship diagram (Figure 4-9), which means the class Product(S_0) can be related to the class Product(S_1) via the class BOM(S_0) and the class Part(S_0). σ is used to denote the path relationship between two classes. Its value can be calculated using the following equation:

$$\sigma(C_{ji}, C_{hk}) = \Delta(C_{ji}, C_1) \times \Delta(C_1, C_2) \times \cdots \times \Delta(C_{m-1}, C_m) \times \Delta(C_m, C_{hk}) \qquad (4\text{-}4)$$

Where $C_1, C_2, \cdots C_{m-1}, C_m$ are m intermedial classes existing in the path between C_{ji} and C_{hk}. If there is no intermedial classes existing in the path between C_{ji} and C_{hk}, then $\sigma(C_{ji}, C_{hk}) = \Delta(C_{ji}, C_{hk})$. $\sigma \in [0,1]$. For example,

σ(Product(S_0),Product(S_1)) = Δ(Product(S_0), BOM(S_0))$\times$ Δ(BOM(S_0), Part(S_0)) $\times$ Δ(Part(S_0), Product(S_1))=0.5×0.5×1=0.25

If there are multiple paths existing between two classes, then σ with maximum value is chosen as the final relationship Σ between two classes:

$$\Sigma(C_{ji}, C_{hk}) = \max\left(\sigma_1(C_{ji}, C_{hk}), \sigma_2(C_{ji}, C_{hk}) \cdots \sigma_n(C_{ji}, C_{hk})\right) \quad (4\text{-}5)$$

Where $\sigma_1, \sigma_2 \cdots \sigma_n$ represent the relationship values of n paths existing between classes C_{ji} and C_{hk}. $\Sigma \in [0,1]$. For example, in Figure 4-9, besides the path relating the class Product(S_0) and the class Product(S_1) via the class BOM(S_0) and the class Part(S_0), there is the other path existing between the class Product(S_0) and the class Product(S_1): Product(S_0)—System(S_2)—Component(S_2)—Part(S_0)— Product(S_1).

σ_1(Product(S_0),Product(S_1)) = Δ(Product(S_0), BOM(S_0))$\times$ Δ(BOM(S_0), Part(S_0)) $\times$ Δ(Part(S_0), Product(S_1))=0.5×0.5×1=0.25

σ_2 (Product(S_0),Product(S_1)) = Δ (Product(S_0), System(S_2))$\times$ Δ (System(S_2), Component(S_2))$\times$ Δ(Component(S_2), Part(S_0)) $\times$ Δ(Part(S_0), Product(S_1))
=1×0.5×1×1=0.5

Σ(Product(S_0),Product(S_1))=max(σ_1(Product(S_0),Product(S_1)), σ_2(Product(S_0),Product(S_1))=max(0.25,0.5)=0.5

The value of $\Sigma(C_{ji}, C_{hk})$ of each pair of classes in the relationship diagram (Figure 4-9) is shown in Table 4-1.

Table 4-1: **Σ** of classes in relationship diagram

$\boldsymbol{\Sigma(C_{ji}, C_{hk})}$	Product	BOM	Part	Source	Obso mitigation	Product (S_1)	Product_ Category (S_1)	Supplier (S_1)	System (S_2)	Component (S_2)	Manufacturer (S_2)	Obso mitigation approaches (S_2)	Amplifier (S_3)
Product	1												
BOM	0.5	1											
Part	0.5	0.5	1										
Source	0.25	0.25	0.5	1									
Obsolescence mitigation	0.25	0.25	0.5	0.25	1								
Product(S_1)	0.5	0.5	1	0.5	0.5	1							
Product_Category(S_1)	0.25	0.25	0.5	0.25	0.25	0.5	1						
Supplier(S_1)	0.25	0.25	0.5	0.25	0.25	0.5	0.25	1					
System(S_2)	1	0.5	0.5	0.25	0.25	0.5	0.25	0.25	1				
Component(S_2)	0.5	0.5	1	0.5	0.5	1	0.5	0.5	0.5	1			
Manufacturer(S_2)	0.25	0.25	0.5	0.8	0.25	0.5	0.25	0.8	0.25	0.25	1		
Obsolescence mitigation approaches(S_2)	0.25	0.25	0.5	0.25	1	0.5	0.25	0.25	0.25	0.5	0.25	1	
Amplifier(S_3)	0.4	0.4	0.8	0.4	0.4	0.8	0.4	0.4	0.4	0.8	0.4	0.4	1

4.7 Indentify Groups of Classes

The groups of classes in the global ontology and local ontologies are identified using the hierarchical clustering method. The method builds the hierarchy from the individual class by progressively merging groups. Φ is used to denote the relationship between two clusters, its value can be determined using the following equation:

$$\Phi = \max(\Sigma(C_{ji}, C_{hk}) | \forall C_{ji} \in one\ cluster, \forall C_{hk} \in the\ other\ cluster) \quad (4\text{-}6)$$

Clustering is an iterative process and starts by placing each class in a cluster by itself. At each iteration, the clusters with the greatest values of Φ are merged. As the result of clustering, classes are classified into groups at different levels of affinity (according to the values of Φ) to form a hierarchical structure, as shown in Figure 4-10.

4.8 Create Mappings between Classes

Groups of classes are formed with using the clustering method. In each group of classes, there exists a 0~1 class from the global ontology.

- If a group contains a class from the global ontology, like groups G_1, G_2, G_3, G_4, G_6, the class from the global ontology is the representative of the group, that is, G_1: Part, G_2: Source, G_3: BOM, G_4: Product, G_6: Obsolescence mitigation. Mappings from the representative class to other classes (which may belong to different local ontologies) in the same group needs to be built. For example, the class Part is the representative of G_1, and the mapping relationship between the class Part and other classes in G_1 is shown in Figure 4-11.
- If the group does not contain any class from the global ontology, like the group G_5, this group may be discarded. Otherwise it needs to be added to the global ontology. If the group is added to the global ontology, a new class that represents the unified view of all the classes in the group needs to be created, and mappings that relate the new class and the existing classes in the group also need to be built.

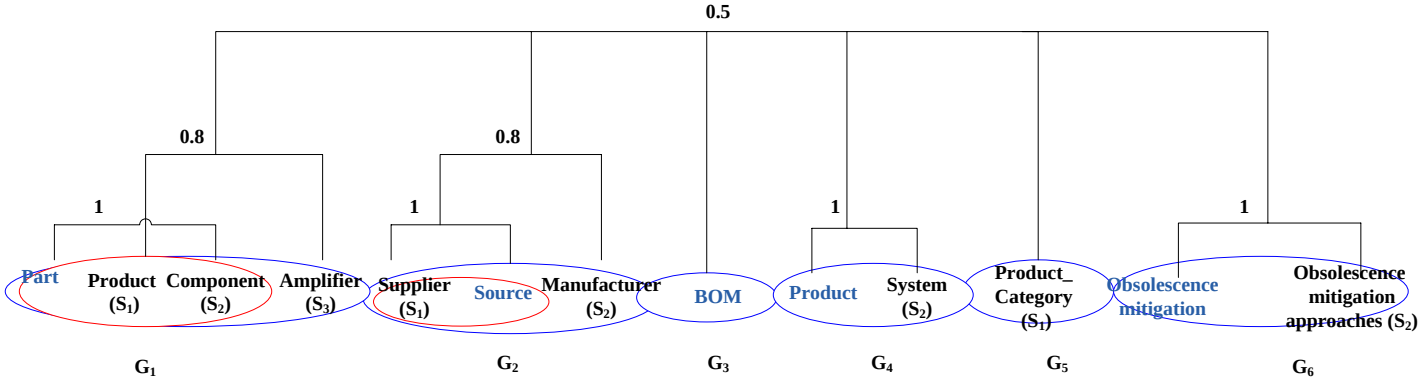


Figure 4-10: Groups of classes identified with cluster method

Class	Slots						
Part	name	source	characteristics	interface	function	inventory	cost
Product (S_1)	product_id	supplier_id	product_ characteristics	product_interface	product_ function	availability	price
Component (S_2)	component_ name	discontinued_ by	description	description	description	discontinued_ by	cost
Amplifier (S_3)	id/name				function	quality	price

Figure 4-11: Mapping for classes in group G_1

4.9 Conclusion

In this chapter, the procedure of applying ontology-based hybrid approaches for information integration is provided. A quantitative method and clustering technique are applied in the procedure. As shown in Figures 4-10 and 4-11, relationships between ontology classes representing heterogeneous data sources can be identified after applying the approach. The ontology-based hybrid approach combined with the obsolescence ontology scheme developed in the last chapter can be used for obsolescence data management in the obsolescence management information system, which will be introduced in Chapter 6.

Chapter 5 Decision Support

This chapter is focused on the decision model development for proactive and strategic obsolescence management. Section 5.1 introduces an obsolescence forecasting method based on the product life cycle. Section 5.2 presents the development and application of design refresh planning models for strategic obsolescence management (Section 5.2.1-5.2.3). Section 5.2.1 provides the overview of the approach. The mathematical model development for making the plan for design refreshes to minimize the cost considering the data uncertainty is introduced in Section 5.2.2, followed by the demonstration of applying models for the ECU system's hardware and software obsolescence problems in Section 5.2.3.

5.1 Obsolescence Forecasting

The obsolescence forecasting method is based on the development of models for the product life cycle. If the life cycle curve of a product with six life cycle stages can be represented by the curve in Figure 4-4(a), the obsolescence forecasting method proposed by Solomon et al. fits the sales data of the product with the Gaussian distribution to obtain the product life cycle curve (Solomon, Sandborn, Pecht, 2000). Indeed, products which have the Gaussian distribution life cycle are common. Examples include most electronic components, which are vulnerable to DMSMS obsolescence. The Gaussian distribution has been used by the Electronic Industries Association (EIA) to define a standardized product life cycle curve, and hence is well known and familiar to equipment suppliers (Anonymous, 1997). The equation of the Gaussian distribution life cycle curve is:

$$f(x) = ke^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{5-1}$$

Where $f(x)$ is the sales of the product at time x. The curve is characterized by parameters k, μ, and σ. k is the sales peak, μ is the mean time corresponding to the sales peak, and σ is the standard deviation.

If the life cycle of a product has the Gaussian distribution, the time ranges of the six life cycle stages can be calculated: introduction $(\mu - 3\sigma, \mu - 2\sigma)$, growth $(\mu - 2\sigma, \mu - \sigma)$, maturity $(\mu - \sigma, \mu)$, saturation$(\mu, \mu + \sigma)$, decline $(\mu + \sigma, \mu + 2\sigma)$, and phase-out $(\mu + 2\sigma, \mu + 3\sigma)$. For obsolescence forecasting, Solomon's obsolescence forecasting method defines the zone of obsolescence to refer to a period of time in which a product has high probability of being obsolete. And its time range is determined as $(\mu + 2.5\sigma, \mu + 3.5\sigma)$[5-1] (Solomon, Sandborn, Pecht, 2000).

The life cycle curve of a product is obtained by fitting the historical sales data [5-2] with the Gaussian distribution. The least squares method (Wolberg, 2006) is applied for data fitting, which minimizes the sum of squared residuals, residuals being the difference between a historical sale data value and the fitted value provided by the Gaussian distribution life cycle model. The optimum is obtained when the sum of squared residuals is minimized:

$$\underset{k,\mu,\sigma}{Minimize}\, S = \sum_{i=1}^{n}(y_i - f(x_i))^2 = \sum_{i=1}^{n}\left(y_i - ke^{\frac{-(x_i-\mu)^2}{2\sigma^2}}\right)^2 \qquad (5\text{-}2)$$

Where S is the sum of squared errors; x_i is a time point; y_i is a historical sale data at time x_i; $f(x) = ke^{\frac{-(x-\mu)^2}{2\sigma^2}}$ is the expression of the Gaussian distribution life cycle curve with three decision variables k, μ, σ.

The minimum of the sum of squared residuals can be found by setting the gradient to zero:

$$\frac{\partial S}{\partial k} = 0;\ \frac{\partial S}{\partial \mu} = 0;\ \frac{\partial S}{\partial \sigma} = 0 \qquad (5\text{-}3)$$

[5-1] The time range of the zone of obsolescence can be determined using data mining of historical data (e.g., last-order or last-ship dates) to achieve more accurate obsolescence forecasting (Sandborn, Mauro, Knox, 2007).

[5-2] The sales data is mainly in the form of number of units shipped. If it is not available, sales in market dollars or percentage market share may be used, as long as the total market does not increase appreciably over time (Solomon, Sandborn, Pecht, 2000).

where $\frac{\partial S}{\partial k}, \frac{\partial S}{\partial \mu}, \frac{\partial S}{\partial \sigma}$ are functions of variables k, μ, σ. Or the problem can be directly solved with the nonlinear programming solver using iterative refinement. When the minimum of the sum of squared residuals is found, three parameters of the life cycle curve k, μ, σ are determined.

The procedure of obsolescence forecasting for the product with the Gaussian distribution life cycle is proposed based on Solomon's obsolescence forecasting method, which consists of the following steps (Solomon, Sandborn, Pecht, 2000), as shown in Figure 5-1:

- Step 1: Obtain the life cycle curve of the product by fitting the historical sales data with the Gaussian distribution using the least squared method. The future sales data can be predicted together with the life cycle curve.
- Step 2: Determine the three parameters of the life cycle curve k, μ, and σ[5-3].
- Step 3: Calculate the time range of the zone of obsolescence $(\mu + 2.5\sigma, \mu + 3.5\sigma)$.
- Step 4: Forecast if the product will be obsolete in the specific future time. If the specific future time is in/after the zone of obsolescence, the product is forecasted to be obsolete at that time; otherwise, it is not.

This procedure of obsolescence forecasting can be represented with SWRL rules, and queries are executed based on these rules. The SWRL rules for obsolescence forecasting are presented in Section 6.2.3 and Figure 6-12.

[5-3] For some products, within the same type of the product, life cycle curves characterized by parameters k,μ, and σ can vary with some primary attributes of the product. Examples are memory chips whose life cycle curves vary with different memory size. Memory size is the primary attribute describing the memory chip that evolves over time (Solomon, Sandborn, Pecht, 2000; Pecht, Solomon, Sandborn, Wilkinson, 2002; Sandborn, Mauro, Knox, 2007). For these products, if the primary attributes of the product are not considered, the parameters k,μ, and σ obtained from the sales data of the product are only average values for that product.

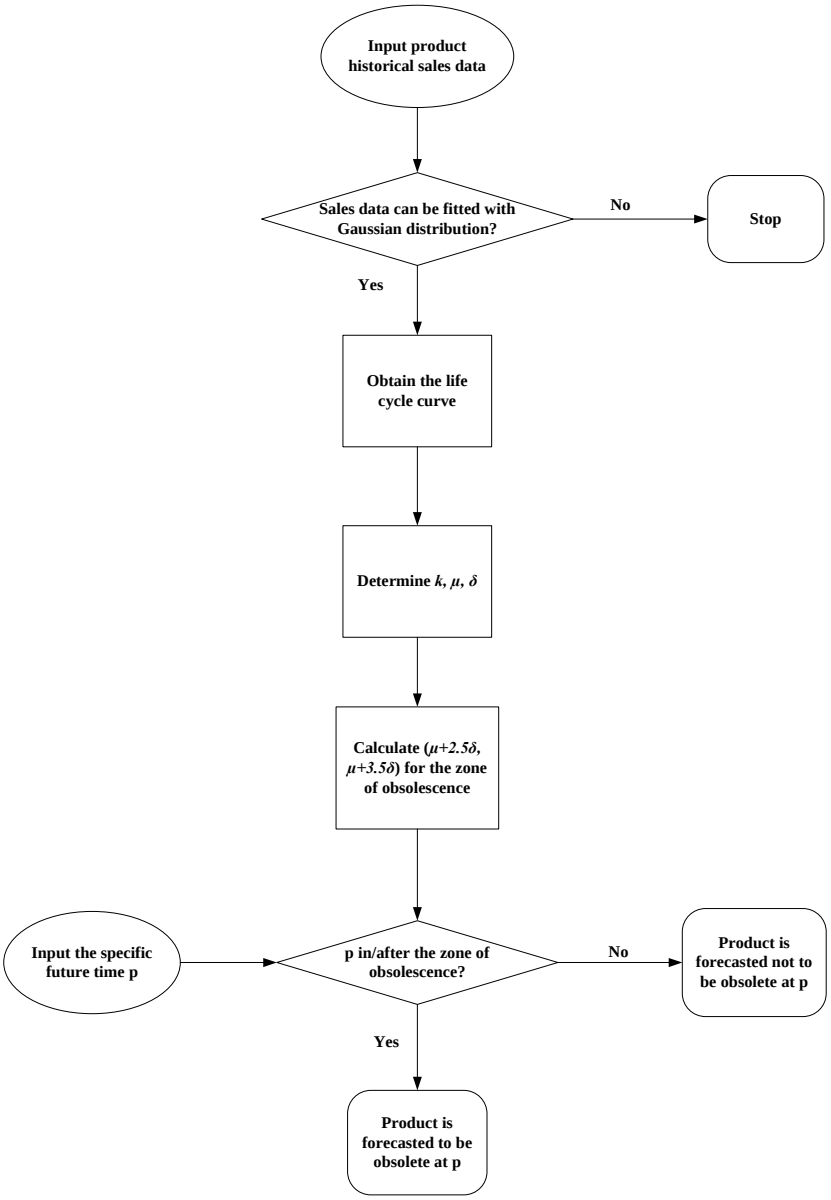


Figure 5-1: Flow diagram of obsolescence forecasting for product with Gaussian distribution life cycle

One part type in the electronic engine control unit (ECU) is used for demonstrating the procedure of obsolescence forecasting. The ECU has been used as an example for demonstration of the application of the ontology framework in Chapter 3. The part type is 64Mbit monolithic flash memory. Its recorded sales data for the 1992-2002 time period are shown in Table 5-1. This memory was not introduced into the market until 1996.

Table 5-1: Recorded sales data of 64Mbit monolithic flash memory for 1992-2002

Year	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002
Unit shipments (Millions)	N/A	N/A	N/A	N/A	0.1	1	12	35	90	200	360

The sales data are fitted with the Gaussian distribution function. When the objective of minimizing the sum of squared residuals is achieved, the life cycle curve of the 64Mbit monolithic flash memory is obtained, as shown in Figure 5-2. The historical sales data are also shown in the same figure (blue solid), but the data are fitted so well that the curve line of historical sales data is nearly completely covered by the life cycle curve (red dash). The values of three parameters of the life cycle curve k, μ and δ are 543.411 millions, 2003.861 years and 2.0345 years. The time ranges of the six life cycle stages are introduction (1997.758, 1999.792), growth (1999.792, 2001.827), maturity (2001.827, 2003.861), saturation (2003.861, 2005.896), decline (2005.896, 2007.930) and phase-out (2007.930, 2009.965). And the time range of the zone of obsolescence is (2008.947, 2010.982). The forecasted sales data after 2002 are shown in Table 5-2.

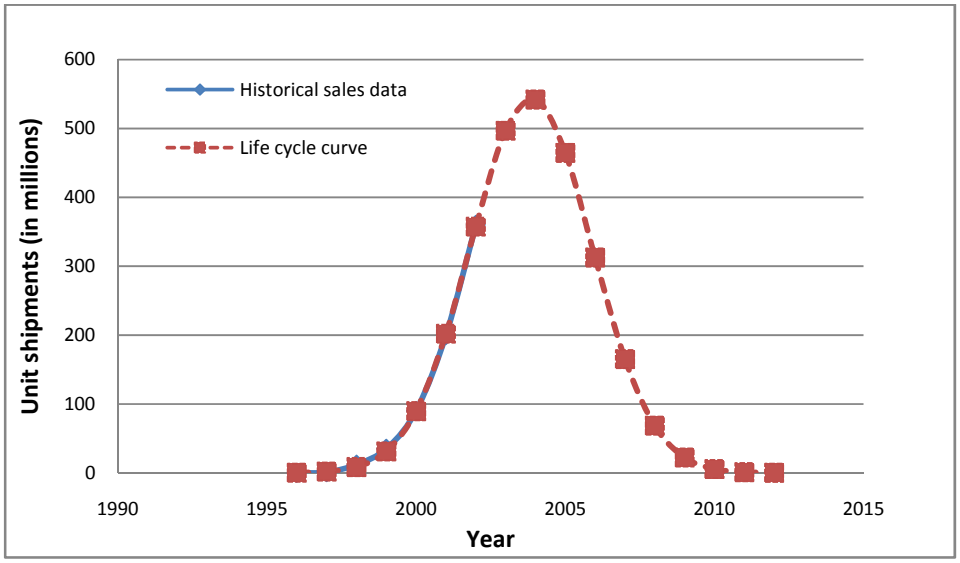


Figure 5-2: Life cycle curve of 64 Mbit monolithic flash memory

Table 5-2: Forecasted sales data of 64Mbit monolithic flash memory after 2002

Year	Recorded Sales data	Life cycle curve	Residual	Squared residual
1996	0.1	0.311133913	-0.21113	0.044578
1997	1	1.84221418	-0.84221	0.709325
1998	12	8.566537579	3.433462	11.78866
1999	35	31.28549001	3.71451	13.79758
2000	90	89.73317458	0.266825	0.071196
2001	200	202.132216	-2.13222	4.546345
2002	360	357.5942078	2.405792	5.787836
2003		496.8414811	Forecasted Sales data	Minimum sum =36.74533
2004		542.1477424		
2005		464.6114338		
2006		312.7048136		
2007		165.2919388		
2008		68.61849912		
2009		22.37191647		
2010		5.728456208		
2011		1.151978674		
2012		0.181938125		

5.2 Design Refresh Planning

With the information input from obsolescence forecasting, design refresh planning can be applied to strategically managing DMSMS obsolescence for sustainment-dominated systems. As introduced in Section 2.4.1.3, design refresh planning includes several predetermined design refreshes applied on the system (for the obsolete parts and predicted obsolete parts) with long-term mitigation approaches (e.g., emulation, upgrade of similar part, etc) at some discrete time points during the life cycle of the system. And requalification is required if there is design change. If some parts are obsolete during the time periods between two design refreshes, short-term mitigation approaches (e.g., stock, last time buy, etc) are applied on a part-specific basis until the next design fresh. The design refresh plan is to achieve the goal of minimizing the overall life cycle cost of the system under the influence of obsolescence by determining:

- When to proceed with design refresh;
- What obsolete/non-obsolete system components should be replaced at a specific design refresh.

5.2.1 Overview of Approach

A model assumption is made to simply the development of design refresh planning models. In theory, design refresh can take place at any point on the timeline. This makes determining the time to perform design refresh very difficult since the range of candidate solutions will be the entire timeline. Singh et al. indicated that design refresh that completes at a point in time that is significantly earlier than the start of the next production event will never be as economically advantageous as one that is completed "just in time" for the next production event[5-4](Singh, Sandborn, 2006). This is always true if the rate of interest is non-negative, because the net present cost of a delayed refresh will always be lower than the net present cost of a refresh at an earlier date. In addition, it is in accord with the fact that in reality the procurement of parts is always associated only with production events (Singh, Sandborn, 2006). Based on Singh et al.'s viewpoint, the following assumption is made:

Assumption: Design refreshes only happen at the time points of production events on the timeline.

According to the assumption, the number of possible time points for design refreshes becomes finite and locations can be easily determined. Suppose there is a production plan consisting of n production events, which happen at n time points during the life cycle of the product. At each time point of a production event, design refresh may or may not take place. So there are 2^n-1 possible combinations. (It is assumed that the production plan is known in this dissertation, since determination of the production plan is not the focus of this research.)

In the assumption, the time length of executing a design refresh is neglected compared to the long life cycle of the system. In some situations, a design refresh may take a long time period on the timeline, and cannot be neglected, which is not discussed in this research.

[5-4] A production event is scheduled to replace product that fails in the field due to wear out or overstress during the product's usage life.

5.2.1.1 Deterministic Cost Formula

The life cycle cost of the system can be calculated under different design refresh plan scenarios, and the design refresh plan with the minimum life cycle cost will be chosen. If the obsolescence dates of parts in the system are treated deterministically, the deterministic cost model can be developed. The overall life cycle costs $C_{overall}$ includes production costs $C_{production}$, design refresh costs $C_{refresh}$, and maintenance costs $C_{maintenance}$:

$$C_{overall} = C_{production} + C_{refresh} + C_{maintenance} \tag{5-4}$$

Production costs $C_{production}$ are the costs associated with the production planning. The production planning determines productions in production events to meet demand requirements including the demand required to replace systems that fail in the field during their usage life. Design refresh costs $C_{refresh}$ are the costs associated with the design refresh planning. They include the costs of dealing with obsolescence of parts or system in each design refresh. Maintenance costs $C_{maintenance}$ indicate the costs of maintenance activities for obsolescence or failure of parts during periods between two sequential production events. There is no design refresh during these periods, and short-term mitigation approaches are used for the obsolescence of parts. Production costs do not change for the system under different design refresh planning. Production costs and part of maintenance costs for failure of parts are related to reliability management, not obsolescence management. To simplify the cost model, only design refresh costs and short-term mitigation costs for obsolescence are taken into consideration. The simplified cost formula for obsolescence management is:

$$C_{overall\ obso} = C_{refresh} + C_{mitigation} \tag{5-5}$$

Where $C_{overall\ obso}$ are overall obsolescence management costs, and $C_{mitigation}$ are short-term mitigation costs for obsolescence during periods with no design refresh. The objective is to minimize the overall obsolescence management costs:

$$\begin{matrix} \min_i C_{overall\ obso,i} & \forall i \in \{Refresh\ candidates\} \\ s.t. & Constraints \end{matrix} \quad (5\text{-}6)$$

5.2.1.2 Expected Cost Formula

One important drawback of the deterministic cost model is not considering the data uncertainty. Main data uncertainty is from obsolescence date information. In reality, it is almost impossible to determine the exact future obsolescence date of a part. In the previous obsolescence forecasting, zone of obsolescence $(\mu + 2.5\sigma, \mu + 3.5\sigma)$ is used instead of the exact obsolescence date. To make the model more reasonable, the obsolescence dates are inputted as probability distributions (e.g., normal distribution, triangular distribution, and so on). The output costs also become probability distributions. The expected cost formula can be used:

$$E(C_{overall\ obso}) = E(C_{refresh}) + E(C_{mitigation}) \quad (5\text{-}7)$$

And the objective is to minimize the expected value of the overall obsolescence management costs:

$$\begin{matrix} \min_i E(C_{overall\ obso,i}) & \forall i \in \{Refresh\ candidates\} \\ s.t. & Constraints \end{matrix} \quad (5\text{-}8)$$

5.2.1.3 Real Situation Constraints

The design refresh planning model needs to consider constraints from real situation. These constraints should reflect legislative policies, budget realities, technology upgrade roadmaps, logistic limitations, and obsolescence management limitations (Nelson III, Sandborn, 2011). Different parts' obsolescence has different impact on the system, and different actions should be taken. For example, hardware (e.g., electronic parts) can continuously be used in the system although they are obsolete. So when hardware does become obsolete, it does not need to be replaced immediately. And failed hardware can be replaced with original obsolete parts from existing stock or aftermarket sources. In contrast, obsolete software are not allowed to be used in the system any longer and they must be updated or replaced as soon as possible, because the system may be in some

security risk if it continues operating with obsolete software. Hardware and software obsolescence are typical examples of real situations adding constraints to obsolescence management. To generate constraints that can be used to identify non-viable design refresh plans in real situations, Nelson III et al. (2011) summarized three types of obsolescence events that describe the impact of the part's obsolescence on the system. They are applicable to the management of obsolescence in real systems and sufficient to model all cases encountered (Nelson III, Sandborn, 2011). Examples of hardware and software obsolescence management with design refresh planning will be demonstrated in the following sections. With consideration of the real situation constraints, the design refresh planning model is:

$$\begin{array}{cc} \min_i E(C_{overall\ obso,i}) & \forall i \in \{Refresh\ candidates\} \\ s.t. & Constraints \\ & Real\ situation\ constraints \end{array} \tag{5-9}$$

5.2.2 Mathematical Models

The production plan consists of n production events that take place at the n time points $t_1, t_2, \cdots t_n$ during the life cycle T of the system which starts at t_0 and ends at $t_0 + T$. According to the assumption, possible design refreshes can only take place at these n time points of production events, which divide T into $n+1$ time periods $T_1, T_2, \cdots, T_n, T_{n+1}$. During these $n+1$ time periods, when some parts become obsolete, short-term mitigation approaches will be applied on a part-specific basis until the next design refresh. The timeline for design refresh planning is shown in Figure 5-3. The relationships between time points and time periods are:

$$\begin{gathered} T_k = t_j - t_{j-1}\ (k = j = 1,2,3, \cdots n) \\ T_{n+1} = t_0 + T - t_n \\ \sum_{k=1}^{n+1} T_k = T \end{gathered} \tag{5-10}$$

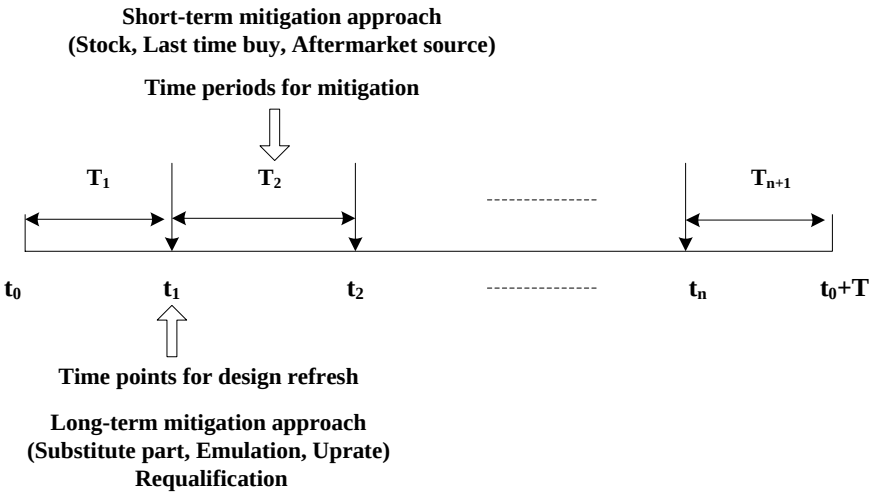


Figure 5-3: Timeline for design refresh planning

5.2.2.1 Deterministic Model

- **Objective function**

$$Minimize\ \ C_{overall\ obso} = C_{refresh} + C_{mitigation}$$
$$= \sum_{j=1}^{n}\left(\sum_{i=1}^{m} C_{ij}^{R}\, x_{ij} + C_{j}^{S} y_{j}\right) + \sum_{k=1}^{n+1} \sum_{i=1}^{m} C_{ik}^{M}\, z_{ik} \qquad (5\text{-}11)$$

Where,

y_j is the binary integer variable defined to represent whether design refresh takes place at the time point t_j:

$$y_j = \begin{cases} 1 & design\ refresh\ takes\ place\ at\ t_j \\ 0 & otherwise \end{cases}$$

x_{ij} is the binary integer variable defined to represent whether part type i is replaced at design refresh taking place at the time point t_j:

$$x_{ij} = \begin{cases} 1 & part\ type\ i\ is\ replaced\ at\ design\ refresh\ taking\ place\ at\ t_j \\ 0 & otherwise \end{cases}$$

z_{ik} is the binary integer variable defined to represent whether part type i becomes obsolete and is replaced using some short-term mitigation approaches during time period T_k:

$$z_{ik} = \begin{cases} 1 & part\ type\ \ i\ becomes\ obsolete\ and\ is\ replaced\ during\ T_k \\ 0 & otherwise \end{cases}$$

C_j^S is the set-up cost of design refresh taking place at t_j. Requalification cost is included if it is necessary.

C_{ij}^R is the design refresh cost for part type i at t_j.

C_{ik}^M is the short-term mitigation cost for part type i during T_k.

$i = 1,2, \cdots m$. (There are m part types in the system.)

$j = 1,2, \cdots n$. (There are n possible time points for design refresh during the life cycle T of the system.)

$k = 1,2, \cdots n + 1$. (There are $n + 1$ time periods during the life cycle T of the system.)

- **Constraints**

The objective function is subject to the following constraints:

Constraint I: $$\sum_{i=1}^{m} x_{ij} \leq M y_j \ (M \gg m) \quad (5\text{-}12)$$

This constraint indicates if design refresh does not take place at the time point t_j, no part type in the system will be replaced at this time point, i.e., if $y_j = 0$, then $x_{1j} = 0, x_{2j} = 0, \cdots, x_{mj} = 0$.

Constraint II: $$d_i = \left(1 - x_{ij}\right) d_i + x_{ij} D_{ij}^R \quad (5\text{-}13)$$

This constraint indicates if part type i is replaced at the design refresh taking place at t_j, i.e., $x_{ij} = 1$, its obsolescence date d_i is updated to the new date, i.e., $d_i = D_{ij}^R$. Otherwise, there is no change in d_i.

d_i is defined to represent the obsolescence date of part type i.

D_{ij}^R is the new obsolescence date for part type i if it is replaced at the design refresh taking place at t_j.

Constraint III: $$d_i \in T_k \implies z_{ik} = 1 \quad (5\text{-}14)$$

This constraint indicates if the obsolescence date of part type i belongs to the time period T_k, part type i will become obsolete during this period. The transform of the constraint will be provided with examples in the following sections.

Constraints IV: $$d_i = (1 - z_{ik})d_i + z_{ik}D_{ik}^M \quad (5\text{-}15)$$

This constraint indicates if part type i is replaced during the time period T_k, i.e., $z_{ik} = 1$, its obsolescence date d_i is updated to the new date, i.e., $d_i = D_{ik}^M$. Otherwise, there is no change in d_i.

D_{ik}^M is the new obsolescence date for part type i if it is replaced using some short-term mitigation approaches during T_k.

In conclusion, the deterministic cost model for design refresh planning can be formulated as follows:

$$Minimize \sum_{j=1}^{n}\left(\sum\nolimits_{i=1}^{m} C_{ij}^R x_{ij} + C_j^S y_j\right) + \sum_{k=1}^{n+1}\sum\nolimits_{i=1}^{m} C_{ik}^M z_{ik}$$

$$s.t.\ \textstyle\sum_{i=1}^{m} x_{ij} \le M y_j\ (M \gg m)$$

$$d_i = \left(1 - x_{ij}\right)d_i + x_{ij}D_{ij}^R$$

$$d_i \in T_k \implies z_{ik} = 1$$

$$d_i = \left(1 - z_{ij}\right)d_i + z_{ij}D_{ij}^M$$

The value of money changed with years caused by the interest rate including percentage discount is not considered in the model.

The "trade off" decision has to be made between replacing non-obsolete parts at a specific design refresh and waiting for them to be obsolete during the time period between two design refreshes and be dealt with short-term mitigation approaches

5.2.2.2 Probabilistic Model

Different from the deterministic model in which parts have deterministic obsolescence dates, the probabilistic model assumes the obsolescence dates of parts follow certain probability distribution. Because of probability distribution obsolescence dates, part type i in the system has l_i modes[5-5] of affecting the obsolescence management solution and cost. These l_i modes have probabilities $P_{i1}, P_{i2}, \cdots P_{il_i}$, and $P_{i1} + P_{i2} + \cdots + P_{il_i} = 1 (i = 1,2, \cdots m)$:

Part type 1 has l_1 modes with probabilities $P_{11}, P_{12}, \cdots P_{1l_1}$ $(P_{11} + P_{22} + \cdots + P_{1l_1} = 1)$;

Part type 2 has l_2 modes with probabilities $P_{21}, P_{22}, \cdots P_{2l_2}$ $(P_{21} + P_{22} + \cdots + P_{2l_2} = 1)$;

…

Part type m has l_m modes with probabilities P_{m1}, P_{m2}, $\cdots P_{ml_m}$ ($P_{m1} + P_{m2} + \cdots + P_{ml_m} = 1$).

For example, the part type in Figure 5-4 whose obsolescence date follows probability distribution $f(x)(a \leq x \leq b)$ has 3 modes:

Mode 1: the part type will be obsolete during time period (t_{k-1}, t_k), the probability is $\int_a^{t_k} f(x)$.

Mode 2: the part type will be obsolete during time period (t_k, t_{k+1}), the probability is $\int_{t_k}^{t_{k+1}} f(x)$.

[5-5] Mode is defined as a manner, way or method of executing obsolescence management.

Mode 3: the part type will be obsolete during time period (t_{k+1},t_{k+2}), the probability is $\int_{t_{k+1}}^{b} f(x)$.

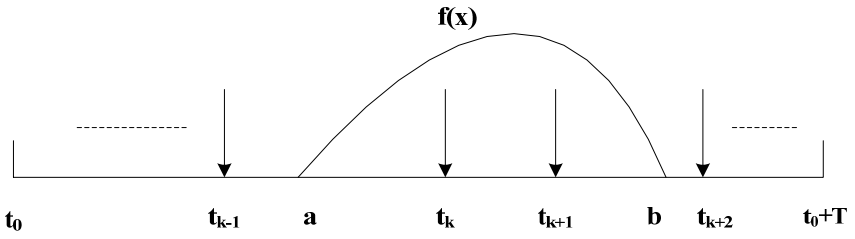


Figure 5-4: Example of part type with obsolescence date following probability distribution $\boldsymbol{f(x)}$

Combining modes of part types in the system can create $\binom{l_1}{1}\binom{l_2}{1}\cdots\binom{l_m}{1}$ final modes of affecting the final obsolescence management solution and cost. The expected minimum obsolescence management cost under the probability distribution obsolescence dates can be calculated as follows:

$$E(minimum\ C_{overall\ obso}) = \sum_{g=1}^{\binom{l_1}{1}\binom{l_2}{1}\cdots\binom{l_m}{1}} (minimum\ C_{overall\ obso,g}) P_g \qquad (5\text{-}16)$$

Where,

$minimum\ C_{overall\ obso,g}$ is the minimum obsolescence management cost of the final mode g;

P_g is the probability of the final mode g, $P_g = P_{1h_1} P_{2h_2} \cdots P_{mh_m}$ ($h_1 \in [1, l_1]$, $h_2 \in [1, l_2]$,$\cdots$ $h_m \in [1, l_m]$) ;

$g = 1,2,\cdots\left[\binom{l_1}{1}\binom{l_2}{1}\cdots\binom{l_m}{1}\right]$. (There are total $\binom{l_1}{1}\binom{l_2}{1}\cdots\binom{l_m}{1}$ final modes.)

$$\sum_{g=1}^{\binom{l_1}{1}\binom{l_2}{1}\cdots\binom{l_m}{1}} P_g = \left(P_{11} + P_{22} + \cdots + P_{1l_1}\right)\left(P_{21} + P_{22} + \cdots + P_{2l_2}\right)\cdots\left(P_{m1} + P_{m2} + \cdots + P_{ml_m}\right) = 1 \times 1 \times \cdots 1 = 1$$

5.2.3 Examples

Examples for demonstration of design refresh planning models for obsolescence management are from an electronic engine control unit (ECU), which has already been used for demonstrating the application of the ontology framework for product obsolescence in Chapter 3. The ECU consists of three hardware boards populated with hundreds of electronic components and software operating the hardware boards. These components may become obsolete during the life cycle of the ECU and need design refresh. Total of 52 types of representative components (42 types of hardware components and 10 types of software components) are selected from the system for demonstration of design refresh planning models, and the information related to obsolescence management are listed in Table 8-1 in Appendix A.

The life cycle of the ECU is 10 years (2000.00-2010.00[5-6]). During the life cycle of the ECU, 3 production events are scheduled in 2003.00, 2005.00, 2008.00. According to the assumption, design refreshes can only take place at these three time points. There are 7 possible combinations. Set-up costs of design refreshes in 2003.00, 2005.00, 2008.00 are provided in Table 5-3. Set-up costs include system change costs, testing costs, requalification costs and any other system-level costs except costs on individual part basis.

[5-6] The date format is consistent with that used in Mitigation of Obsolescence Cost Analysis (MOCA) software tool. MOCA requires that dates be entered as real numbers that have the following format:

a. b

Where a=the year, e.g., 2005; b=fraction of one year corresponding to the calendar date.
For example, September 23, 2009=2009.73 (September 23 is the 266th day of the year, b=266/365=0.73) (Sandborn,2007)

Table 5-3: Set-up costs of design refreshes

Time	2003.00	2005.00	2008.00
Set-up cost C_j^S (\$)	850	950	900

Types of problems that design refresh planning models will solve are summarized in Figure 5-5.

- Hardware obsolescence VS Software obsolescence: Hardware obsolescence management only takes place at time points of possible design refreshes during the life cycle of the system (several discrete time points); Software obsolescence management can take place at any time point during the life cycle of the system (the entire timeline).
- Deterministic model VS Probabilistic model: For the deterministic model, the date of obsolescence is deterministic; For the probabilistic model, the date of obsolescence follows certain probability distributions.
- One replacement VS Two replacements: One replacement means all parts need only one replacement during the life cycle of the system; Two replacements means some parts need two replacements (more than one replacement) during the life cycle of the system.

Examples of these problem types are provided in following sections.

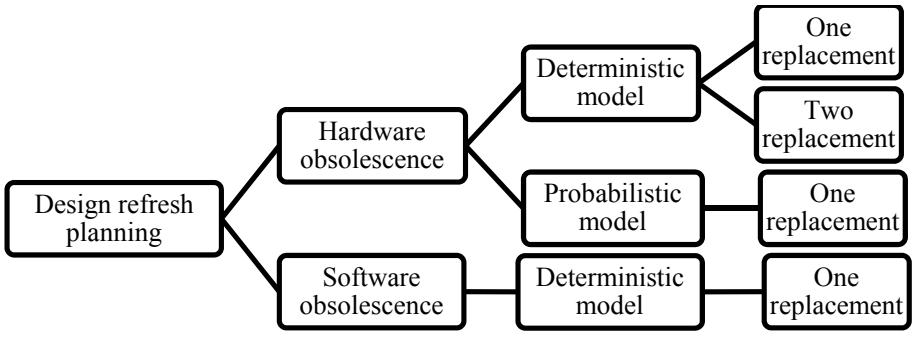


Figure 5-5: Types of problems that design refresh planning models will solve

5.2.3.1 Example 1: Deterministic Model for Hardware Obsolescence with One Replacement

The real situation constraint considered for hardware is that hardware can continuously be used in the system although they are obsolete. When hardware becomes obsolete, it does not need to be replaced immediately until the next design refresh. This makes the hardware obsolescence management simpler compared to other obsolescence management. The design refresh planning model for hardware is a special case of the general design refresh planning model: obsolescence can only be dealt with at the time points of design refreshes. During the time periods between two design refreshes, if obsolescence happens, it can wait until the next design refresh, and no short-term mitigation action needs to be taken. The objective function becomes:

$$Minimize\ C_{overall\ obso} = \sum_{j=1}^{n}\left(\sum_{i=1}^{m} C_{ij}^{R}\, x_{ij} + C_j^S y_j\right) \tag{5-17}$$

Where C_j^Ss are known; C_{ij}^R (design refresh cost for part type i at t_j) can be obtained from the original cost of this part type multiplied by a modifier:

$$C_{ij}^R = Q_i \times C_i \times M_{ij}^R \tag{5-18}$$

Where Q_i is the quantity of part type i; C_i is the original cost of part type i; M_{ij}^R is the modifier of part type i at the jth design refresh. M_{ij}^R is either provided by the user or defaulted using recurring cost multiplier data associated with various obsolescence resolution approaches (McDermott, Shearer, Tomczykowski, 1999; Singh, Sandborn, 2006). M_{ij}^R s used in the example are listed in Table 8-2 in Appendix B.

Deterministic model assumes that parts have deterministic obsolescence dates. According to obsolescence dates, 31 types of hardware components are obsolete during the life cycle of the ECU between 2000.00 and 2010.00 (not including those obsolete in 2010.00). The results about obsolete parts in each time period between possible design refreshes are summarized in Table 5-4.

Table 5-4: Obsolete parts in each time period

Time period	2000.00~2003.00	2003.00~2005.00	2005.00~2008.00	2008.00~2010.00
Obsolete parts	part_15, part_17, part_20, part_21	part_4~part_10, part_12~part_14, part_16, part_18, part_19, part_22~part_26, part_40~part_42	part_11, part_39	part_27, part_32~part_34
Total number	4	21	2	4

If part type i is replaced at the jth design refresh, its obsolescence date d_i will be updated to the new date D_{ij}^R according to the equation (5-13). D_{ij}^R can be calculated using the following equation (Singh, Sandborn, 2006):

$$D_{new} = D_{old} + L\left[\frac{I_o - I_R}{I_o - I_I}\right] \tag{5-19}$$

Where,

D_{new} is new obsolescence date for the replacement part;

D_{old} is old obsolescence date for the predecessor part;

I_o is life cycle code indicating part is obsolete (I_o=6);

I_I is life cycle code indicating part is in the emerging life cycle phase (I_I=1);

I_R is life cycle code of synthesized replacement part;

L is procurement life time (the amount of time the part is available for procurement from its original manufacturer).

The value of I_R is always 2 or 3 because sustainment-dominated systems are extremely risk adverse and may only select parts in the life stage of growth (I_R=2) or maturity (I_R=3). The value of the procurement life time of the replacement part can use that of the predecessor part. If I_R=2, according to the equation (5-19), no hardware component in Table 8-1 in Appendix A needs the second replacement after the first replacement. If I_R=3, part_15 needs the second replacement:

$$D_{new} = D_{old} + L\left[\frac{I_o - I_R}{I_o - I_I}\right] = 2001.75 + 10\left[\frac{6-3}{6-1}\right] = 2007.75 < 2008$$

In this section, I_R=2. All parts need only one replacement.

Design refresh planning models for hardware obsolescence with one replacement can be described as follows: If a hardware part type has been obsolete before a specific design refresh and has never been replaced in previous design refreshes, this part type has to be replaced in this design refresh. Other non-obsolete part types can also be replaced in this design refresh. All obsolete parts need to be replaced only once during the life cycle of the system. The model is to determine when to do design refresh and what parts are replaced at each design refresh for minimizing the overall obsolescence management cost. The model formulation is as follows:

$$Minimize\ C_{overall\ obso} = \sum_{j=1}^{n}\left(\sum_{i=1}^{m} C_{ij}^{R}\, x_{ij} + C_j^{S} y_j\right)$$

$$s.t. \sum_{i=1}^{m} x_{ij} \le M y_j \ (M \gg m)$$

$$\sum_{j=1}^{n} x_{ij} \ge 1$$

If $d_i \le t_j y_j$ and $x_{i1} = 0, x_{i2} = 0, \cdots x_{i(j-1)} = 0$, then x_{ij}=1 (*)

$$x_{ij}, y_j \in \{0,1\}$$

Without the constraint (*), the hardware part type which will be obsolete during the life cycle of the system can be replaced at any design refresh no matter that it is obsolete or not at that design refresh. The model becomes similar to the model of the fixed-charge facility location problem or the transportation problem. The model can be easily solved with the integer programming solver. The obtained optimal solution is: y_1=1, y_2=0, y_3 =1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{73} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,3} = x_{13,1} = x_{14,3} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,3} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{24,1} = x_{25,1} = x_{26,1} = x_{27,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = 1$, other x_{ij} =0;

minimum $C_{overall\ obso}$ =9182.276. That is, two design refreshes take place at 2003.00 and 2008.00 during the life cycle of the ECU, and part_8, part_9, part_11, part_12, part_13, part_16, part_18, part_19, part_20, part_22, part_25, part_27, part_32, part_33, part_34, part_42 are replaced at the 1st design refresh, part_4, part_5, part_6, part_7, part_10, part_14, part_15, part_17, part_21, part_23, part_24, part_26, part_39, part_40, part_41 are replaced at the 2nd design refresh. The minimum obsolescence management cost is 9182.276 dollars. (The values of parameters in the model provided by the ECU example are listed in Appendix C.)

If the constraint (*) is concluded, some decision variables' values in the above solution violate this constraint. For example, the obsolescence date of part_15 is 2001.75. Part_15 has been obsolete at the 1st design refresh at 2003.00, and it has never been replaced before because there is no design refresh before 2003.00. According to the constraint (*), part_15 has to be replaced at the design refresh at 2003.00. But in the above solution, part_15 is decided to be replaced at the 2nd design refresh at 2008.00 to achieve the minimum obsolescence management cost.

In the constraint (*), the first condition $d_i \le t_j y_j$ means the part type i has been obsolete before the design refresh at t_j. The second condition $x_{i1}=0, x_{i2}=0, \cdots x_{i(j-1)}=0$ means the part type i has never been replaced in previous design refreshes. If these two conditions are satisfied, the part type i has to be replaced at the design refresh at t_j, that is, x_{ij}=1. The constraint (*) in the form of "If ⋯ then⋯" can be transformed to be inequality with the following technical lemma for being used in the MILP model.

Lemma: "*If* $d_i \le t_j y_j$ *and* $x_{i1}=0, x_{i2}=0, \cdots x_{i(j-1)}=0$, *then* x_{ij}*=1"* $\Leftrightarrow$ "$(x_{i1}+x_{i2}+\cdots+x_{i(j-1)}+1)d_i \ge t_j y_j+\varepsilon+(\tilde{m}-\varepsilon)x_{ij}$ " ($\tilde{m}$ *is lower bound on* d_i-t_j, ε *is tolerance for constraint violation).*

Proof: $\because x_{i1}=0, x_{i2}=0, \cdots x_{i(j-1)}=0 \Leftrightarrow x_{i1}+x_{i2}+\cdots+x_{i(j-1)}=0$

$\therefore$ If $x_{i1}=0, x_{i2}=0, \cdots x_{i(j-1)}=0$, then $(x_{i1}+x_{i2}+\cdots+x_{i(j-1)}+1)d_i=d_i$. And if $d_i \le t_j y_j$, then $(x_{i1}+x_{i2}+\cdots+x_{i(j-1)}+1)d_i \le t_j y_j$.

If not all x in $x_{i1}, x_{i2}, \cdots x_{i(j-1)}$ equal to zero, then $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \geq 2d_i$

$\because d_i$ is the obsolescence date of the part type i which is obsolete during the life cycle of the system, and t_j is the possible time point for design refresh during the life cycle of the system.

$\therefore d_i$ and t_j have the same time range, i.e., $t_0 \leq d_i \leq t_0 + T$, and $t_0 \leq t_j \leq t_0 + T$

$\because t_0 \gg T$ (t_0 is the beginning time of the system life cycle, $t_0 > 1700$. Normally, t_0 is around 1800~2300. Too remote time is not considered. T is the life cycle of the system, $T < 300$. Normally, T is around 100~200.)

$\therefore 2d_i > t_j$ $(2d_i \geq 2t_0 > t_0 + T \geq t_j)$

$\therefore$ If not all x in $x_{i1}, x_{i2}, \cdots x_{i(j-1)}$ equal to zero, then $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \geq 2d_i > t_j \geq t_j y_j$, i.e., $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \nleq t_j y_j$

$\boldsymbol{\therefore d_i \leq t_j y_j}$ **and** $\boldsymbol{x_{i1} = 0, x_{i2} = 0, \cdots x_{i(j-1)} = 0 \Leftrightarrow (x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \leq t_j y_j}$

$\because$ If $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \leq t_j y_j$, then x_{ij}=1 $\Leftrightarrow$ If x_{ij}=0, then $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \nleq t_j y_j$, i.e., $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i > t_j y_j \Leftrightarrow$ If x_{ij}=0, then $\left(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1\right)d_i \geq t_j y_j + \varepsilon$

For $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \geq t_j y_j + \varepsilon + (\tilde{m} - \varepsilon)x_{ij}$ ($\tilde{m}$ is lower bound on $d_i - t_j$, ε is tolerance for constraint violation),

If x_{ij}=0, then $\left(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1\right)d_i \geq t_j y_j + \varepsilon$. That is what we wanted.

If x_{ij}=1, then $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \geq t_j y_j + \tilde{m}$.That is always true.

($\because (x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \geq d_i, -t_j y_j \geq -t_j$

$\therefore (x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i - t_j y_j \geq d_i - t_j$

$\therefore \left(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1\right)d_i - t_j y_j \geq d_i - t_j \geq \tilde{m})$

$\therefore$ "If x_{ij} =0, then $\left(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1\right)d_i \geq t_j y_j + \varepsilon$ " $\Leftrightarrow$ "$(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \geq t_j y_j + \varepsilon + (\tilde{m} - \varepsilon)x_{ij}$"

$\therefore$ **"If** $(\boldsymbol{x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \leq t_j y_j}$ **, then** $\mathbf{x_{ij}}$ **=1"** $\Leftrightarrow$ **"(** $\boldsymbol{x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \geq t_j y_j + \varepsilon + (\tilde{m} - \varepsilon)x_{ij}}$**"**

$\therefore$ "If $d_i \leq t_j y_j$ and $x_{i1} = 0, x_{i2} = 0, \cdots x_{i(j-1)} = 0$, then x_{ij}=1" $\Leftrightarrow$ "$(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \geq t_j y_j + \varepsilon + (\tilde{m} - \varepsilon)x_{ij}$" ($\tilde{m}$ is lower bound on $d_i - t_j$, ε is tolerance for constraint violation). **Done**.

Combining with the transformed constraint, the model formulation becomes:

$$Minimize\ C_{overall\ obso} = \sum_{j=1}^{n}\left(\sum_{i=1}^{m} C_{ij}^{R} x_{ij} + C_j^S y_j\right)$$

$$s.t. \sum_{i=1}^{m} x_{ij} \leq M y_j\ (M \gg m)$$

$$\sum_{j=1}^{n} x_{ij} \geq 1$$

$$(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \geq t_j y_j + \varepsilon + (\tilde{m} - \varepsilon)x_{ij} \qquad (*)$$

$$x_{ij}, y_j \in \{0,1\}$$

$\tilde{m}$ is lower bound on $d_i - t_j$, ε is tolerance for constraint violation

In the example of the ECU, minimum of d_i is 2000.00, maximum of t_j is 2010.00, so $\tilde{m}$ can be -10. ε can be 0.1. The transformed constraint (*) becomes:

$$(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)d_i \geq t_j y_j + 0.1 - 10.1x_{ij}$$

j =1, 2, or 3, because there are 3 possible time points for design refresh. The above constraint becomes three constraints:

$$d_i \geq t_1 y_1 + 0.1 - 10.1x_{i1}$$

$$(x_{i1} + 1)d_i \geq t_2 y_2 + 0.1 - 10.1x_{i2}$$

$$(x_{i1} + x_{i2} + 1)d_i \geq t_3 y_3 + 0.1 - 10.1x_{i3}$$

The values of parameters in the model provided by the example are listed in Appendix C. The model is solved with the integer programming solver. The obtained optimal solution is: y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{73} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{24,1} = x_{25,1} = x_{26,1} = x_{27,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = 1$, other x_{ij}=0; minimum $C_{overall\ obso}$ =9235.629. That is, two design refreshes take place at the time points of 2003.00 and 2008.00 during the life cycle of the ECU, and part_8, part_9, part_11, part_12, part_13, part_15, part_16, part_17, part_18, part_19, part_20, part_21, part_22, part_25, part_27, part_32, part_33, part_34, part_42 are replaced at the 1st design refresh, part_4, part_5, part_6, part_7, part_10, part_14, part_23, part_24, part_26, part_39, part_40, part_41 are replaced at the 2nd design refresh. The minimum obsolescence management cost is 9235.629 dollars.

Compared to the solution of the model without the constraint (*), there is no change in the number of design refreshes and the time when design refreshes take place. But part_15, part_17 and part_21 are replaced at the 1st design refresh at 2003.00 instead of the 2nd design refresh at 2008.00, because these three part types have been obsolete and have not been replaced before 2003.00, which is in accordance with the constraint (*). Because of the added constraint (*), the minimum obsolescence management cost obtained from the model also increases. Comparison of the solutions obtained from the models with or without the constraint (*) for the example of the ECU are shown in Figure 5-6.

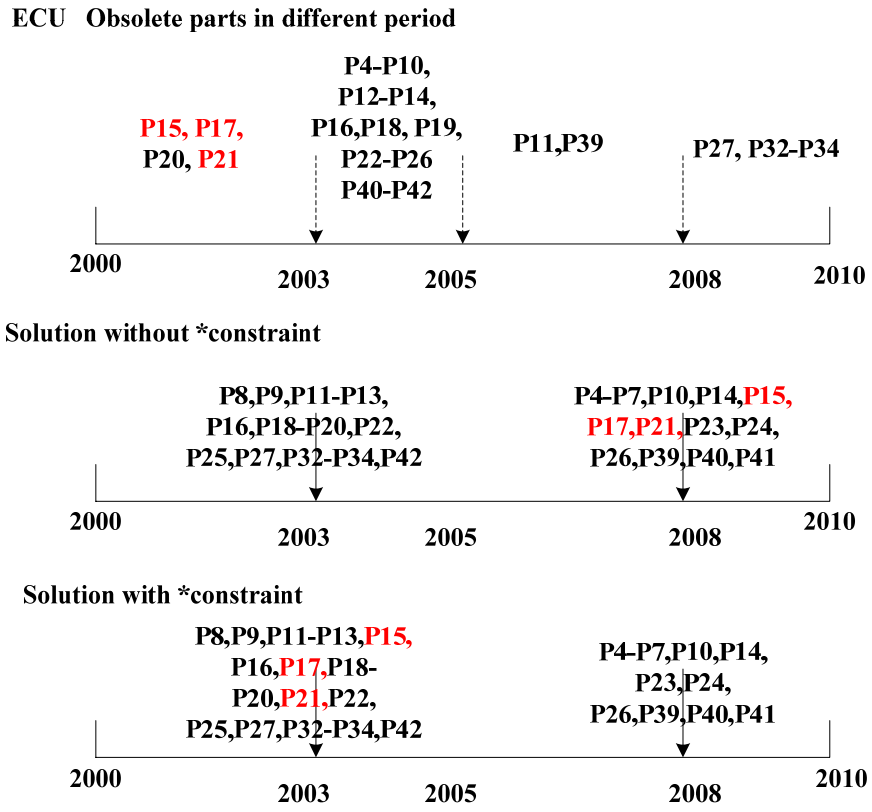


Figure 5-6: Comparison of solutions of models with or without constraint (*) for ECU

5.2.3.2 Example 2: Deterministic Model for Hardware Obsolescence with Two Replacements

The model for hardware obsolescence with two replacements means some hardware part types need two replacements during the life cycle of the system. It is the special case of some part types needing more than one replacement during the life cycle of the system. In the example of the ECU, if I_R=3, part_15 needs two replacements during the life cycle of the ECU.

The problem about two replacements (more than one replacement in general) is not easily solved by simply revising the previous model for hardware obsolescence with one replacement. But if there are not many part types in the system needing two replacements, the problem can be solved by dividing the problem into two sub-problems: one sub-problem for part types needing two replacements, the other sub-problem for left

part types needing one replacement. In the example of the ECU, only one part type part_15 needs two replacements. All possible ways of design refreshes for part_15 can be enumerated at first. Then for each way of design refresh for part_15, the obsolescence management cost for part_15 can be calculated. The minimum obsolescence management cost for other parts except part_15 can be obtained by solving the previous model for hardware obsolescence with one replacement not including part_15. The overall obsolescence management cost can be calculated by adding these two costs together. Finally, the overall obsolescence management costs for all possible ways of design refreshes for part_15 can be obtained and the minimum value can be found. The procedure is shown in Figure 5-7 (i=12 represents part_15):

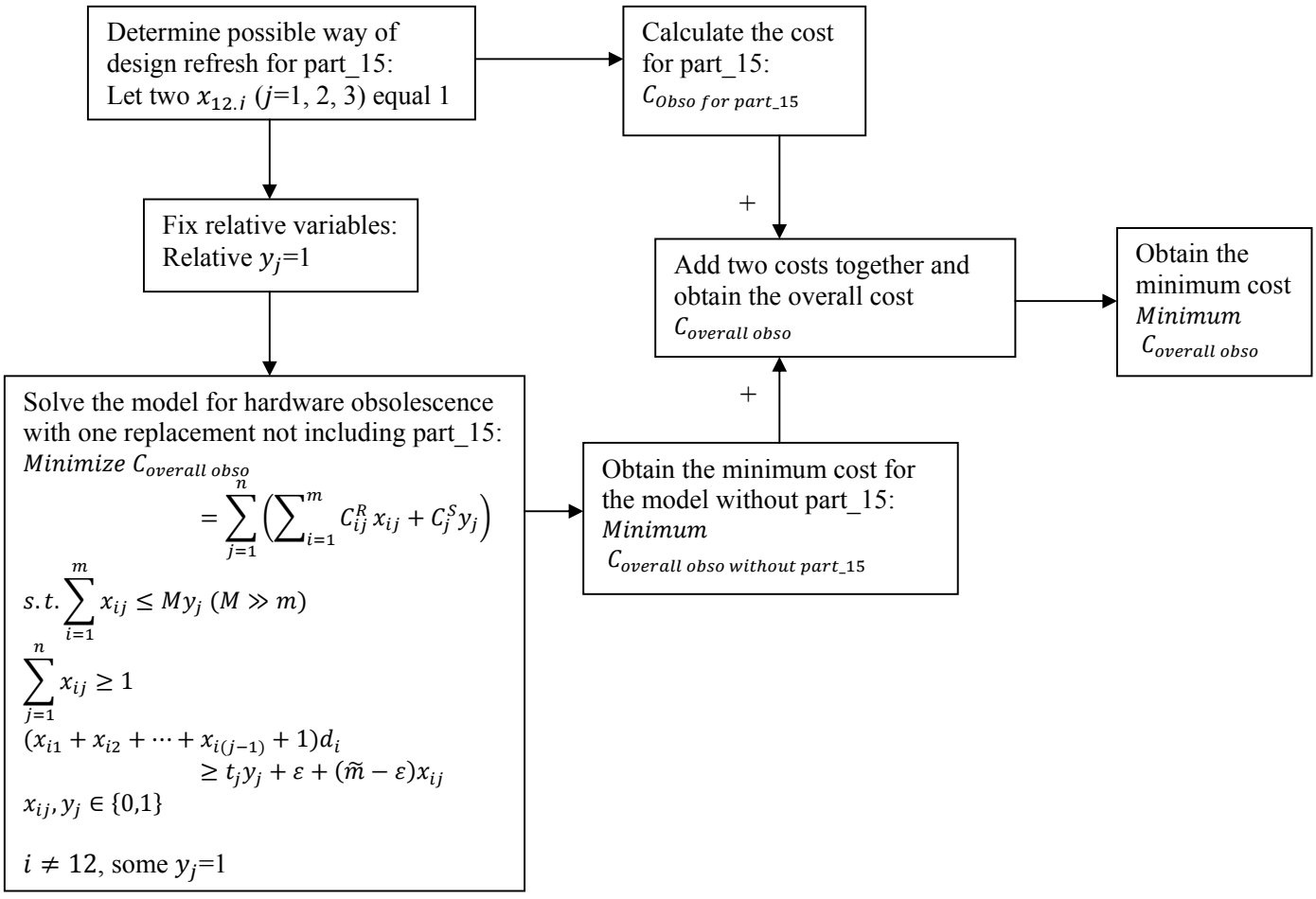


Figure 5-7: Procedure of solving problem about part_15 in ECU needing two replacements

Total costs for all possible ways of design fresh for part_15 are listed in Table 5-5. The final optimal solution is: Two design refreshes take place at the time points of 2003.00 and 2008.00 during the life cycle of the ECU, and part_8, part_9, part_11, part_12, part_13, part_16, part_17, part_18, part_19, part_20, part_21, part_22, part_25, part_27, part_32, part_33, part_34, part_42 are replaced at the 1st design refresh, part_4, part_5, part_6, part_7, part_10, part_14, part_23, part_24, part_26, part_39, part_40, part_41 are replaced at the 2nd design refresh. Part_15 are replaced at both 1st and 2nd design refresh. The minimum obsolescence management cost is 9474.324 dollars.

Table 5-5: Total costs for all possible ways of design refreshes for part_15

Possible $x_{12,j}$	Fixed y_j	$C_{Obso\ for\ part\_15}$	$C_{overall\ obso\ without\ part\_15}$	$C_{overall\ obso}$
$x_{12,1}$=1, $x_{12,2}$=1	y_1=1, y_2=1	630.18	9249.91	9880.09
$x_{12,1}$=1, $x_{12,3}$=1	y_1=1, y_3=1	529.7019	8944.622	**9474.324**
$x_{12,2}$=1, $x_{12,3}$=1	y_2=1, y_3=1	577.8685	9299.91	9877.779

5.2.3.3 Example 3: Probabilistic Model for Hardware Obsolescence with One Replacement

The probabilistic model assumes the obsolescence dates of parts follow certain probability distribution. In the example of the ECU, obsolescence dates of parts follow triangular distribution, as shown in Table 8-1 in Appendix A. In total 31 part types which are obsolete during the life cycle of the ECU between 2000.00 and 2010.00 (not including parts which are obsolete in 2010.00), part_10 and part_27 have 2 modes with certain probabilities of affecting the obsolescence management solution and cost because the probability distributions of their obsolescence dates fall into two continuous time periods divided by possible design refresh, part_34 has 3 modes with certain probabilities, and other parts have 1 mode with probability 1.

The calculation of probabilities for modes of single part type is shown using the example of part_10. The obsolescence date of part_10 has triangular distribution in the range (2002.66, 2003.325):

$$f(x) = 4.523(x - 2002.66)\ (2002.66 \leq x \leq 2003.325)$$

The range can be divided into two sub-ranges by the possible design refresh event at 2003.00: (2002.66, 2003.00) which belongs to (2000.00, 2003.00) and (2003.00, 2003.325) which belongs to (2003.00, 2005.00), as shown in Figure 5-8. Possible obsolescence dates of part_10 in the range (2002.66, 2003.00) have the same obsolescence management solution (mode 1). And possible obsolescence dates of part_10 in the range (2003.00, 2003.325) have the other same obsolescence management solution (mode 2). The possibility of mode 1 is:

$$P(2002.66 \leq x \leq 2003.00) = \int_{2002.66}^{2003.00} f(x)dx = 0.2614$$

The possibility of mode 2 is:

$$P(2003.00 \leq x \leq 2003.325) = 1 - P(2002.66 \leq x \leq 2003.00) = 0.7386$$

Obsolescence management solution modes of part_10, part_27 and part_34 and their probabilities are listed in Table 5-6. Other parts (e.g., part_4) have only 1 mode with probability 1 because their time ranges of possible obsolescence dates (e.g., (2003.52, 2004.4)) belong to the same time period between two possible design refreshes (e.g., (2003.00, 2005.00)).

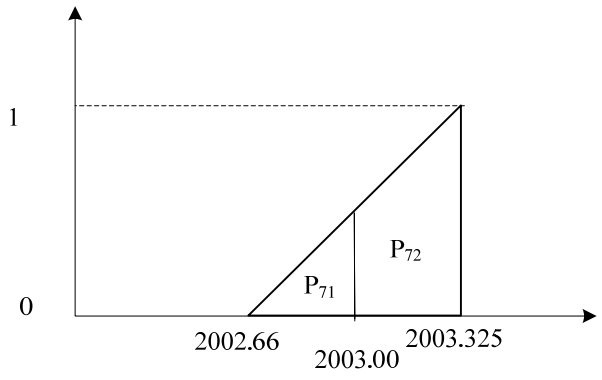


Figure 5-8: Obsolescence date of part_10 following triangular distribution

Table 5-6: Obsolescence management solution modes of parts and their probabilities

Part	Probability of modes
7(part_10)	$P_{71} = P(2002.66\sim2003)$=0.2614, $P_{72} = P(2003\sim2003.325)$=0.7386
24(part_27)	$P_{24,1} = P(2009.15\sim2010)$=0.1224, $P_{24,2} = P(2010\sim2011.58)$=0.8776
27(part_34)	$P_{27,1} = P(2007.02\sim2008)$=0.0606, $P_{27,2} = P(2008\sim2010)$=0.5, $P_{27,3} = P(2010\sim2011)$=0.4394

Combining 2 modes of part_10, 2 modes of part_24 and 3 modes of part_34 can create total 12 ($\binom{2}{1}\binom{2}{1}\binom{3}{1}$) modes of final obsolescence management models. Optimal solutions of these obsolescence management models and their probabilities are listed in Table 5-7. And the distribution of minimum obsolescence management costs is shown in Figure 5-9.

Table 5-7: Optimal solutions of 12 final obsolescence management models and their probabilities

	Probability	Decision variables	Objective
1	P_1=$P_{71}P_{24,1}P_{27,1}$ =0.0019	y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{71} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{24,1} = x_{25,1} = x_{26,1} = x_{27,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = 1$, other x_{ij}=0	minimum $C_{overall\ obso}$ = 9374.227
2	P_2=$P_{72}P_{24,1}P_{27,1}$ =0.0055	y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{73} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{24,1} = x_{25,1} = x_{26,1} = x_{27,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = 1$, other x_{ij}=0;	minimum $C_{overall\ obso}$ = 9235.629
3	P_3=$P_{71}P_{24,2}P_{27,1}$ =0.0139	y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{71} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{25,1} = x_{26,1} = x_{27,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = 1$, other x_{ij}=0;	minimum $C_{overall\ obso}$ = 9374.112
4	P_4=$P_{72}P_{24,2}P_{27,1}$ =0.0393	y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{73} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{25,1} = x_{26,1} = x_{27,1} = x_{28,3} = x_{29,3} = x_{30,3} =$	minimum $C_{overall\ obso}$ = 9235.513

		$x_{31,1} = 1$, other x_{ij}=0;	
5	P_5=$P_{71}P_{24,1}P_{27,2}$ =0.0160	y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{71} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{24,1} = x_{25,1} = x_{26,1} = x_{27,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = 1$, other x_{ij}=0;	minimum $C_{overall\ obso}$ = 9374.227
6	P_6=$P_{72}P_{24,1}P_{27,2}$ =0.0452	y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{73} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{24,1} = x_{25,1} = x_{26,1} = x_{27,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = 1$, other x_{ij}=0;	minimum $C_{overall\ obso}$ = 9235.629
7	P_7=$P_{71}P_{24,2}P_{27,2}$ =0.1147	y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{71} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{25,1} = x_{26,1} = x_{27,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = 1$, other x_{ij}=0;	minimum $C_{overall\ obso}$ = 9374.112
8	P_8=$P_{72}P_{24,2}P_{27,2}$ =0.3241	y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{73} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{25,1} = x_{26,1} = x_{27,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = 1$, other x_{ij}=0;	minimum $C_{overall\ obso}$ = 9235.513
9	P_9=$P_{71}P_{24,1}P_{27,3}$ =0.0141	y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{71} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{24,1} = x_{25,1} = x_{26,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = 1$, other x_{ij}=0;	minimum $C_{overall\ obso}$ = 9370.294
10	P_{10} =$P_{72}P_{24,1}P_{27,3}$ =0.0397	y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{73} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{24,1} = x_{25,1} = x_{26,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = 1$, other x_{ij}=0;	minimum $C_{overall\ obso}$ = 9231.695
11	P_{11} =$P_{71}P_{24,2}P_{27,3}$ =0.1008	y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{71} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} =$	minimum $C_{overall\ obso}$ = 9370.178

		$x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{25,1} = x_{26,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = 1$, other x_{ij}=0;	
12	P_{12} $=P_{72}P_{24,2}P_{27,3}$ =0.2848	y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{73} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{25,1} = x_{26,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = 1$, other x_{ij}=0;	minimum $C_{overall\ obso}$ = 9231.580

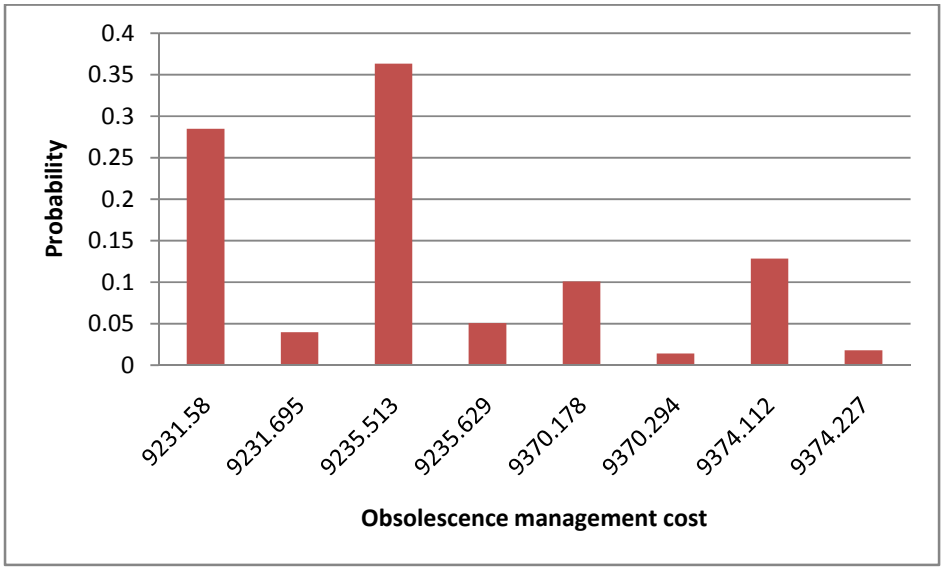


Figure 5-9: Distribution of minimum obsolescence management costs

As the minimum obsolescence management costs under different modes and their probabilities are known, the finial optimal obsolescence management cost is the expectation of these costs:

$E(minimal\ C_{overall\ obso}) =$
$\sum_{i=1}^{12}(minimal\ C_{overall\ obso,i})P_i$=9231.58×0.2848+9231.695×0.0397+9235.513×0.3634 +9235.629×0.0507+9370.178×0.1008+9370.294×0.0141+9374.112×0.1286+9374.227 ×0.0179=9270.029

5.2.3.4 Example 4: Deterministic Model for Software Obsolescence with One Replacement

Software are not allowed to be used in the system any longer when they are obsolete, and they need to be updated and replaced immediately. During the time period between two design refreshes, when software go obsolete, short-term mitigation approaches need to be applied. The design refresh planning model for software is more complicated than the model for hardware: Obsolescence is not only dealt with at design refreshes, but also during the time periods between design refreshes.

Besides hardware components, the ECU has a number of software components which operate the hardware. Types of representative software components are selected from the system for demonstration of design refresh planning, and their information related to obsolescence management are listed in Table 8-1 in Appendix A. According to the equation (5-19) for calculating the new obsolescence date for the replacement part, all software part types need only one replacement.

Design refresh planning models for software obsolescence with one replacement can be described as follows: Obsolescence is not only dealt with at the time points of design refreshes like design refresh planning for hardware, but also during the time periods between design refreshes. Obsolete parts and non-obsolete parts which are forecasted to be obsolete in the near future can be replaced at design refreshes. If parts go obsolete during the time periods between design refreshes, short-term mitigation approaches need to be applied on these obsolete parts and costs need to be accounted. All obsolete parts need to be replaced only once during the life cycle of the system. The model is to determine when to do design refresh, what parts are replaced at each design refresh, and what parts are replaced using short-term mitigation approaches when the design refresh is not available for minimizing the overall obsolescence management costs. The model formulation is as follows:

$$Minimize\ C_{overall\ obso} = \sum_{j=1}^{n}\left(\sum\nolimits_{i=1}^{m} C_{ij}^{R}\, x_{ij} + C_j^S y_j\right) + \sum_{k=1}^{n+1}\sum\nolimits_{i=1}^{m} C_{ik}^{M}\, z_{ik}$$

$$s.t. \sum_{i=1}^{m} x_{ij} \le M y_j\ (M \gg m)$$

$$\sum_{j=1}^{n} x_{ij} + \sum_{k=1}^{n+1} z_{ik} \ge 1$$

If $t_{j-1} < d_i < t_j$ and $x_{i1} = 0, x_{i2} = 0, \cdots x_{i(j-1)} = 0$, then z_{ik}=1 (*)

$$x_{ij}, y_j, z_{ik} \in \{0,1\}$$

In the objective function, C_j^S s are known, C_{ij}^R can be calculated using equation (5-18), and C_{ik}^M can be calculated in the similar way:

$$C_{ik}^M = Q_i \times C_i \times M_{ik}^M \quad (5\text{-}20)$$

Where Q_i is the quantity of part type i; C_i is the original cost of part type i; M_{ij}^M is the modifier of part type i at the kth time period between two possible design refreshes. The values of M_{ij}^R and M_{ij}^M of software components in the ECU are provided in Table 8-3 in Appendix B.

In the constraint (*), the first constraint $t_{j-1} < d_i < t_j$ means the part type i will go obsolete during the time period T_k ($T_k = t_j - t_{j-1}$). The second constraint $x_{i1} = 0, x_{i2} = 0, \cdots x_{i(j-1)} = 0$ means the part type i has never been replaced in previous design refreshes. If these two conditions are satisfied, the part type i has to be replaced using short-mitigation approaches during the time period T_k, that is, z_{ik}=1. The constraint (*) in the form of "If ⋯ then⋯" can be transformed to be inequality with the following technical lemma for being used in the MILP model.

Lemma: *"If* $t_{-1} < d_i < t_j$ *and* $x_{i1} = 0, x_{i2} = 0, \cdots x_{i(j-1)} = 0$, *then* z_{ik}*=1"* $\Leftrightarrow$ "$(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \geq (t_{j-1} + t_j)d_i + \widetilde{m}\, z_{ik}$ *" (* $\widetilde{m}$ *is lower bound on* $\left(t_{j-1}t_j + d_i^2\right) - (t_{j-1} + t_j)d_i$*).*

Proof: $\because t_{j-1} < d_i < t_j \Leftrightarrow \left(t_{j-1} - d_i\right)\left(t_j - d_i\right) < 0 \Leftrightarrow t_{j-1}t_j + d_i^2 < (t_{j-1} + t_j)d_i$

And $\because x_{i1} = 0, x_{i2} = 0, \cdots x_{i(j-1)} = 0 \Leftrightarrow x_{i1} + x_{i2} + \cdots + x_{i(j-1)} = 0$

$\therefore$ If $x_{i1} = 0, x_{i2} = 0, \cdots x_{i(j-1)} = 0$, then $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) = t_{j-1}t_j + d_i^2$. And if $t_{j-1}t_j + d_i^2 < (t_{j-1} + t_j)d_i$, then ($x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) < (t_{j-1} + t_j)d_i$.

If not all x in $x_{i1}, x_{i2}, \cdots x_{i(j-1)}$ equal to zero, then ($x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \geq 2(t_{j-1}t_j + d_i^2)$.

$\because d_i$ is the obsolescence date of the part type i which is obsolete during the life cycle of the system, and t_{j-1}, t_j are the possible time points for design refresh during the life cycle of the system.

$\therefore d_i, t_{j-1}, t_j$ have the same time range, i.e., $t_0 \leq d_i, t_{j-1}, t_j \leq t_0 + T$

$\therefore 2t_0^2 \leq t_{j-1}t_j + d_i^2, (t_{j-1} + t_j)d_i \leq 2(t_0 + T)^2$

$\because t_0 \gg T$ $(t_0 > 1700, T < 300)$

$\therefore \left(\sqrt{2} - 1\right)t_0 > T$

$\therefore 2t_0^2 > (t_0 + T)^2$

$\therefore 2(t_{j-1}t_j + d_i^2) \geq (t_{j-1} + t_j)d_i$ $(2\left(t_{j-1}t_j + d_i^2\right) \geq 4t_0^2 > 2(t_0 + T)^2 \geq (t_{j-1} + t_j)d_i)$

∴ If not all x in $x_{i1}, x_{i2}, \cdots x_{i(j-1)}$ equal to zero, then $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \geq 2(t_{j-1}t_j + d_i^2) \geq (t_{j-1} + t_j)d_i$, i.e., $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \not< (t_{j-1} + t_j)d_i$

∴ **“ $t_{j-1} < d_i < t_j$ and $x_{i1} = 0, x_{i2} = 0, \cdots x_{i(j-1)} = 0$ ” ⇔ “ $t_{j-1}t_j + d_i^2 < (t_{j-1} + t_j)d_i$ and $x_{i1} = 0, x_{i2} = 0, \cdots x_{i(j-1)} = 0$ ” ⇔ “ $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) < (t_{j-1} + t_j)d_i$”**

∵ “If $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) < (t_{j-1} + t_j)d_i$, then z_{ik} =1” ⇔ “If z_{ik}=0, then$(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \not< (t_{j-1} + t_j)d_i$, i.e., $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \geq (t_{j-1} + t_j)d_i$”

For $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \geq (t_{j-1} + t_j)d_i + \tilde{m}\, z_{ik}$ ($\tilde{m}$ is lower bound on $(t_{j-1}t_j + d_i^2) - (t_{j-1} + t_j)d_i$),

If z_{ik}=*0*, then $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \geq (t_{j-1} + t_j)d_i$. That is what we want.

If z_{ik}=1, then $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \geq (t_{j-1} + t_j)d_i + \tilde{m}$. That is always true. (∵ $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) - (t_{j-1} + t_j)d_i \geq (t_{j-1}t_j + d_i^2) - (t_{j-1} + t_j)d_i \geq \tilde{m}$)

∴ “If z_{ik} =0, then $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \geq (t_{j-1} + t_j)d_i$ ” ⇔ “$(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \geq (t_{j-1} + t_j)d_i + \tilde{m}\, z_{ik}$”

∴ **“If $(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) < (t_{j-1} + t_j)d_i$, then z_{ik} =1” ⇔ “$(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \geq (t_{j-1} + t_j)d_i + \tilde{m}\, z_{ik}$”**

∴ “If $t_{j-1} < d_i < t_j$ and $x_{i1} = 0, x_{i2} = 0, \cdots x_{i(j-1)} = 0$, then z_{ik}=1” ⇔ “$(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \geq (t_{j-1} + t_j)d_i + \tilde{m}\, z_{ik}$” ($\tilde{m}$ is lower bound on $(t_{j-1}t_j + d_i^2) - (t_{j-1} + t_j)d_i$). **Done**.

Combining with the transformed constraint, the model formulation becomes:

$$Minimize\ C_{overall\ obso} = \sum_{j=1}^{n}\left(\sum_{i=1}^{m} C_{ij}^{R}\, x_{ij} + C_{j}^{S} y_{j}\right) + \sum_{k=1}^{n+1}\sum_{i=1}^{m} C_{ik}^{M}\, z_{ik}$$

$$s.t. \sum_{i=1}^{m} x_{ij} \le M y_{j}\ (M \gg m)$$

$$\sum_{j=1}^{n} x_{ij} + \sum_{k=1}^{n+1} z_{ik} \ge 1$$

$$(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1)(t_{j-1}t_j + d_i^2) \ge (t_{j-1} + t_j)d_i + \tilde{m}\, z_{ik} \qquad (*)$$

$$x_{ij}, y_j, z_{ik} \in \{0,1\}$$

$\tilde{m}$ is lower bound on $\left(t_{j-1}t_j + d_i^2\right) - (t_{j-1} + t_j)d_i$

In the example of the ECU, minimums of d_i,t_{j-1},t_j are all 2000, and maximums of d_i,t_{j-1},t_j are all 2010, so minimum of $\left(t_{j-1}t_j + d_i^2\right)$ is 8000000, maximum of $(t_{j-1} + t_j)d_i$ is 8080200, so $\tilde{m}$ can be -80200. The transformed constraint (*) becomes:

$$\left(x_{i1} + x_{i2} + \cdots + x_{i(j-1)} + 1\right)\left(t_{j-1}t_j + d_i^2\right) \ge \left(t_{j-1} + t_j\right)d_i - 80200\, z_{ik}$$

j=1, 2, or 3, because there are 3 possible time points for design refreshes. The above constraint becomes four constraints:

$$(t_0 t_1 + d_i^2) \ge (t_0 + t_1)d_i - 80200\, z_{i1}$$

$$(x_{i1} + 1)(t_1 t_2 + d_i^2) \ge (t_1 + t_2)d_i - 80200\, z_{i2}$$

$$(x_{i1} + x_{i2} + 1)(t_2 t_3 + d_i^2) \ge (t_2 + t_3)d_i - 80200\, z_{i3}$$

$$(x_{i1} + x_{i2} + x_{i3} + 1)(t_3 t_4 + d_i^2) \ge (t_3 + t_4)d_i - 80200\, z_{i4}$$

The values of parameters in the model provided by the ECU example are listed in Appendix D. The model is solved with the integer programming solver. The obtained optimal solution is: $z_{13} = z_{22} = z_{32} = z_{41} = z_{51} = z_{62} = z_{74} = z_{83} = z_{92} = z_{10,2} = 1$, other z_{ik}, x_{ij}, y_j =0; minimum $C_{overall\ obso}$ =2994.855. That is, all parts are replaced with short-term mitigation approaches during the time period between possible design refreshes when they go obsolete. No design refresh takes place because of its high set-up cost. The minimum obsolescence management cost is 2994.855 dollars.

5.2.3.5 Example 5: Design Refresh Planning for ECU System

The design refresh planning model for the ECU system combines the model for hardware obsolescence (Section 5.2.3.1) and the model for software obsolescence (Section 5.2.3.4). Let I_R=2, then all parts including hardware and software need only one replacement. It is assumed that parts have specific obsolescence dates, so the model is deterministic. The model formulation is as follows:

$$Minimize\ C_{overall\ obso} = \sum_{j=1}^{n}\left(\sum_{i=1}^{m=41} C_{ij}^{R}\, x_{ij} + C_j^{S} y_j\right) + \sum_{k=1}^{n+1}\sum_{i=32}^{m=41} C_{ik}^{M}\, z_{ik}$$

$$s.t. \sum_{i=1}^{m=41} x_{ij} \le M y_j\ (M \gg m)$$

$$\sum_{j=1}^{n} x_{ij} \ge 1\ (i = 1,2,\cdots 31)$$

$$d_i \ge t_1 y_1 + 0.1 - 10.1 x_{i1} (i = 1,2,\cdots 31)$$

$$(x_{i1} + 1) d_i \ge t_2 y_2 + 0.1 - 10.1 x_{i2} (i = 1,2,\cdots 31)$$

$$(x_{i1} + x_{i2} + 1) d_i \ge t_3 y_3 + 0.1 - 10.1 x_{i3} (i = 1,2,\cdots 31)$$

$$\sum_{j=1}^{n} x_{ij} + \sum_{k=1}^{n+1} z_{ik} \ge 1 (i = 32,33,\cdots 41)$$

$$(t_0 t_1 + d_i^2) \geq (t_0 + t_1) d_i - 80200\, z_{i1} (i = 32{,}33, \cdots 41)$$

$$(x_{i1} + 1)(t_1 t_2 + d_i^2) \geq (t_1 + t_2) d_i - 80200\, z_{i2} (i = 32{,}33, \cdots 41)$$

$$(x_{i1} + x_{i2} + 1)(t_2 t_3 + d_i^2) \geq (t_2 + t_3) d_i - 80200\, z_{i3} (i = 32{,}33, \cdots 41)$$

$$(x_{i1} + x_{i2} + x_{i3} + 1)(t_3 t_4 + d_i^2) \geq (t_3 + t_4) d_i - 80200\, z_{i4} (i = 32{,}33, \cdots 41)$$

$$x_{ij}, y_j, z_{ik} \in \{0{,}1\}$$

The values of parameters in the model provided by the ECU example are listed in Appendix E. The model is solved with the integer programming solver. The obtained optimal solution is: y_1=1, y_2=0, y_3=1; $x_{13} = x_{23} = x_{33} = x_{43} = x_{51} = x_{61} = x_{73} = x_{81} = x_{91} = x_{10,1} = x_{11,3} = x_{12,1} = x_{13,1} = x_{14,1} = x_{15,1} = x_{16,1} = x_{17,1} = x_{18,1} = x_{19,1} = x_{20,3} = x_{21,3} = x_{22,1} = x_{23,3} = x_{24,1} = x_{25,1} = x_{26,1} = x_{27,1} = x_{28,3} = x_{29,3} = x_{30,3} = x_{31,1} = x_{32,1} = z_{33,2} = x_{34,1} = z_{35,1} = z_{36,1} = x_{37,1} = x_{38,3} = x_{39,1} = z_{40,2} = x_{41,1} = 1$, other x_{ij}, z_{ik}=0; minimum $C_{overall\ obso}$ =11968.88. That is, two design refreshes take place at the time points of 2003.00 and 2008.00 during the life cycle of the ECU, and hardware part_8, part_9, part_11, part_12, part_13, part_15, part_16, part_17, part_18, part_19, part_20, part_21, part_22, part_25, part_27, part_32, part_33, part_34, part_42, software part_51, part_53, part_56, part_58, part_60 are replaced at the 1st design refresh. Hardware part_4, part_5, part_6, part_7, part_10, part_14, part_23, part_24, part_26, part_39, part_40, part_41, software part_57 are replaced at the 2nd design refresh. Software part_52, part_54, part_55, part_59 are replaced using short-term mitigation approach. The minimum obsolescence management cost is 11968.88 dollars.

Compared to the independent models for hardware obsolescence (Section 5.2.3.1) and software obsolescence (Section 5.2.3.4), there is no change in the solutions for hardware parts. Some software parts can be dealt with at design refreshes instead of using short-term mitigation approaches because they can share high set-up costs with hardware parts. The minimum obsolescence management cost 11968.88 is less than the sum of 9235.639 and 2994.855, where 9235.639 is the minimum obsolescence management cost for only

hardware obsolescence (Section 5.2.3.1), and 2994.855 is the minimum obsolescence management cost for only software obsolescence (Section 5.2.3.4).

5.3 Conclusion

In this chapter, the decision models for proactive and strategic obsolescence management are developed. First, an obsolescence forecasting method based on the product life cycle is presented. Then, with the obsolescence information input from forecasting, various design refresh planning models are provided for strategically solving the electronic system's hardware and software obsolescence problems. Decision models are an important part of the obsolescence management information system, which will be introduced in Chapter 6.

Chapter 6 System Development and Evaluation

This chapter discusses the information system development and evaluation for obsolescence management by synthesizing the methods and results from previous three chapters. Section 6.1 introduced the overview of the system. Section 6.2 provides the results about the ontologies and rules realized in Protégé. The method for system evaluation is presented in Section 6.3.

6.1 Overview of System

Research work of the previous three chapters supports DMSMS obsolescence management from three aspects: knowledge representation, information integration, and decision support. Knowledge representation establishes a general DMSMS obsolescence knowledge scheme for information sharing, reuse and collaboration on obsolescence issues. Information integration proposes a methodology for integrating heterogeneous data sources in existing DMSMS obsolescence management tools into the general DMSMS obsolescence knowledge scheme. Decision support provides optimal models of design refresh planning to realize proactive and strategic management for DMSMS obsolescence based on clear, consistent, and complete information. The approach and methods used are summarized in Figure 6-1.

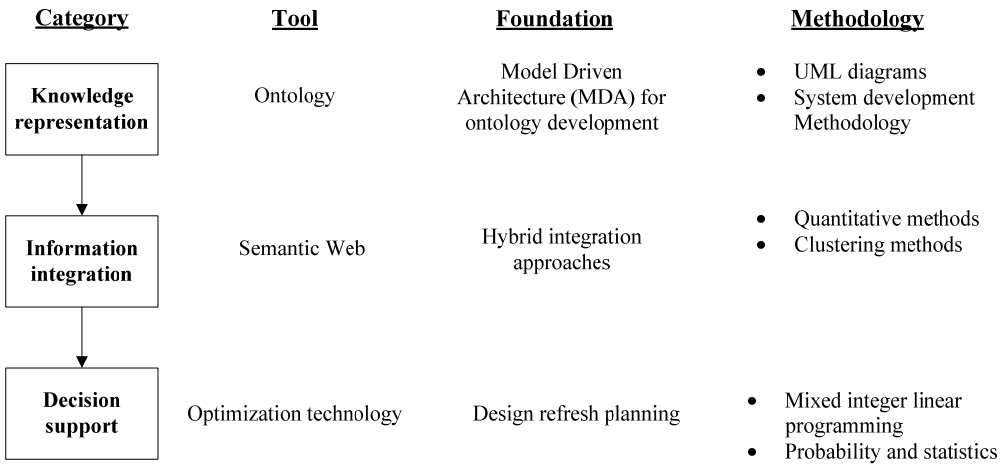


Figure 6-1: Summary of methods used in knowledge representation and decision support for DMSMS obsolescence

Results of research work can be utilized in building the DMSMS obsolescence management information system. The architecture of the information system is shown in Figure 6-2. Obsolescence ontologies from knowledge representation consist of the obsolescence knowledge base in charge of multiple data sources with the approach of information integration for obsolescence data management. Models of design refresh planning are decision support models for obsolescence management. Obsolescence ontologies for knowledge representation, and rules and queries for information integration are developed with OWL (Web Ontology Language), SWRL (Semantic Web Rule Language) and SQWRL (Semantic Query-enhanced Web Rule Language) in ontology editor Protégé. Design fresh planning models for decision support are solved with an integer programming solver based on data sources under the management of obsolescence ontologies. Results of the system development are provided in the next section.

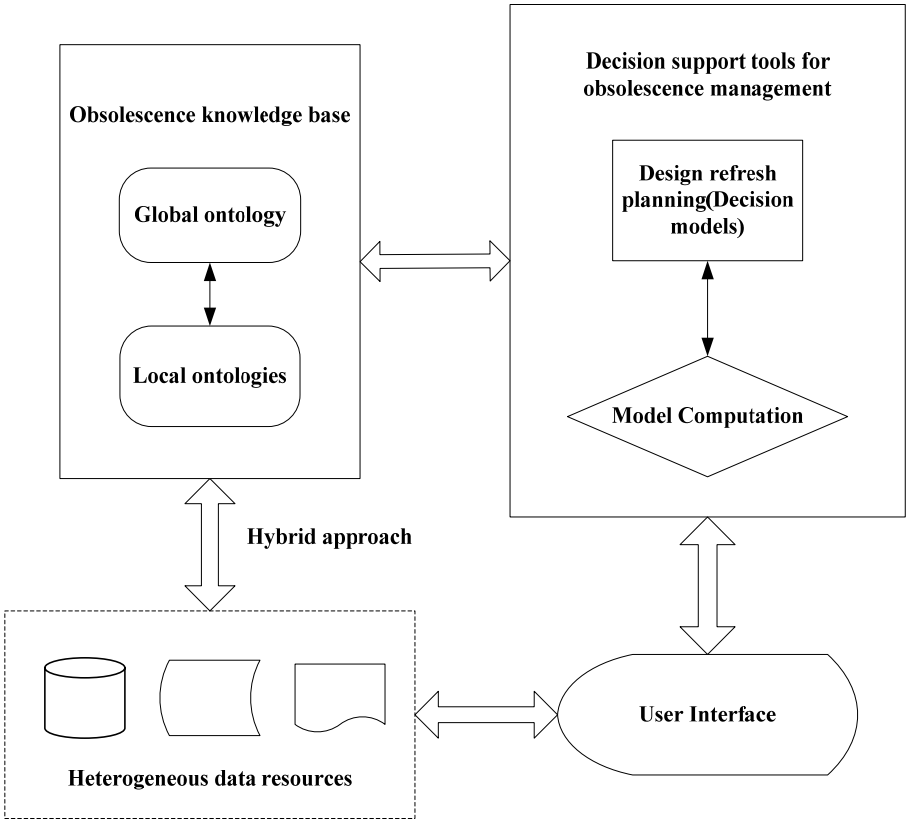


Figure 6-2: Architecture of obsolescence management information system

6.2 Results in Protégé

The obsolescence ontology is built in the ontology editor Protégé. Parts of the results related to knowledge representation, information integration and decision support are provided in the following subsections.

6.2.1 Results of Knowledge Representation

Protégé has a plug-in UML back end, which allows the sharing of UML models between UML tools and Protégé by using the UML XMI (XML Metadata Interchange) format. The use case diagram and the class diagram for DMSMS obsolescence management (Figure 3-2 and Figure 3-3) are developed in the UML back end. Obsolescence ontologies can be created automatically according to the class diagram. The created ontologies need further refinement applying the procedure of ontology development to form knowledge representation scheme for the domain of obsolescence. Classes of obsolescence ontologies are arranged in a hierarchical structure, as shown in Figure 6-3.

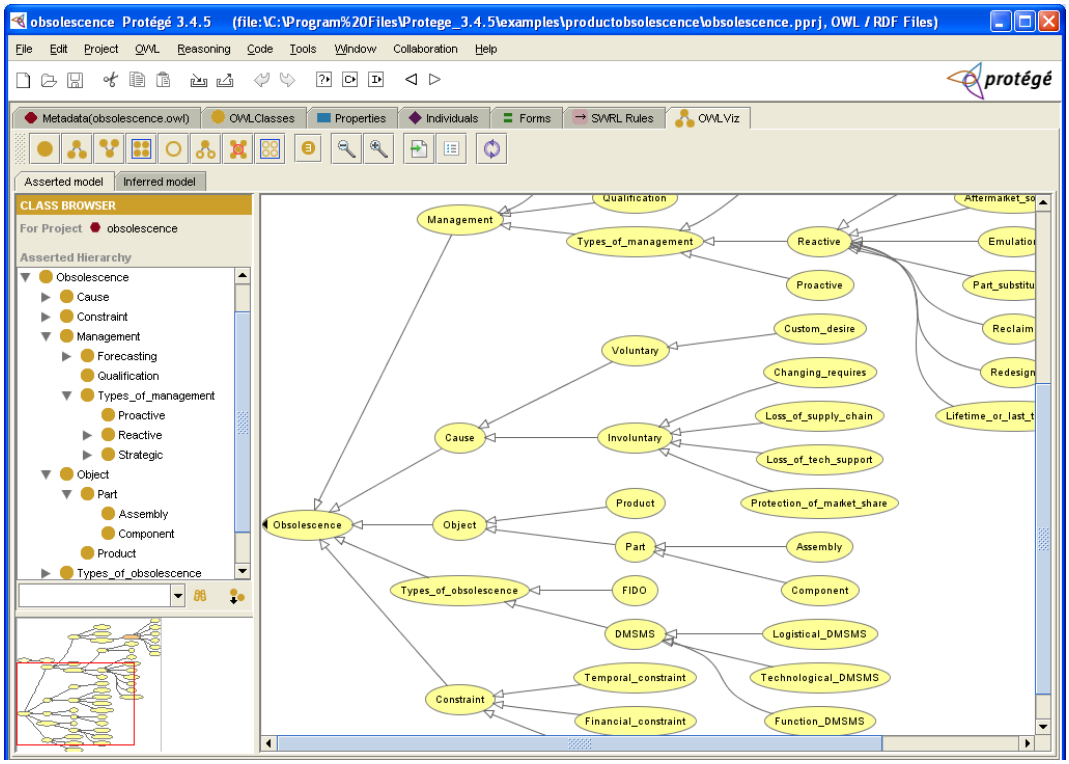


Figure 6-3: Class hierarchy of obsolescence ontologies

Instance of obsolescence classes can be created by specifying values of slots for data management related to the obsolescence problem. The example of DMSMS obsolescence is an electronic engine control unit (ECU). Figure 6-4 shows an instance of the class Part, which is a hardware part of the ECU system. It is a 'Microcircuit' provided by 'Vishay'. Its life time is '10.0' years and the obsolescence date is '2004.4'. It costs '2.9' dollars. The ECU needs '3' of these parts for building the system. Information of other parts in the ECU can be managed in the same way. The owner of the ECU can use this information for managing DMSMS obsolescence during the life cycle of the system.

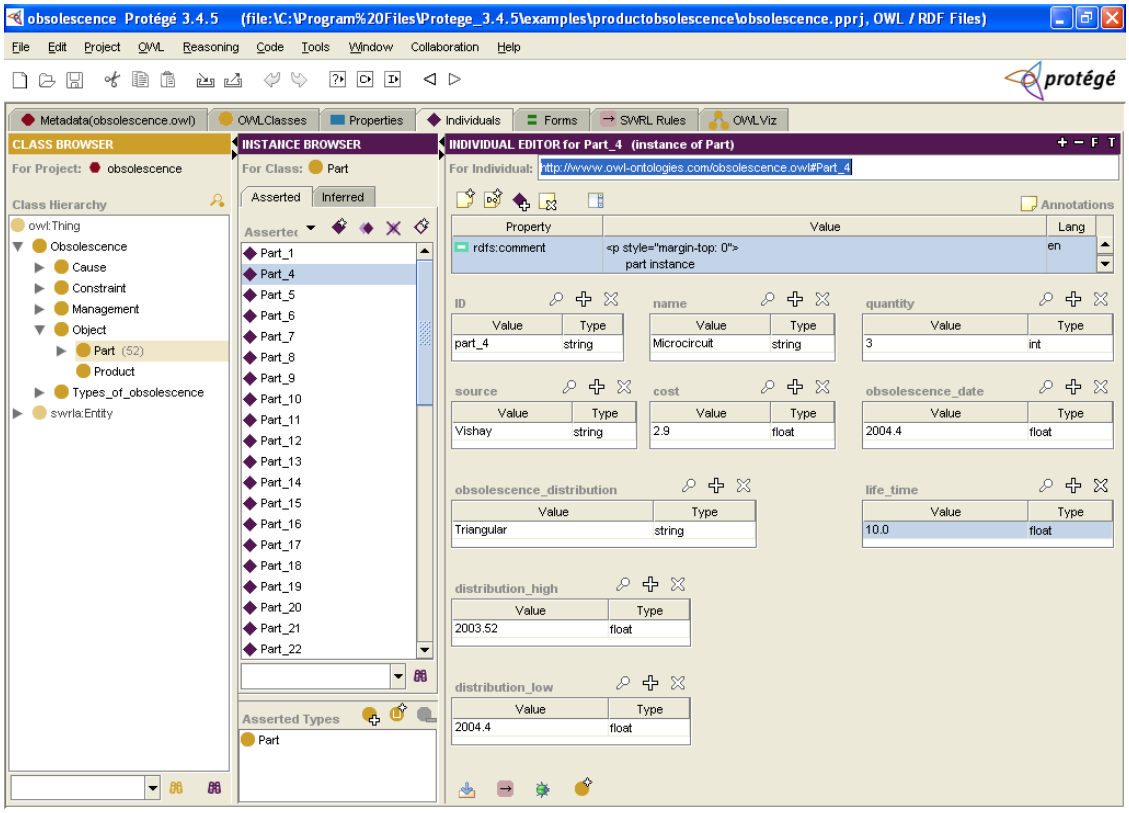


Figure 6-4: Instance of obsolescence ontologies

6.2.2 Results of Information Integration

OWL is a Semantic Web language used in the development of obsolescence ontologies for solving the problem of semantic heterogeneity existing in web information. OWL is powerful in logic description with restrictions and logic operators. Figure 6-5 shows the logic descriptions of product life cycle applying the covering axiom and BOM of an

electronic system applying the closure axiom. Product life cycle is described with its life stages which are introduction, growth, maturity, saturation, decline, and phase-out. The BOM of an electronic system shows it consists of hardware components microprocessor, memory, passive components and software components operating system, data management software, application software.

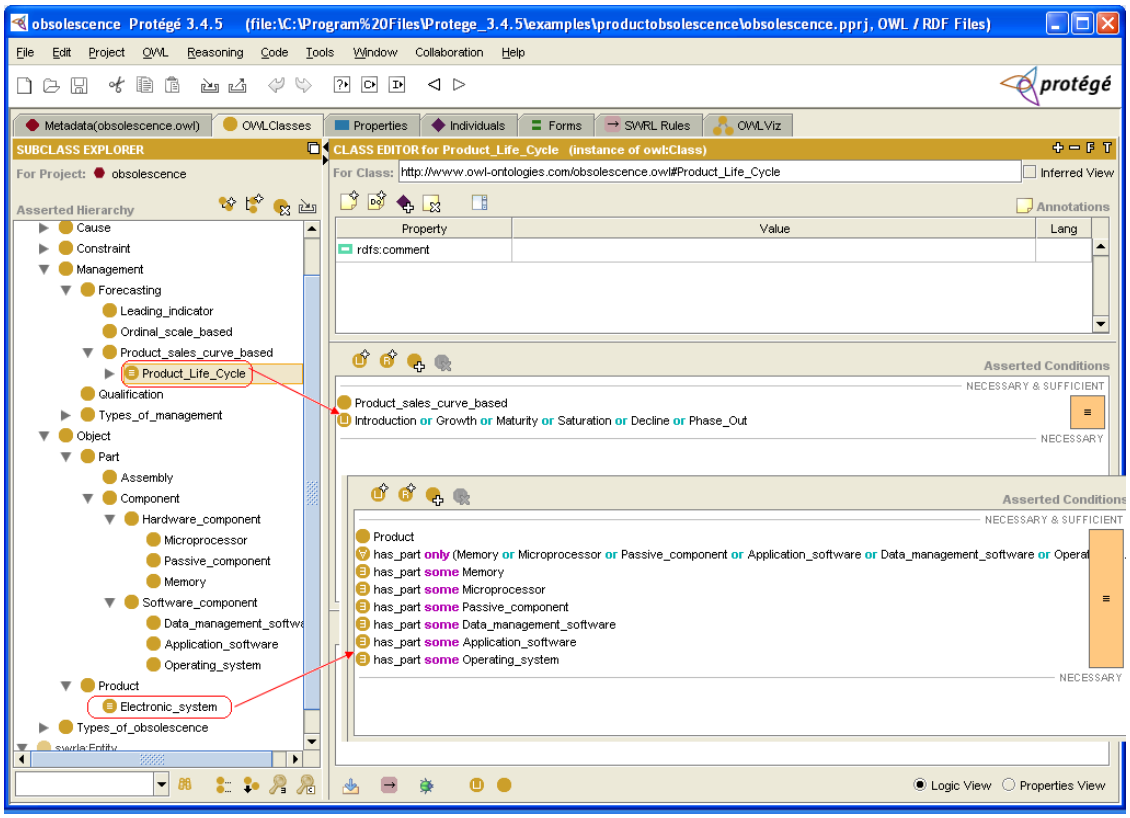


Figure 6-5: Logic description for obsolescence ontologies

The ontology-based hybrid approach for information integration uses the global ontology for providing a shared vocabulary for the specification of the semantics of knowledge in a domain, and local ontologies describing structures of multiple data sources. The global obsolescence ontology scheme has been developed in a systematic way by combining the use of UML diagrams and is realized with OWL, as shown in Figure 6-3, 6-4, 6-5. Three local ontologies (S_1, S_2, S_3) describing three local data sources are imported for integration with the global ontology, as shown in Figure 6-6. The class prefix is used to indicate the ontology to which it belongs. For example, S_3: Amplifier indicates the class

Amplifier belongs to the local ontology S_3. The classes without prefix are from the global ontology. The global ontology and local ontologies have URL for identifying themselves. For example, the global ontology's URL is http://www.owl-ontologies.com/Obsolescence.owl#. Three local data sources have been introduced in Chapter 4. The local ontology S_1 has classes Product, Product_Category, Supplier. The local ontology S_2 has classes Component, System, Manufacturer, Obsolescence mitigation approaches. And the local ontology S_3 has the class Amplifier.

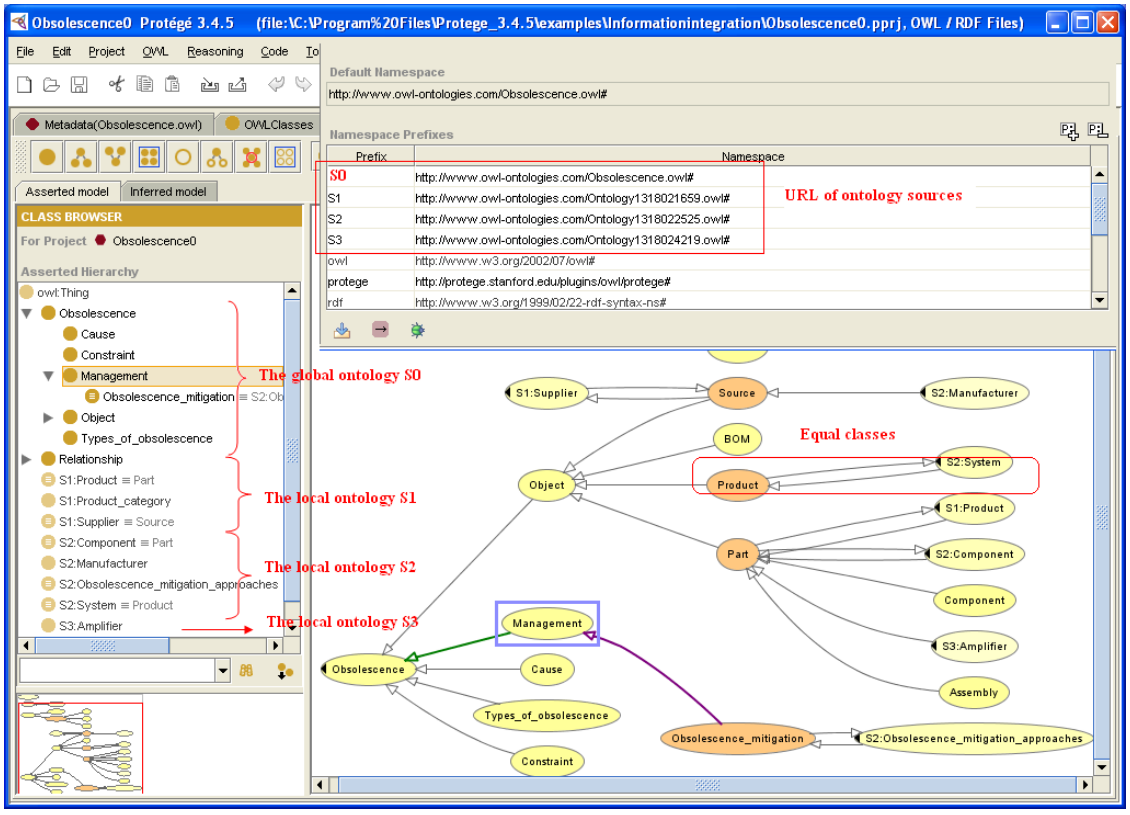


Figure 6-6: Three local ontologies imported for integrating with global ontology

Three types of relationships between classes in the global ontology and local ontologies can be identified using SWRL rules and SQWRL queries according to the semantic relationship values δ, and are assigned with new values of Δ:

- Equal ($0.8 < \delta \leq 1$) and $\Delta_{=} = 1$

Semantic_relationship(? $x1$) ∧ has_value(? $x1$, ? $y1$) ∧ swrlb:greaterThan(? $y1$ 0.8)∧swrlb:lessThanOrEqual(?$y1$, 1)⟶Equal(?$x1$)

Equal(?$x1$)⟶sqwrl: select(?$x1$)

Equal(?$x1$)⟶has_value(?$x1$,1.0)

- Include ($0.5 < \delta \leq 0.8$) and $\Delta_{\supset} = 0.8$

Semantic_relationship(? $x2$) ∧ has_value(? $x2$, ? $y2$) ∧ swrlb:greaterThan(? $y2$, 0.5)∧swrlb:lessThanOrEqual(?$y2$, 0.8)⟶Include(?$x2$)

Include(?$x2$)⟶sqwrl: select(?$x2$)

Include(?$x2$)⟶has_value(?$x2$,0.8)

- Relate ($0.1 < \delta \leq 0.5$) and $\Delta_{\sim} = 0.5$

Semantic_relationship(? $x3$ ∧ has_value(? $x3$, ? $y3$) ∧ swrlb:greaterThan(? $y3$, 0.1)∧swrlb:lessThanOrEqual(?$y3$, 0.5)⟶Relate(?$x3$)

Relate(?$x3$)⟶sqwrl: select(?$x3$)

Relate(?$x3$)⟶has_value(?$x3$,0.5)

Figure 6-7 shows SWRL rules and SQWRL queries for identifying three types of semantic relationships between classes in Protégé, and results of equal semantic relationship between classes found: sr_Part2S1_Product, sr_Product2S2_System, sr_Part2S2_Component, sr_Source 2S1_Supplier, sr_Obsolescence_mitigation2S2_Obsolescence_mitigation_approaches.

sr_Part2S1_Product represents the semantic relationship between the class Part in the global ontology and the class Product in the local ontology S_1, where 'sr' represents semantic relationship, and '2' links two classes together. Running the SWRLJessBridge, the OWL ontologies and SWRL rules are input to the Jess rule engine for execution, and the output results are new values assigned to semantic relationships between classes. For example, in Figure 6-8, sr_Part2S1_Product has value 0.91, so it belongs to the type of equal semantic relationship, and a new value of 1.0 is assigned.

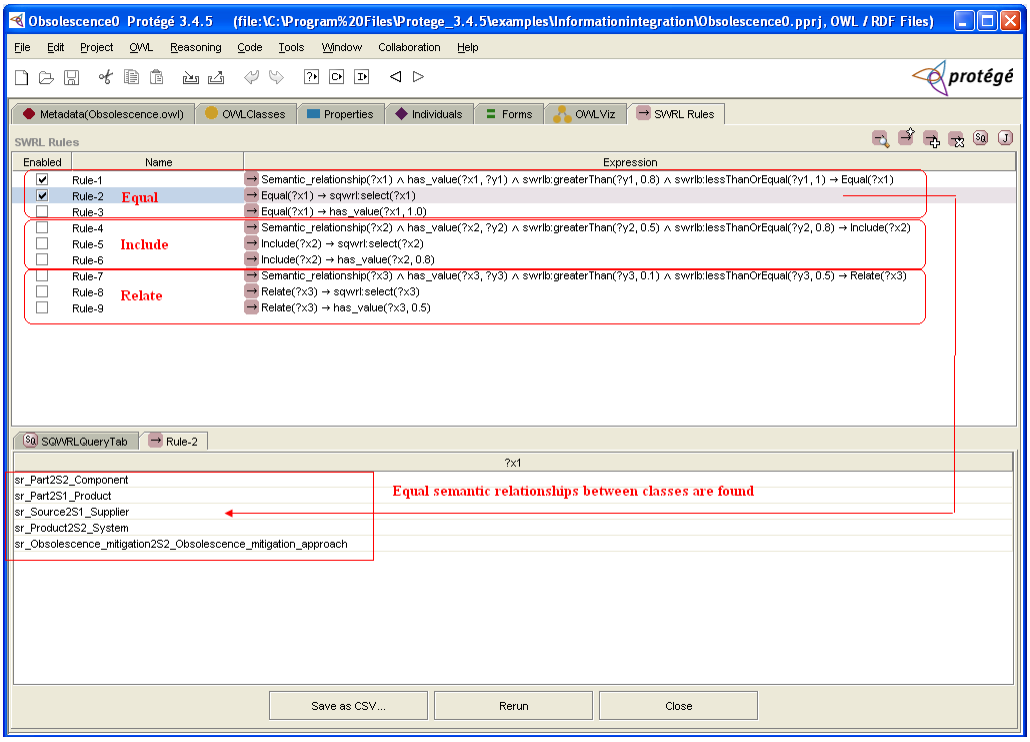


Figure 6-7: SWRL and SQWRL for identifying relationships between classes

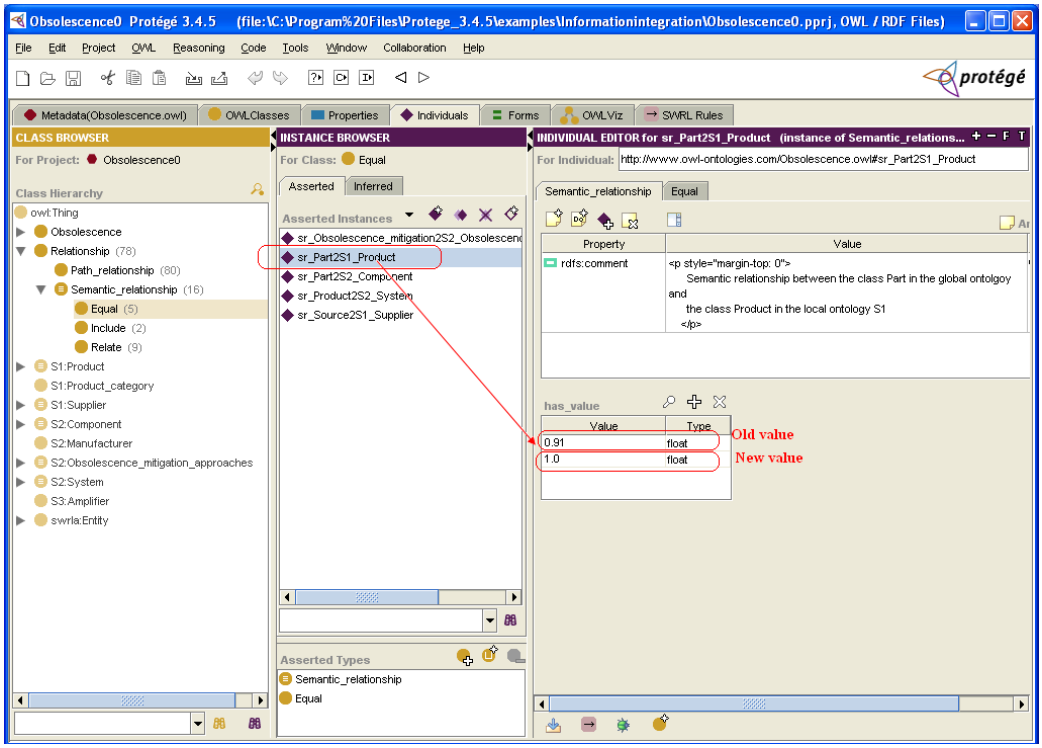


Figure 6-8: Assign new values to types of semantic relationships

If there is a path existing between two classes in the relationship diagram (Figure 4-9), the path relationship between these two classes can be calculated by multiplying the semantic relationships of classes existing in the path together. The SWRL rules for calculating the path relationships are as follows:

For example, there are two paths existing between the class Product in the global ontology S_0 and the class Product in the local ontologyS_1:

Path1: Product(S_0)—BOM(S_0)— Part(S_0)—Product(S_1)

σ_1 (Product(S_0),Product(S_1)) = Δ(Product(S_0), BOM(S_0))× Δ(BOM(S_0), Part(S_0)) × Δ(Part(S_0), Product(S_1))=0.5×0.5×1=0.25

*has_value(sr_Product2BOM, ? a1) ∧ has_value(sr_BOM2Part, ? b1) ∧ has_value(sr_Part2S1_Product, ? c1) ∧ swrlb: multiply(? m1 , ? a1 , ? b1 , ?c1)→has_value(**pr_Product2S1_Product_1**, ?m1)*

Path2: Product(S_0)—System(S_2)—Component(S_2)—Part(S_0)— Product(S_1)

σ_2 (Product(S_0),Product(S_1)) = Δ (Product(S_0), System(S_2)) × Δ (System(S_2), Component(S_2))× Δ(Component(S_2), Part(S_0)) × Δ(Part(S_0), Product(S_1))
=1×0.5×1×1=0.5

*has_value(sr_Product2S2_System, ? a2) ∧ has_value(sr_S2_System2S2_Component, ?b2)∧ has_value(sr_S2_Component2Part, ?c2)∧ has_value(sr_Part2S1_Product, ?d2)∧ swrlb: multiply(?m2, ?a2, ?b2, ?c2,?d2)→has_value(**pr_Product2S1_Product_2**, ?m2)*

The final relationship between two classes is the maximum among the path relationships. The SWRL rule for finding the final relationship is as follows:

Σ(Product(S_0),Product(S_1))=max(σ_1(Product(S_0),Product(S_1)), σ_2(Product(S_0),Product(S_1))=max(0.25,0.5)=0.5

*has_value(pr_Product2S1_Product_1, ? aa1) ∧ has_value(pr_Product2S1_Product_2, ? bb1) ∧ sqwrl: makebag(? mm1 , ? aa1) ∧ sqwrl: makebag(?mm1,?bb1)∧sqwrl:max(?max1,?mm1)→has_value(**r_Product2S1_Product**, ?mm1)*

pr_Product2S1_Product_1 represents the path relationship 1 between Product in the global ontology S_0 and Product in the local ontology S_1, where 'pr' represents path relationship, the first number '2' link two classes together, and the second number '1' represents the path relationship 1. Likewise, pr_Product2S1_Product_2 represents the path relationship 2 between Product in the global ontology S_0 and Product in the local ontology S_1. r_Product2S1_Product represents the final relationship between these two classes. The SWRL rules for calculating the path relationships between two classes and the results in Protégé are shown in Figure 6-9 and 6-10.

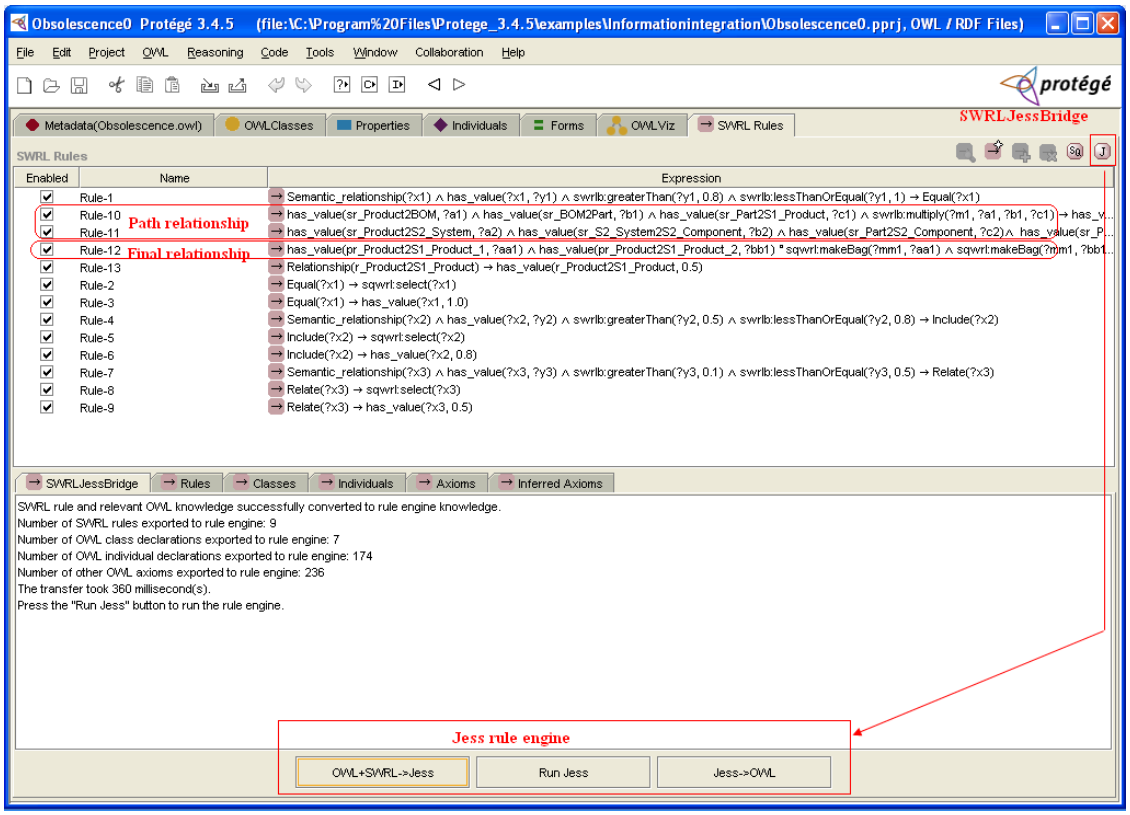


Figure 6-9: Calculate path relationship

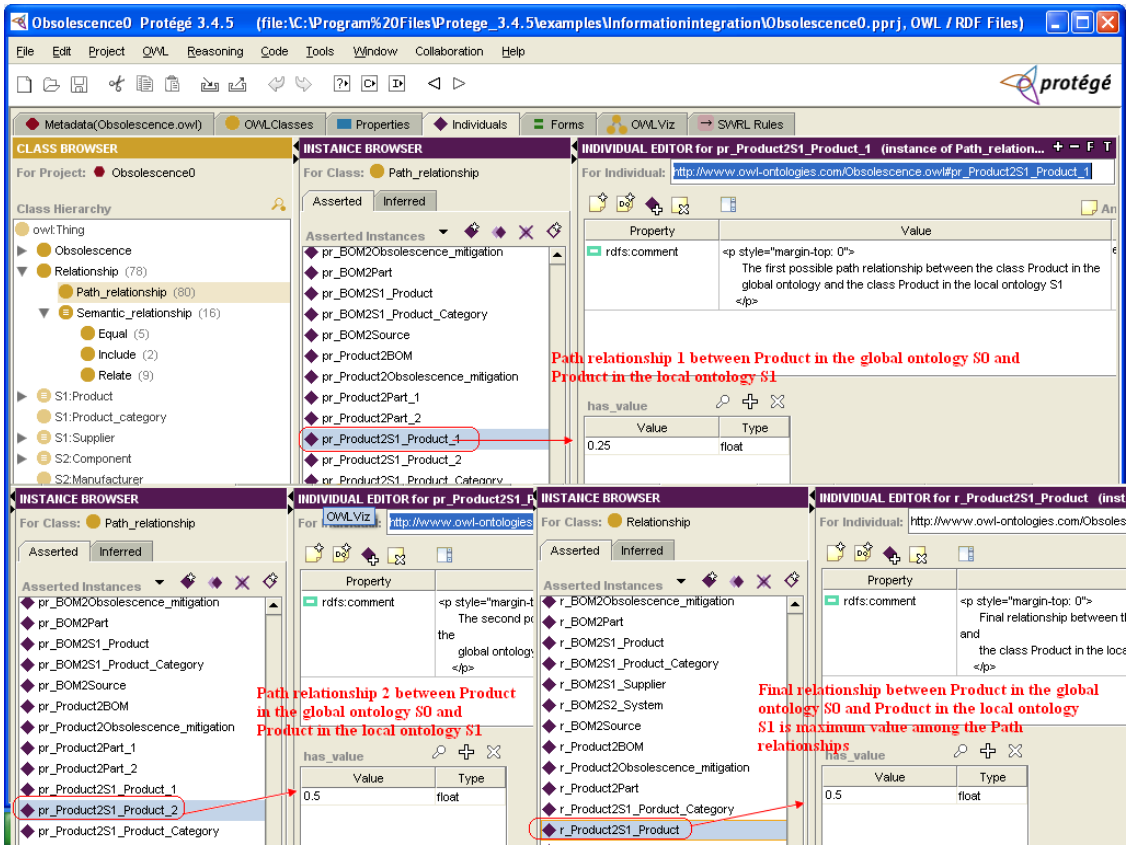


Figure 6-10: Results of path relationship calculation

The clustering method is applied based on the relationships between classes, and the group of classes representing the same domain concept can be identified. For example, the class Product from the global ontology S_0, the class Product from the local ontology S_1, the class Component from the local ontology S_2, and the class Amplifier from the local ontology S_3 form the group. Figure 6-11 shows the mapping of the slots for these classes. The slot product_id of the class Product (S_1), the slot component_name of the class Component (S_2), and the slot name of the class Amplifier (S_3) are all equivalent to the slot name of the class Part (S_0). Applying the ontology-based hybrid approach, the relationships between classes in the global ontology and the local ontologies in charge of heterogeneous data sources can be identified, and the local ontologies can be mapped to the global ontology.

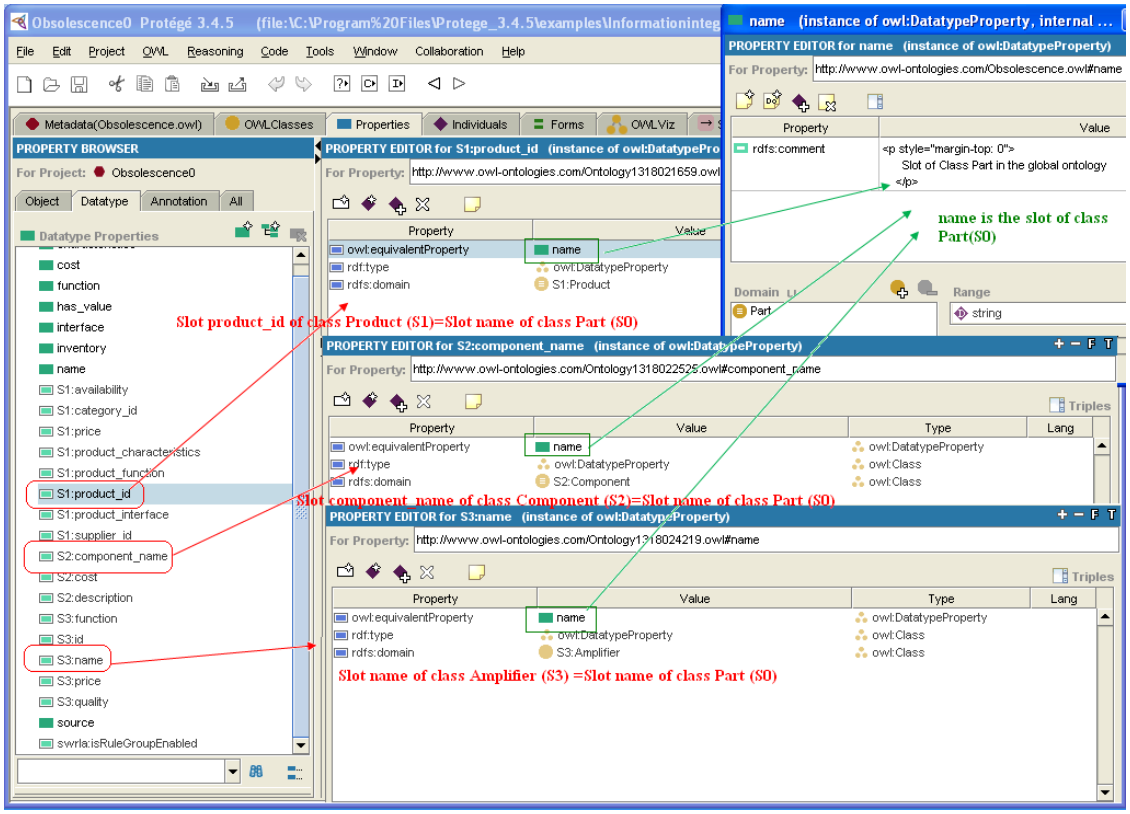


Figure 6-11: Mapping of slots for classes in the same group

6.2.3 Results of Decision Support

The obsolescence dates of parts in the system are important information inputs for proactive and strategic management of DMSMS obsolescence, which can be obtained from obsolescence forecasting. The obsolescence forecasting method based on product life cycle model has been described in Section 5.1. It can be represented with the following SWRL rule:

Part (?x)∧ has_zone_of_obsolescence (?x,?y) ∧Zone_of_obsolescence(?y) ∧ start (?y,?z) ∧ swrlb: lessThan (?z, p)⟶ Forecasted_ to_be_obsolete_ in_ p (?x)

Forecasted_ to_be_obsolete_ in_ p (?x)→sqwrl:select(?x)

Meaning, the part is forecasted to be obsolete in the specific future time p if the start time of the zone of obsolescence is less than p. p is a constant representing the time value. Forecasted_to_be_obsolete_in_p is a defined class. After executing the SQWRL query,

the part instances which are forecasted to be obsolete in p are automatically classified into this class. For example, 64Mbit monolithic flash memory is the part of the ECU and the time range of its zone of obsolescence is (2008.947, 2010.982). It is forecasted to be obsolete in 2010 because the start time of its zone of obsolescence 2008.947 is less than 2010. If p is 2010, then 64Mbit monolithic flash memory appears in the class Forecasted_to_be_obsolete_in_2010, as shown in Figure 6-12.

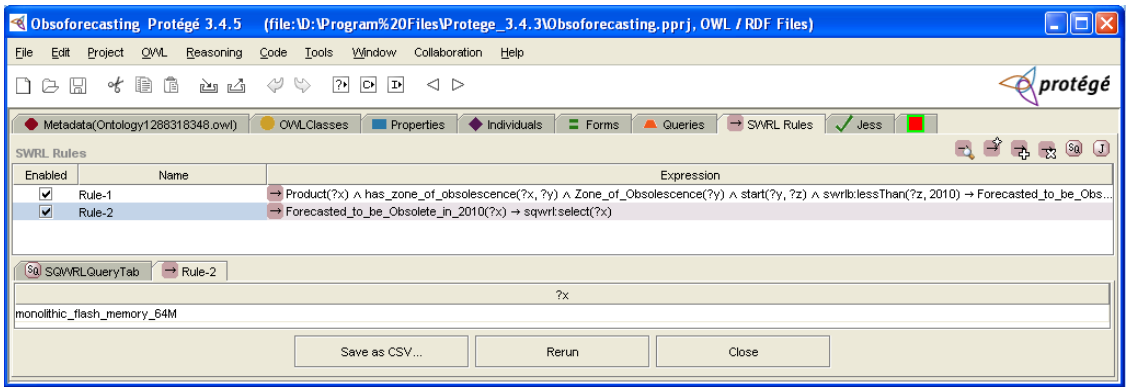


Figure 6-12: Example of SWRL rule and result for obsolescence forecasting

Design refresh planning is a way of strategically managing obsolescence with the input from obsolescence forecasting. It is used to schedule the time and content of design refreshes during the support life of the system to minimize overall obsolescence management cost. If parts are assumed to have specific obsolescence dates, deterministic models are considered. Otherwise, obsolescence dates are assumed to follow some probability distribution, and probabilistic models need to be built. Various design refresh planning models are developed for managing hardware obsolescence and software obsolescence. Results presented in Protégé are shown in Figure 6-13 and 6-14. Figure 6-13 shows the design fresh plan for the ECU. It is obtained by solving the deterministic model using the integer programming solver, so the total obsolescence management cost has a specified value 11968.88. The plan conveys the information about the time and content of doing design refreshes. The time choices of doing design refreshes are predetermined. They are 2003.00, 2005.00 and 2008.00 in the example of the ECU. The time 2003.00 and 2005.00 are selected for performing design refreshes. The content of design refresh indicates which parts including obsolete parts and non obsolete parts are

replaced in the specific design refresh. For example, Part_11, Part_12, Part_13, Part_15 and so on are replaced at the design refresh at 2003.00. Clicking a specific part (e.g., Part_11) , the interface showing the information about this part including name, source, cost, obsolescence date and so on pops up, which is similar to Figure 6-4. The ECU consists of hardware and software. Since there is software, short-term mitigation approaches need to be taken immediately if software components are obsolete during the time period between two design refreshes. Here Part_54 and Part_55 are replaced during the time period 2000.00-2003.00, Part_52 and Part_59 are replaced during the time period 2003.00-2008.00 using short-term mitigation approaches. If there is only hardware included, hardware can wait for replacement until next design refresh, so short-term mitigation approaches do not need to be considered.

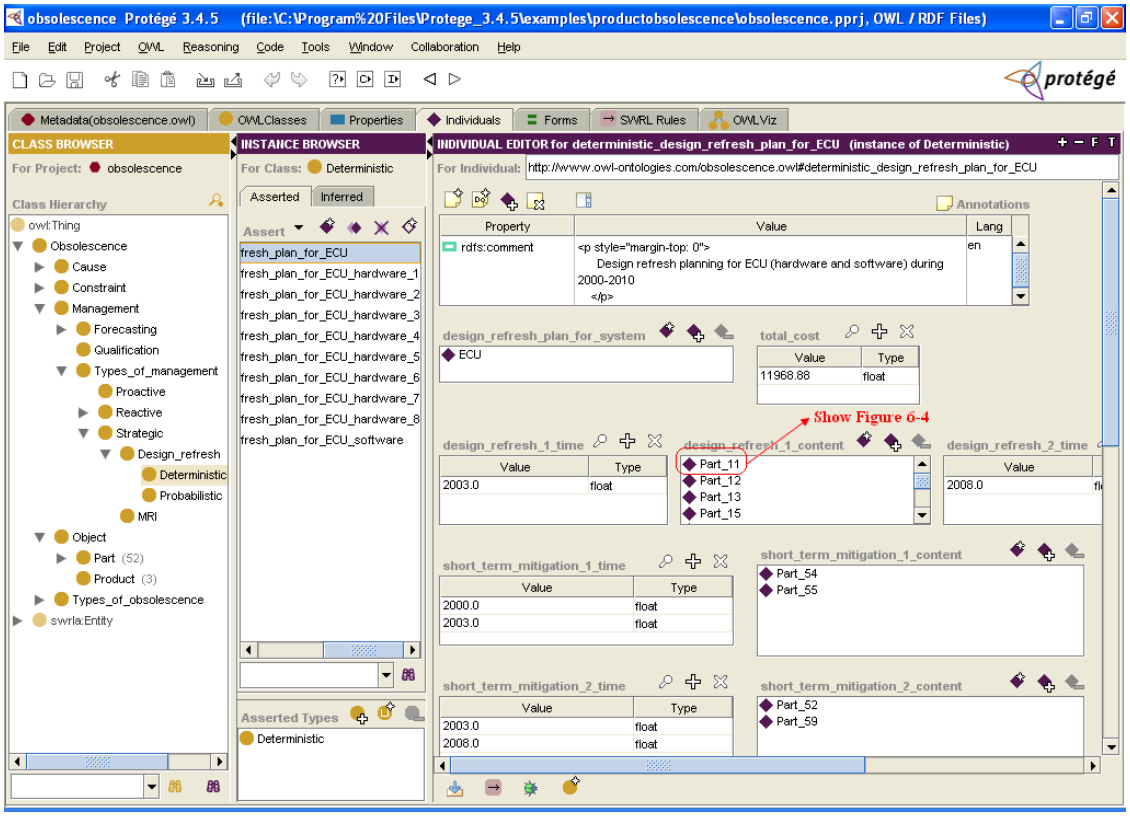


Figure 6-13: Deterministic design refresh plan for ECU

Figure 6-14 shows the design refresh plan for the ECU hardware, which is obtained from the probabilistic model. Because uncertainty of obsolescence dates is considered and

obsolescence dates are assumed to follow triangular probability distributions, the plan consists of different modes of obsolescence management solution with different total cost and probabilities of applying these modes: for example, 'deterministic_design_refresh_plan_for_ECU_hardware_1' with the total cost '9231.58' and the probability of applying this mode '0.2848', 'deterministic_design_refresh_plan_for_ECU_hardware_2' with the total cost '9231.695' and the probability of applying this mode '0.0397', and so on. Clicking one specific mode (e.g., deterministic_design_refresh_plan_for_ECU_hardware_1), the interface showing the information about this mode pops up, which is similar to Figure 6-13. Different from Figure 6-13, there is no information about short-term mitigation approaches in the interface because the mode is a design refresh plan for hardware obsolescence. With all modes considered, the cost distribution chart can be obtained, as shown in Figure 6-14. As the costs under different modes and their probabilities are known, the cost expectation for obsolescence management can be calculated. For example, 9270.029 is the cost calculated for the design refresh plan for ECU hardware.

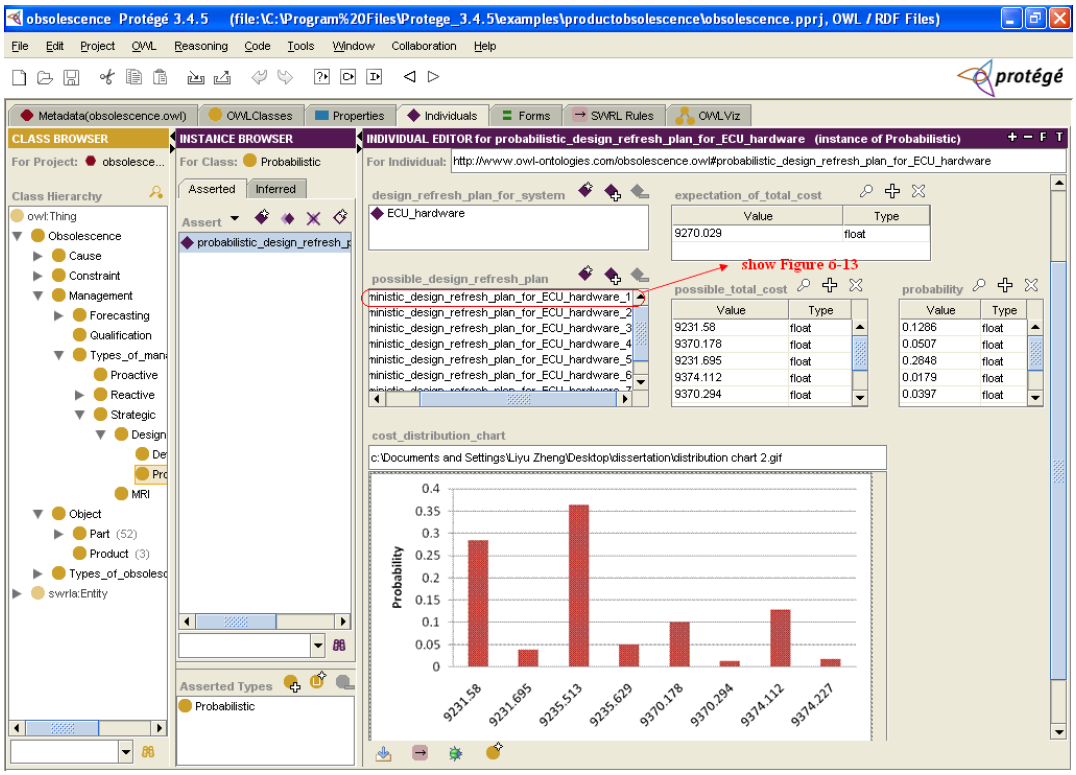


Figure 6-14: Probabilistic design refresh plan for ECU hardware

6.3 System Evaluation

Guida et al. (1993) indicated that the central task of knowledge-based system evaluation is to verify the actual system behavior aligns with the specification. This task is accomplished in two steps: first, evaluating KBS design versus KBS specification and later, evaluating actual KBS ontology versus KBS design (Figure 2-10). Based on Guida's approach, evaluation on the obsolescence management information system is divided into three parts: evaluation on obsolescence use case design, evaluation on obsolescence ontology definition, and evaluation on decision support model development for obsolescence management, as show in Figure 6-15. Assignments and methods used for evaluating each part are proposed as follows:

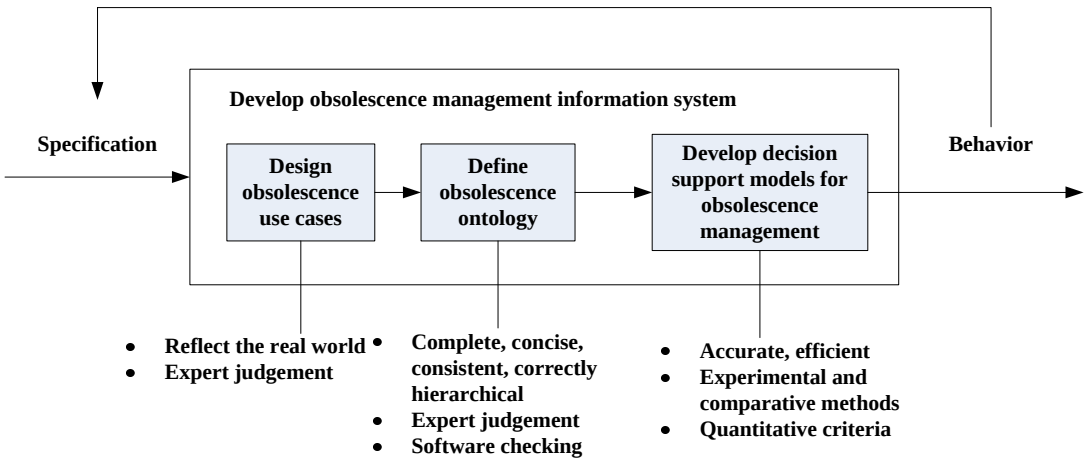


Figure 6-15: Evaluation on obsolescence management information system

- **Obsolescence use case design**

An ontology is a model of a real domain in the world and the concepts in the ontology must reflect this reality. The use case serves as a bridge between the real world and the ontology models. The objective of the evaluation on obsolescence use case design is to check whether the obsolescence use cases can reflect the real world, i.e., how product obsolescence affects daily life in production, maintenance, supply chain management, etc. The method of expert judgment can be used through a survey. Domain experts can give suggestions according to their work experience. Based on the feedback from domain

experts, the design of use cases can be improved. The update of use cases is also very important in the phase of ontology maintenance. If the ontology is outdated, errors will occur because the ontology model cannot represent the present reality.

- **Obsolescence ontology definition**

The objective of the evaluation of the obsolescence ontology is to check whether it is complete, concise, consistent, and correctly described in the hierarchy. Complete and concise checking can be taken by domain experts and ontology experts together to see if the ontology classes can completely support the use cases: No necessary knowledge is missing (complete), and no unnecessary knowledge exists (concise). To verify generality and completeness of the ontology, checking will consider not only the product obsolescence instance (ECU) but also that the skill obsolescence instance is used to support the developed obsolescence ontology.

Consistency and hierarchy checking can be completed using Protégé. Protégé can help to find logic errors in the ontology, reduce, and possibly eliminate syntax errors and logic errors. Figure 6-16 shows consistency and hierarchy checking for the developed obsolescence ontology in Protégé. Consistency checking can detect the error that disjoint classes have the same subclass. For example, the class Types of obsolescence, Cause, Object, Management, and Constraint in the obsolescence ontology are disjoint classes because they represent different concepts in the domain of obsolescence. There is no class which can be the subclass of one and another of these 5 classes at the same time. Hierarchy checking in Protégé-OWL is based on what is known as the open world assumption (OWA). OWA ensures that something cannot be assumed to not exist until it is explicitly stated that it does not exist. For example, Protégé may assign subclasses of Strategic (management) (e.g., Design refresh) to be subclasses of Proactive (management) if there is no restriction indicating that they can only be subclasses of Strategic (management), because if an obsolescence management approach is a strategic management one, it is at least a proactive one.

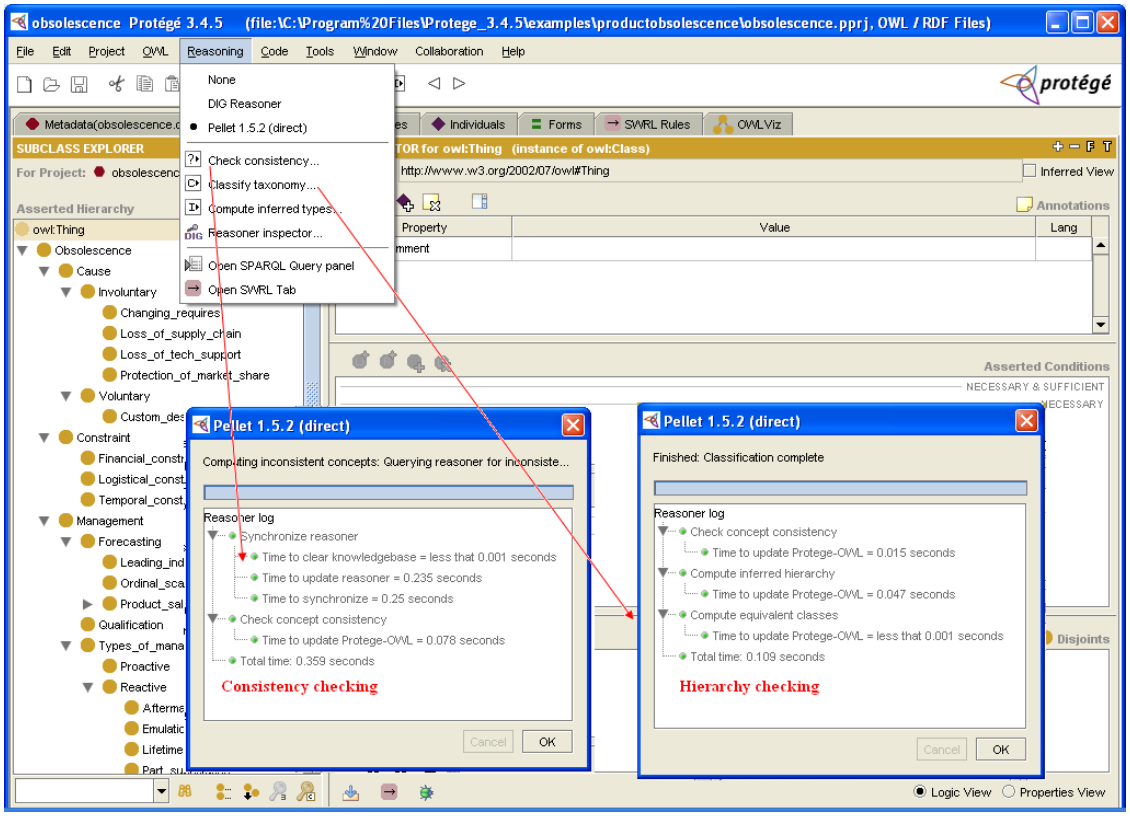


Figure 6-16: Consistency and hierarchy checking for obsolescence ontology in Protégé

- **Decision support model development for obsolescence management**

The objective of the evaluation of decision support models for obsolescence management is to check whether the outcomes are achieved for accuracy and efficiency. Important aspects aligned with the research purpose are to manage obsolescence more accurately and efficiently. Experimental methods can be used and some quantitative criteria are defined for evaluation. For example, an obsolescence forecasting method can be tested with a set of historical data. Mean squared error from forecasted outcomes and actual observation data can be used to evaluate accuracy of the obsolescence forecasting method. For evaluating efficiency of strategic obsolescence management, the solution to design refresh planning needs to be determined whether it is feasible first, and further whether it is optimal. The results can be compared with those of existing obsolescence management tools like the CALCE MOCA design fresh planning tool.

Evaluation on each part of the system development tries to ensure that the system behavior can satisfy the specification. Developing a knowledge-based information system with ontology is usually an iterative process. After an initial version of the ontology is defined, it can be evaluated and debugged by using it in applications or problem-solving methods or by discussing it with experts in the field. As a result, it will almost certainly need to be revised. This process of iterative design will likely continue through the entire life of the ontology.

Sandborn et al. (2007) have expanded a detail set of evaluation criteria for evaluating DMSMS tools, databases, and services from a taxonomy which was introduced in Section 2.6. These evaluation criteria range from general topics, aggregation/collaboration environments, part data management, part list monitoring, platform/system analysis and management, and strategic planning. They appear in the form of questions. An evaluation on our obsolescence management information system design by answering these questions is provided below:

Evaluation criteria Q&A

Q: What is the update frequency on the data?

A: UML can facilitate the update of ontology to adapt to changing data.

Q: What is the source of the data?

A: Heterogeneous data sources distributing in existing DMSMS tools and databases.

Q: Has the accuracy of the data been measured?

A: Data preprocessing can detect and eliminate data redundancy and data errors.

Q: What is the tool/database's taxonomy?

A: Ontology helps to determine the tool/database's taxonomy.

Q: Are data conflicts resolved? (I.e., when different sources of information disagree, is there a process for providing a status?)

A: Yes.

Q: How are data conflicts resolved? (What happens when multiple data sources disagree?)

A: Semantic Web technology is introduced for solving the problem of data conflicting. OWL is used to construct ontologies. The ontology-based hybrid approach is applied for integrating heterogeneous data sources.

Q: Breadth of coverage: Are military parts or COTS parts included? (If so what and how many)

A: Yes, obsolescence ontology has potential to integrate databases of military parts and COTS parts.

Q: Are Aftermarket sources considered?

A: Yes, Aftermarket sources as one of obsolescence mitigation approaches are considered.

Q: Are software drivers and/or firmware included?

A: Yes, software obsolescence is included and the design refresh planning model for managing software obsolescence is developed.

Q: What is the form of the forecast (date, risk color, life-code, etc.)?

A: Forecasting is to determine the zone of obsolescence (dates when a product has high probability of being obsolete) for the product based on the product life cycle.

Q: What is the source of the data (or how is the forecast determined)?

A: Annual sales data of products.

Q: Are uncertainties on forecasts estimated?

A: The zone of obsolescence (a period of time) is used instead of a specific obsolescence date considering uncertainties on forecasts.

Q: Has accuracy of the data been measured?

A: Mean squared error (error between forecasted value and observed value) is used.

Q: What type of planning is performed?

A: Design refresh planning.

Q: What type of analysis is performed within the planning (I.e., is optimization supported? If optimization is supported, what is optimized?)

A: Optimization is supported with MILP models. The overall obsolescence management cost during the life cycle of the product is minimized.

Q: What is the scope of the planning/analysis (I.e., is analysis supported through a range of physical hierarchies? Are hardware and software considered in the analysis?)

A: Analysis can support system, subsystem, component, etc. Hardware and software are considered in the analysis.

Q: Is uncertainty analysis performed?

A: With the assumption that the obsolescence dates of parts in the system follow certain probability distribution, the probabilistic design refresh planning model with expected total cost is developed.

Q: Is there a linkage to life cycle cost?

A: Yes, the product life cycle cost model is defined.

Q: Is there a linkage to technology roadmapping?

A: Yes, the design refresh plan is made based on the production plan, which reflects technology roadmapping.

Q: Is there a linkage to logistics planning and control?

A: No, not yet.

…

Chapter 7 Contributions and Recommendations

In this dissertation, methods for knowledge representation and decision support for managing product obsolescence have been proposed, realized and refined. Section 7.1 reviews the research objectives and the proposed research approach, concluding with answers to the research questions posed in Chapter 1. The resulting contributions of the research are summarized in Section 7.2. Limitations of the research and opportunities for future research are given in Section 7.3.

7.1 Research Summary

As stated in Chapter 1, the purpose of the research is to establish an ontology-based knowledge representation scheme for information sharing, reuse, and collaboration on obsolescence issues across different organizations, and develop decision models to support proactive and strategic management for overall cost savings in managing obsolescence. Diminishing Manufacturing Sources and Material Shortages (DMSMS) obsolescence is selected as the context for this study, which is defined as the loss or impending loss of original manufacturers of items or suppliers of items or raw materials for long field-life sustainment-dominated systems.

For this purpose, the obsolescence ontology has been developed in a systematic way with the use of UML diagrams for capturing the domain knowledge in obsolescence. The obsolescence hierarchy, slots, relationships are defined for date management and information sharing in the domain of obsolescence. The ontology-based hybrid approach has been proposed for integrating heterogeneous data sources in existing obsolescence management tools. Applying this approach, local ontologies are defined in charge of heterogeneous data sources, and the global ontology is defined for representing the built obsolescence knowledge representation scheme. The procedure of identifying the relationships between the global ontology and local ontologies and mapping the classes in local ontologies to the global ontology has been presented. Based on the obsolescence knowledge representation scheme, the obsolescence forecasting method based on the product life cycle has been represented with SWRL rules for proactive management.

Various design refresh planning models have been developed with mathematical programming technique for strategic obsolescence management. These models are able to determine optimal design refresh plans to minimize the overall obsolescence management costs with and without the consideration of the uncertainty of the obsolescence dates. They can be applied for hardware and software obsolescence. Results from knowledge representation, information integration, and decision support can be utilized for the information system development. The approach to the obsolescence management information system development has been investigated. The system consists of obsolescence data management, decision models for obsolescence management, and the user interface. The system evaluation focuses on verifying if the actual behavior of the system satisfies the specification. Related assignments and methods have been defined.

In summary, the resulting research method yields a systematic means to present a common understanding of domain knowledge, integrate multiple heterogeneous data resources, and support decision making with the consideration of certain constraints and economical objectives.

The use of this method to support knowledge representation and decision making in product obsolescence is exploited in the context of four broad categories of research questions.

1. Questions on Knowledge Representation

Q1. How should an obsolescence ontology be developed? What tool and methodology are used to support the ontology development process and help in the evolution, updating and maintenance of the ontology?

Q2. What concepts and properties related to obsolescence need to be structured in the ontologies? Is the developed obsolescence ontology generalizable, i.e., suitable to a wide range of application?

The proposed process of the obsolescence ontology development includes (1) determine domain and scope with UML, (2) define classes and class hierarchy using system design

approaches with consideration of reusing existing ontologies, (3) Define slots of classes and describe allowed values for slots, (4) Establish instances by filling in the values for slots, (5) Evaluate and revise the developed ontologies. The obsolescence ontology development process is introduced (Section 3.2 to Section 3.6). Besides product obsolescence, the process can also be used for the ontology development in the domain of other problems.

The UML tool is utilized in the obsolescence knowledge discovery and capture (introduced in Section 3.2). The UML tool for obsolescence ontology development starts from design of use cases for obsolescence management, and then classes supporting use cases can be created. Use Case Diagrams and Class Diagrams are effective mechanisms for capturing the functional requirements of the system and discovering knowledge in the domain. They also support design, coding, testing, and validation activities during the system development. Hence, utilizing the UML tool can not only capture and visualize the design of the ontologies, but also help in the evolution, updating and maintenance of the ontologies.

A systematic methodology of combining the top-down and bottom-up approaches is used for developing the obsolescence class hierarchy (introduced in Section 3.3). With the top-down approach, the top-level classes Type, Cause, Object, Management, Constraint, which are subclasses of the class Obsolescence, are generated by asking questions (e.g., "What objects easily incur obsolescence?", "What is the cause of obsolescence?", "How to manage obsolescence?", etc). In the opposite direction, there has already been a pool of classes from the UML diagrams. These classes can be assigned to be subclasses of the top-level classes. For example, the classes Product and Part are subclasses of the class Object, and the classes Reactive, Proactive, Strategic are subclasses of the class Management. This is a bottom-up process. Classes developed in the opposite directions can finally meet together by using the blended method of combining the top-down and bottom-up approaches together.

Classes representing obsolescence concepts are managed in a hierarchical manner. The taxonomy of obsolescence is Types of obsolescence, Cause, Object, Management, and

Constraint. Take Cause for example, it has subclasses Involuntary cause and Voluntary cause. Involuntary cause includes Loss of supply chain, Changing requires, Loss of tech support, and Protection of market share, and Voluntary cause is Customer desire. The class hierarchy for product obsolescence is shown in Figure 3-5. The properties of concepts are represented with the slots of classes. For example, important slots of the class Part include source, inventory, characteristics, interface, and function. The meanings of these slots are introduced in Section 3.4. The important slots of classes related to product obsolescence are listed in Table 3-1. The developed obsolescence ontology can not only support product obsolescence, but also other obsolescence. Its generality is demonstrated in Section 3.6, where it is used for representing the skill obsolescence problem.

2. Questions on Information Integration

Q3. What technology can be applied for solving semantic heterogeneity for web-based implementation?

Q4. What approach can be used for integrating information from distributed heterogeneous data sources? How can this approach be applied?

Semantic Web is currently the most effective way of solving semantic heterogeneity in the web information. OWL (Web Ontology Language) as a Semantic Web language is used in the obsolescence ontology development for the purpose of web information integration. The obsolescence ontology built with OWL is introduced in Section 4.2.

The ontology-based hybrid approach is used for integrating information from existing heterogeneous data sources. There are local ontologies and the global ontology in the hybrid approach. Local ontologies describe heterogeneous data sources, and data from a relation database, object-oriented database and a file are used as examples of data sources. The global ontology has been developed from knowledge representation and is used to be in charge of local ontologies. The procedure of applying the hybrid approach for information integration is included: (1) Build OWL ontologies, (2) Quantify semantic relationship, (3) Discretize semantic relationships, (4) Create relationship diagram, (5)

Calculate path relationships, (6) Identify groups of classes, (7) Create mappings between classes. The procedure is described in Section 4.2 through Section 4.8. As a result of applying the approach, the relationships between local ontologies and the global ontology can be identified, and the classes in local ontologies can be mapped to the classes in the global ontology.

3. Questions on Decision Support

Q5. Considering that forecasting results are the important inputs for proactive and strategic obsolescence management, what obsolescence forecasting method can be developed based on the ontological knowledge representation scheme?

Q6. Considering that strategic obsolescence management is a promising way of significantly reducing overall cost related to obsolescence in the long term, what method is used to realize strategic obsolescence management? How can this method be applied?

Forecasting results are the important inputs for proactive and strategic obsolescence management. The method for obsolescence forecasting is developed based on the product life cycle. In this method, product life cycle stages and the zone of obsolescence are defined for the products with Gaussian distribution life cycle curves and represented by using ontology. The product life cycle curve is obtained by fitting the historical sales data with the Gaussian distribution. And the obsolescence date can be forecasted. The obsolescence forecasting method based on the product life cycle is described in Section 5.1.

Design refresh planning is one of effective ways of realizing strategic obsolescence management. Design refresh planning is to determine (1) when to proceed with design refresh and (2) What obsolete and non-obsolete parts should be replaced at a specific design refresh to minimize the overall life cycle cost of the system under the influence of obsolescence. If specific obsolescence dates can be obtained, models are deterministic and can be solved using the mixed integer linear programming technique. Otherwise, obsolescence dates are assumed to follow certain probability distribution. Different modes of obsolescence management solutions with optimal costs and their probabilities

need to be counted. The final optimal obsolescence management cost is an expectation value. Development of design refresh planning models is described in Section 5.2.1 and Section 5.2.2. Developed design refresh planning models can solve various types of problems including hardware obsolescence versus software obsolescence, deterministic model versus probabilistic model, and one replacement versus two replacements. Examples about design refresh planning models managing obsolescence of an electronic engine control unit (ECU) are introduced in Section 5.2.3.

4. Questions on System Development and Evaluation

Q7. How can the obsolescence management information system be developed and evaluated?

Q8. Which ontology editor (such as Protégé, OilEd, OntoBuilder, etc) is best suited and how is the obsolescence ontology built in the chosen ontology editor?

Preliminary work is discussed on the obsolescence management information system development and evaluation. Three main components of the obsolescence management information system are obsolescence knowledge base, decision support tools for obsolescence management, and the user interface. Results from Chapter 3 to Chapter 5 can be used in the system development, which is discussed in Section 6.1. The obsolescence management information system evaluation is to verify if the actual system behavior satisfies the specification. The evaluation assignments and methods are defined for each phase during the system development. The system evaluation is discussed in Section 6.3.

Protégé is used for the ontology development. Screenshots of the results in Protégé are presented in Section 6.2. They are the obsolescence class hierarchy and the instances of the classes related to knowledge representation, SWRL rules and SQWRL queries for quantifying and identifying relationships between classes in the global ontology and local ontologies using hybrid approaches for information integration, and SWRL rules and SQWRL queries for obsolescence forecasting and data inputs and outputs for design refresh planning models related to decision support.

7.2 Contributions of Research

The research has contributed to the knowledge representation and decision support in product obsolescence, especially in DMSMS obsolescence through DMSMS knowledge represented with ontology and design refresh planning models developed. The research contributions can be illustrated with outcomes of applying the proposed research methods to examples see Chapter 3 to Chapter 6.

A primary contribution of this research is that it offers a formal specification of knowledge in obsolescence, which can facilitate the sharing of knowledge and collaboration on obsolescence issues between the distributed organizations. The obsolescence ontology is to provide a unified scheme for the obsolescence domain to support obsolescence resolution activities as well as obsolescence part data management. It can be generalized to support other types of obsolescence (e.g., skill obsolescence). The procedure for ontology development facilitates the evolution, updating and maintenance of the developed ontology adapting to the rapidly changing requirements.

Secondly, the proposed ontology-based hybrid approach contributes to information integration for heterogeneous data sources. After the relationships among data are identified, data inconsistency and data conflict can be found and eliminated. The various existing obsolescence data sources can be incorporated into the general obsolescence knowledge representation scheme. The information in the multiple existing obsolescence management tools can be utilized. A broader picture of obsolescence management can be provided.

Thirdly, decision models have been developed to proactively and strategically manage DMSMS obsolescence. Previous methods mainly focused on reactive management which means obsolescence is solved after it has already happened. The predicted obsolescence dates of parts from the obsolescence forecasting method is the most important information for changing obsolescence management from reactive to proactive and strategic. With the obsolescence forecasting information input, design refresh planning can realize strategic obsolescence management. Mathematical programming is introduced for model development to support obsolescence management. Statistical technique is

applied to deal with the uncertainty of the models. The developed design refresh planning models help organizations to systematically execute design refresh activities which replace obsolete or predicted obsolete parts by combining multiple resolution approaches during the life cycle of the system to minimize the overall obsolescence management cost.

Moreover, the obsolescence ontology is built with OWL for web information integration. The obsolescence forecasting method is represented with SWRL rules for the purpose of formalization. The obsolescence management information system including data management and decision models has begun to take shape.

7.3 Limitations of Research and Opportunities for Future Research

This topic has been studied in the context of DMSMS obsolescence and lays the foundation for a generalized and comprehensive framework for a broadly defined system to support obsolescence management. In the future, the ultimate success of the obsolescence management approach will rely on developing a more integrated web-centric system that allows information sharing, reuse, and collaboration on obsolescence issues across different organizations.

The current obsolescence knowledge representation scheme includes the limited concepts which are in the general level. If it is applied for a real industrial obsolescence case, more specific concepts need to be added. Hence the obsolescence knowledge representation scheme needs to be enriched to integrate all obsolescence resolution activities to support various types of obsolescence. But there is a trade-off between generality and detail. Sometimes if the scheme fits one case too well, it cannot work well for other cases. For this problem, a large amount of use cases in the obsolescence domain need to be developed. Then specific concepts can be grouped to be associated with different use cases. As the scale of the scheme is enlarged, more problems such as concept conflict and overlap will appear and need to be solved.

The current decision models need to be improved for obsolescence management. The presented obsolescence forecasting method is only suitable for obsolescence forecasting

for some products whose life cycles can be obtained and follow the Gaussian distribution. This obsolescence forecasting method is used to show that it can be formalized with ontology rules to work with the obsolescence knowledge representation scheme. More general and effective obsolescence forecasting methods need to be developed and included. The developed design fresh planning models can only work for hardware and software obsolescence, although hardware and software obsolescence is the most important representative of DMSMS obsolescence. The models are also not complete for the application. For example, for part replacements at design refreshes, there are multiple choices of the replacement parts with different sources, costs and life cycle cycles. The choice will affect the final obsolescence management solution and cost. This is not considered in the current models.

For information management as a decision support system, much work about searching information, locating information and utilizing information for decision support currently is finished manually, not automatically. Ports that enable the obsolescence knowledge representation scheme to be in charge of multiple data sources, and connect the data management tools and decision support tools together need to be studied and developed. It is essential to develop those ports since more decision support tools such as statistical tools and simulation tools will be introduced in the future for enriching the function of decision support to satisfy customers' requirements. The ports for those decision support tools are also required.

Finally, the system development and evaluation of this research is at the methodology level. In the future, the real system needs to be developed to support decision making for effective and efficient obsolescence management with the most cost saving.

Appendices

Appendix A: Background of ECU

An example for demonstration of the design refresh planning models for obsolescence management is an electronic engine control unit (ECU) produced by a major automation and control company. The engine controlled by this unit is designed for regional and business class aircrafts, and the unit is mounted on the engine, as shown in Figure 8-1.

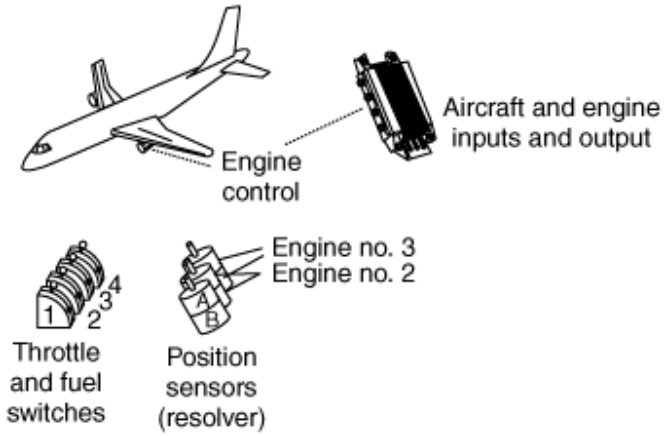


Figure A-1: Usage of ECU

(http://www.answers.com/topic/european-economic-community)

The ECU is a closed loop device that controls the engine to produce a determined amount of thrust by monitoring engine parameters, including: acceleration, deceleration, engine speed and temperature, oil pressure and temperature, ambient temperature and pressure, and fuel inlet temperature and pressure. The ECU uses these parameters to actuate compressor guide vanes, bleed valves, and fuel flow to produce the desired thrust (Cunningham, 2001).

The ECU consists of three circuit card assemblies (CCA) populated with commercial surface mount components and housed in an aluminum (6061-T6) chassis. The three CCAs are identified as the I/O, CPU, and EMI. The I/O and CPU CCAs are mounted flush to the chassis, whereas the EMI CCA is mounted to the chassis by three connectors. An assembly schematic is shown in Figure 8-2.

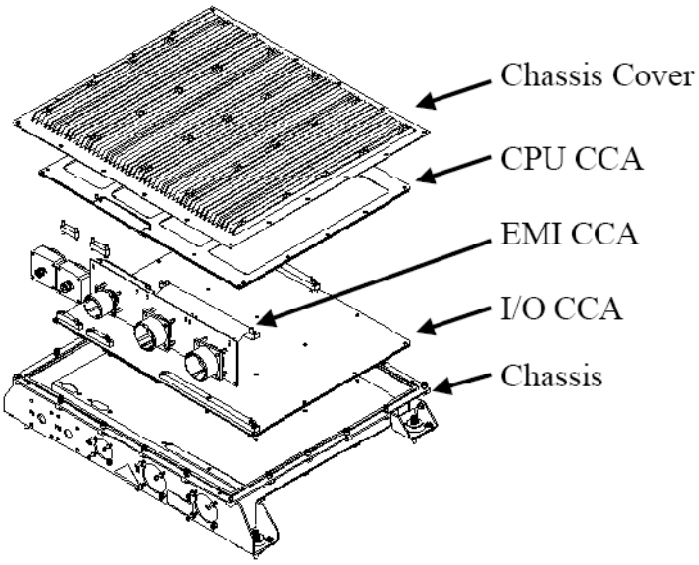


Figure A-2: Assembly schematic of ECU (Cunningham, 2002)

The I/O and CPU CCAs are composed of 10-layer high temperature FR-4 printed wiring boards (PWBs) with components populated on a single side. An aluminum (6061-T6) heat sink is bonded to the unpopulated side of the PWBs. The I/O and CPU CCAs are populated by 2039 and 1602 components respectively, with the majority of the passive components on the I/O CCA and the microprocessors and memory devices on the CPU CCA. The EMI CCA is a 10 layer high-temperature FR-4 PWB with 308 components populated on either side and does not incorporate a heat sink. Components of interest within the assembly include microprocessors, memory devices, and many small ceramic passive devices. Some critical components for the operation of the ECU are a 144 gull-wing leaded plastic quad flat pack (PQFP) package, two 132 gull-wing leaded PQFP micro-controllers, and a series of 44 gull-wing leaded and 32 J-leaded memory devices. All of which are relatively large components mounted on the CPU CCA. Passive devices make up the majority of the total number of components and are placed throughout the three CCAs (Cunningham, 2002).

The ECU also has software operating those three CCAs like operating system, control application software, data management software, and so on. The operating system software is responsible for the direct control and management of hardware and basic

system operations, as well as running application software. The control application software is to realize the control function of the ECU. The data management software is used in charge of data storage and data exchange in the execution of the ECU. Other software components have their own functions. The selected components including hardware and software from the ECU system used for demonstration of the design refresh planning models for obsolescence management are listed in Table 8-1.

Table A-1: Components selected from ECU for demonstration of design refresh planning models for obsolescence management

ID	Name	Quantity	Cost($)	Obs date	Obs dist	Dist low	Dist high	Life time
part_1	Assorted	1	564.107	2100	None	0	0	100
part_4	MICROCIRCUIT	3	2.9	2004.4	Triangular	2003.52	2004.4	10
part_5	MICROCIRCUIT	1	27.98	2004.05	Triangular	2003.24	2004.05	10
part_6	MICROCIRCUIT	27	0.39	2004.05	Triangular	2003.24	2004.05	10
part_7	INTEGRATED CKT	6	1.03	2004.05	Triangular	2003.24	2004.05	10
part_8	MICROCIRCUIT	1	5.78	2004.8	Triangular	2003.84	2004.8	10
part_9	MICROCIRCUIT	11	1.74	2004.325	Triangular	2003.46	2004.325	10
part_10	MICROCIRCUIT	13	2.14	2003.325	Triangular	2002.66	2003.325	10
part_11	MICROCIRCUIT	6	5.1	2006.4	Triangular	2005.12	2006.4	10
part_12	MICROCIRCUIT	2	5.45	2004.8	Triangular	2003.84	2004.8	10
part_13	MICROCIRCUIT	1	2.45	2004.125	Triangular	2003.3	2004.125	10
part_14	MICROCIRCUIT	1	6.78	2004.8	Triangular	2003.84	2004.8	10
part_15	MICROCIRCUIT	1	14.85	2001.75	Triangular	2001.2	2001.75	10
part_16	MICROCIRCUIT	1	0.35	2003.75	Triangular	2003	2003.75	10
part_17	MICROCIRCUIT	2	0.23	2002.75	Triangular	2002.2	2002.75	10
part_18	MICROCIRCUIT	8	0.18	2003.75	Triangular	2003	2003.75	10
part_19	MICROCIRCUIT	2	48.4	2004.8	Triangular	2003.84	2004.8	10
part_20	MICROCIRCUIT	6	0.36	2002.75	Triangular	2002.2	2002.75	10
part_21	MICROCIRCUIT	1	0.75	2002.75	Triangular	2002.2	2002.75	10
part_22	MICROCIRCUIT	6	13.19	2004.4	Triangular	2003.52	2004.4	10
part_23	MICROCIRCUIT	1	16.95	2004.4	Triangular	2003.52	2004.4	10
part_24	MICROCIRCUIT	3	6.2	2004.4	Triangular	2003.52	2004.4	10
part_25	MICROCIRCUIT	4	0.55	2003.725	Triangular	2003.1	2003.725	10
part_26	MICROCIRCUIT	7	0.452	2003.775	Triangular	2003.02	2003.775	10
part_27	DIODE	2	0.05	2009.65	Triangular	2009.15	2011.58	20
part_28	DIODE	1	0.05	2016.95	Triangular	2016.95	2020.34	20
part_29	SEMICONDUCTOR	21	0.3	2012.5	Triangular	2012.5	2015	20
part_30	SEMICONDUCTOR	6	0.32	2012.5	Triangular	2012.5	2015	20
part_31	SEMICONDUCTOR	37	0.074	2011.65	Triangular	2011.65	2013.98	20
part_32	SEMICONDUCTOR	3	0.03	2009.15	Triangular	2009.15	2010	20

part_33	SEMICONDUCTOR	26	0.04	2009.9	Triangular	2008.92	2009.9	20
part_34	SEMICONDUCTOR	9	0.16	2009.9	Triangular	2007.02	2011	20
part_35	SEMICONDUCTOR	3	0.23	2010	Triangular	2008	2012	20
part_36	SEMICONDUCTOR	3	0.208	2010	Triangular	2008	2012	20
part_37	SEMICONDUCTOR	1	2.15	2011.4	Triangular	2009.12	2013.68	20
part_38	SEMICONDUCTOR	1	1.076	2013.35	Triangular	2010.68	2016.02	20
part_39	MICROCIRCUIT	1	7.5	2007.65	Triangular	2006.12	2007.65	10
part_40	MICROCIRCUIT	2	0.156	2003.75	Triangular	2003	2003.75	10
part_41	MICROCIRCUIT	1	0.194	2004.5	Triangular	2003.6	2004.5	10
part_42	MICROCIRCUIT	2	34.95	2004.05	Triangular	2003.24	2004.05	10
part_43	MICROCIRCUIT	2	9.22	2010	Triangular	2008	2010	10
part_44	SEMICONDUCTOR	4	0.32	2016.65	Triangular	2013.32	2019.98	20
part_51	CONTROL APPLICATION	1	10.9	2006.8	Triangular	2006.84	2006.8	10
part_52	DATA MANAGEMENT	1	16.95	2004.4	Triangular	2003.52	2004.4	10
part_53	CONTROL APPLICATION	1	1.44	2003.75	Triangular	2003	2003.75	10
part_54	DATA MAMAGEMENT	1	2.16	2002.75	Triangular	2002.2	2002.75	10
part_55	CONTROL APPLICATION	1	0.75	2002.75	Triangular	2002.2	2002.75	10
part_56	OPERATING SYSTEM	1	79.14	2004.4	Triangular	2003.52	2004.4	10
part_57	CONTROL APPLICATION	1	16.95	2008.4	Triangular	2008.52	2008.4	10
part_58	CONTROL APPLICATION	1	18.6	2007.4	Triangular	2006.52	2007.4	10
part_59	DATA MANAGEMENT	1	3.164	2003.775	Triangular	2003.02	2003.775	10
part_60	DATA MANAGEMENT	1	10.53	2004.05	Triangular	2003.24	2004.05	10

Appendix B: Cost Modifiers of Components for Design Refresh Planning Models

Table B-1: Cost modifiers $\boldsymbol{M_{ij}^R}$ of hardware components

Modifier $\boldsymbol{M_{ij}^R}$	2003.00	2005.00	2008.00	**Modifier** $\boldsymbol{M_{ij}^R}$	2003.00	2005.00	2008.00
part_1	1	1	1	part_24	16.2825	18.31694	16.05453
part_4	17.57268	3.469684	16.91923	part_25	15.00241	16.2825	18.31694
part_5	21.28752	17.57268	3.469684	part_26	20.36794	15.00241	16.2825
part_6	20.4207	21.28752	17.57268	part_27	1.155663	20.36794	15.00241
part_7	100	20.4207	21.28752	part_28	1.474459	1.155663	20.36794
part_8	16.56302	100	20.4207	part_29	3.38894	1.474459	1.155663
part_9	24.38545	16.56302	100	part_30	3.060418	3.38894	1.474459
part_10	21.54499	24.38545	16.56302	part_31	3.214502	3.060418	3.38894
part_11	20.06087	21.54499	24.38545	part_32	2.809195	3.214502	3.060418
part_12	18.90471	20.06087	21.54499	part_33	2.16678	2.809195	3.214502
part_13	16.07375	18.90471	20.06087	part_34	2.731535	2.16678	2.809195
part_14	22.83995	16.07375	18.90471	part_35	4.937234	2.731535	2.16678
part_15	19.59641	22.83995	16.07375	part_36	2.182749	4.937234	2.731535
part_16	22.53688	19.59641	22.83995	part_37	3.170874	2.182749	4.937234
part_17	20.96095	22.53688	19.59641	part_38	3.094986	3.170874	2.182749
part_18	17.26866	20.96095	22.53688	part_39	21.57055	3.094986	3.170874
part_19	15.18758	17.26866	20.96095	part_40	20.44014	21.57055	3.094986
part_20	17.10369	15.18758	17.26866	part_41	23.27412	20.44014	21.57055
part_21	15.73953	17.10369	15.18758	part_42	15.81894	23.27412	20.44014
part_22	16.05453	15.73953	17.10369	part_43	16.91923	15.81894	23.27412
part_23	18.31694	16.05453	15.73953	part_44	3.469684	16.91923	15.81894

Table B-2: Cost modifiers $\boldsymbol{M_{ij}^R}$ s and $\boldsymbol{M_{ij}^M}$ s of software components

Modifier $\boldsymbol{M_{ij}}$	**Design refresh** ($\boldsymbol{M_{ij}^R}$)			**Short-term mitigation**($\boldsymbol{M_{ij}^M}$)			
	2003.00	2005.00	2008.00	(2000.00, 2003.00)	(2003.00, 2005.00)	(2005.00, 2008.00)	(2008.00, 2010.00)
part_51	18.90471	20.06087	21.54499	29.4578	24.77464	26.62055	28.67152
part_52	18.31694	16.05453	15.73953	17.40423	18.17121	18.87907	20.48976
part_53	17.26866	20.96095	22.53688	22.92667	20.13988	20.33338	19.44645
part_54	17.10369	15.18758	17.26866	16.2837	15.47047	16.76666	17.59952
part_55	15.73953	17.10369	15.18758	71.86667	68.32125	85.33533	86.76666
part_56	16.05453	15.73953	17.10369	18.2953	17.37472	17.53357	17.69701
part_57	18.31694	16.05453	15.73953	23.62882	21.42389	19.69034	17.61299
part_58	16.2825	18.31694	16.05453	18.67839	17.20643	17.47194	16.66766
part_59	20.36794	15.00241	16.2825	22.21389	15.80927	17.83306	19.93927
part_60	20.4207	21.28752	17.57268	20.41875	21.84623	19.43029	19.81946

Appendix C: Parameters in Deterministic Models of Design Refresh Planning for Hardware Obsolescence

n=3, there are possible 3 time points for doing design refresh during the life cycle of the system;

m=31, there are 31 types of parts become obsolete and need replacement during the life cycle of the system;

M=100 ($M \gg m$); C_1^S=850, C_2^S=950, C_3^S=900; t_1=2003.00, t_2=2005.00, t_3=2008.00

Table C-1: Costs $\boldsymbol{C_{ij}^R}$ s of hardware components for design refreshes

$\boldsymbol{C_{ij}^R}$	1	2	3	$\boldsymbol{C_{ij}^R}$	1	2	3
1(part_4)	152.8823	30.18625	147.1973	17(part_20)	36.94396	32.80518	37.3003
2(part_5)	595.6248	491.6835	97.08176	18(part_21)	11.80465	12.82777	11.39069
3(part_6)	215.03	224.1576	185.0403	19(part_22)	1270.555	1245.627	1353.586
4(part_7)	618	126.1999	131.5569	20(part_23)	310.4722	272.1242	266.7851
5(part_8)	95.73427	578	118.0317	21(part_24)	302.8545	340.6952	298.6142
6(part_9)	466.7376	317.0163	1914	22(part_25)	33.00531	35.8215	40.29728
7(part_10)	599.3817	678.4033	460.7833	23(part_26)	64.44417	47.46763	51.51783
8(part_11)	613.8627	659.2768	746.1948	24(part_27)	0.115566	2.036794	1.500241
9(part_12)	206.0614	218.6635	234.8404	25(part_32)	0.252828	0.289305	0.275438
10(part_13)	39.3807	46.31655	49.14914	26(part_33)	2.253452	2.921563	3.343082
11(part_14)	154.8549	108.98	128.174	27(part_34)	3.933411	3.120164	4.045241
12(part_15)	291.0067	339.1733	238.6952	28(part_39)	161.7791	23.2124	23.78156
13(part_16)	7.887908	6.858743	7.993983	29(part_40)	6.377324	6.730011	0.965636
14(part_17)	9.642035	10.36697	9.014348	30(part_41)	4.515179	3.965387	4.184687
15(part_18)	24.86687	30.18376	32.45311	31(part_42)	1105.744	1626.861	1428.766
16(part_19)	1470.158	1671.606	2029.019				

Table C-2: Obsolescence dates $\boldsymbol{d_i}$ s of hardware components

$\boldsymbol{i}$	$\boldsymbol{d_i}$	$\boldsymbol{i}$	$\boldsymbol{d_i}$	$\boldsymbol{i}$	$\boldsymbol{d_i}$
1(part_4)	2004.4	12(part_15)	2001.75	23(part_26)	2003.775
2(part_5)	2004.05	13(part_16)	2003.75	24(part_27)	2009.65
3(part_6)	2004.05	14(part_17)	2002.75	25(part_32)	2009.15
4(part_7)	2004.05	15(part_18)	2003.75	26(part_33)	2009.9
5(part_8)	2004.8	16(part_19)	2004.8	27(part_34)	2009.9
6(part_9)	2004.325	17(part_20)	2002.75	28(part_39)	2007.65
7(part_10)	2003.325	18(part_21)	2002.75	29(part_40)	2003.75
8(part_11)	2006.4	19(part_22)	2004.4	30(part_41)	2004.5
9(part_12)	2004.8	20(part_23)	2004.4	31(part_42)	2004.05
10(part_13)	2004.125	21(part_24)	2004.4		
11(part_14)	2004.8	22(part_25)	2003.725		

Appendix D: Parameters in Deterministic Models of Design Refresh Planning for Software Obsolescence

n=3, there are possible 3 time points for doing design refresh during the life cycle of the system;

m=10, there are 10 types of software parts become obsolete and need replacement during the life cycle of the system;

M=100 ($M \gg m$);

C_1^S=850, C_2^S=950, C_3^S=900;

Table D-1: Costs $\boldsymbol{C_{ij}^R}$ s and $\boldsymbol{C_{ik}^M}$ s of software components for design refresh and short-term mitigation

	$\boldsymbol{C_{ij}^R}$			$\boldsymbol{C_{ik}^M}$			
	1	**2**	**3**	**1**	**2**	**3**	**4**
1(part_51)	206.0614	218.6635	234.8404	321.09	270.0436	290.164	312.5196
2(part_52)	310.4722	272.1242	266.7851	295.0017	308.002	320.0002	347.3014
3(part_53)	24.86687	30.18376	32.45311	33.0144	29.00143	29.28007	28.00289
4(part_54)	36.94396	32.80518	37.3003	35.17279	33.41622	36.21599	38.01496
5(part_55)	11.80465	12.82777	11.39069	53.9	51.24094	64.0015	65.075
6(part_56)	1270.555	1245.627	1353.586	1447.89	1375.035	1387.607	1400.541
7(part_57)	310.4722	272.1242	266.7851	400.5085	363.1349	333.7513	298.5402
8(part_58)	302.8545	340.6952	298.6142	347.4181	320.0396	324.9781	310.0185
9(part_59)	64.44417	47.46763	51.51783	70.28475	50.02053	56.4238	63.08785
10(part_60)	215.03	224.1576	185.0403	215.0094	230.0408	204.601	208.6989

t_0=2000.00, t_1=2003.00, t_2=2005.00, t_3=2008.00, t_4=2010.00

Table D-2: Obsolescence dates $\boldsymbol{d_i}$ s of software components

$\boldsymbol{i}$	$\boldsymbol{d_i}$	$\boldsymbol{i}$	$\boldsymbol{d_i}$	$\boldsymbol{i}$	$\boldsymbol{d_i}$
1(part_51)	2006.8	5(part_55)	2002.75	9(part_59)	2003.775
2(part_52)	2004.4	6(part_56)	2004.4	10(part_60)	2004.05
3(part_53)	2003.75	7(part_57)	2008.4		
4(part_54)	2002.75	8(part_58)	2007.4		

Appendix E: Parameters in Design Refresh Planning Model for Component Obsolescence in ECU including Hardware and Software

n=3, there are possible 3 time points for doing design refresh during the life cycle of the system;

m=41, there are total 41 types of parts in the ECU (31 types of hardware parts and 10 types of software parts) become obsolete and need replacement during the life cycle of the system;

M=100 ($M \gg m$);

C_1^S=850, C_2^S=950, C_3^S=900;

Table E-1: Costs $\boldsymbol{C_{ij}^R}$ s and $\boldsymbol{C_{ik}^M}$ s of components in ECU for design refresh and short-term mitigation

	$\boldsymbol{C_{ij}^R}$			$\boldsymbol{C_{ik}^M}$			
	1	**2**	**3**	**1**	**2**	**3**	**4**
1(part_4)	152.8823	30.18625	147.1973				
2(part_5)	595.6248	491.6835	97.08176				
3(part_6)	215.03	224.1576	185.0403				
4(part_7)	618	126.1999	131.5569				
5(part_8)	95.73427	578	118.0317				
6(part_9)	466.7376	317.0163	1914				
7(part_10)	599.3817	678.4033	460.7833				
8(part_11)	613.8627	659.2768	746.1948				
9(part_12)	206.0614	218.6635	234.8404				
10(part_13)	39.3807	46.31655	49.14914				
11(part_14)	154.8549	108.98	128.174				
12(part_15)	291.0067	339.1733	238.6952				
13(part_16)	7.887908	6.858743	7.993983				
14(part_17)	9.642035	10.36697	9.014348				
15(part_18)	24.86687	30.18376	32.45311				
16(part_19)	1470.158	1671.606	2029.019				
17(part_20)	36.94396	32.80518	37.3003				
18(part_21)	11.80465	12.82777	11.39069				
19(part_22)	1270.555	1245.627	1353.586				
20(part_23)	310.4722	272.1242	266.7851				
21(part_24)	302.8545	340.6952	298.6142				
22(part_25)	33.00531	35.8215	40.29728				
23(part_26)	64.44417	47.46763	51.51783				
24(part_27)	0.115566	2.036794	1.500241				
25(part_32)	0.252828	0.289305	0.275438				
26(part_33)	2.253452	2.921563	3.343082				
27(part_34)	3.933411	3.120164	4.045241				
28(part_39)	161.7791	23.2124	23.78156				

29(part_40)	6.377324	6.730011	0.965636				
30(part_41)	4.515179	3.965387	4.184687				
31(part_42)	1105.744	1626.861	1428.766				
32(part_51)	206.0614	218.6635	234.8404	321.09	270.0436	290.164	312.5196
33(part_52)	310.4722	272.1242	266.7851	295.0017	308.002	320.0002	347.3014
34(part_53)	24.86687	30.18376	32.45311	33.0144	29.00143	29.28007	28.00289
35(part_54)	36.94396	32.80518	37.3003	35.17279	33.41622	36.21599	38.01496
36(part_55)	11.80465	12.82777	11.39069	53.9	51.24094	64.0015	65.075
37(part_56)	1270.555	1245.627	1353.586	1447.89	1375.035	1387.607	1400.541
38(part_57)	310.4722	272.1242	266.7851	400.5085	363.1349	333.7513	298.5402
39(part_58)	302.8545	340.6952	298.6142	347.4181	320.0396	324.9781	310.0185
40(part_59)	64.44417	47.46763	51.51783	70.28475	50.02053	56.4238	63.08785
41(part_60)	215.03	224.1576	185.0403	215.0094	230.0408	204.601	208.6989

t_0=2000.00, t_1=2003.00, t_2=2005.00, t_3=2008.00, t_4=2010.00

Table E-2: Obsolescence dates $\boldsymbol{d_i}$ s of components in ECU

$\boldsymbol{i}$	$\boldsymbol{d_i}$	$\boldsymbol{i}$	$\boldsymbol{d_i}$	$\boldsymbol{i}$	$\boldsymbol{d_i}$
1(part_4)	2004.4	15(part_18)	2003.75	29(part_40)	2003.75
2(part_5)	2004.05	16(part_19)	2004.8	30(part_41)	2004.5
3(part_6)	2004.05	17(part_20)	2002.75	31(part_42)	2004.05
4(part_7)	2004.05	18(part_21)	2002.75	32(part_51)	2006.8
5(part_8)	2004.8	19(part_22)	2004.4	33(part_52)	2004.4
6(part_9)	2004.325	20(part_23)	2004.4	34(part_53)	2003.75
7(part_10)	2003.325	21(part_24)	2004.4	35(part_54)	2002.75
8(part_11)	2006.4	22(part_25)	2003.725	36(part_55)	2002.75
9(part_12)	2004.8	23(part_26)	2003.775	37(part_56)	2004.4
10(part_13)	2004.125	24(part_27)	2009.65	38(part_57)	2008.4
11(part_14)	2004.8	25(part_32)	2009.15	39(part_58)	2007.4
12(part_15)	2001.75	26(part_33)	2009.9	40(part_59)	2003.775
13(part_16)	2003.75	27(part_34)	2009.9	41(part_60)	2004.05
14(part_17)	2002.75	28(part_39)	2007.65		

References

Anonymous (1992). Knowledge-based systems evaluation: A survey. Technical Report. University of Brescia, Department of Automation Industry, Brescia, Italy.

Anonymous (1997). Product life cycle data model. American Standard ANSI/EIA-724.

Barahona, F., Bermon, S., Gunluk, O., Hood, S. (2005). Robust capacity planning in semiconductor manufacturing. Naval Research Logistics, 52(5), 459-468.

Bergamaschi, S., Castano, S., De Capitani di Vimercati, S., Montanari, S., Vincini, M. (1998). An intelligent approach to information integration. First International Conference on Formal Ontology Information Systems, 253-268.

Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web. Scientific American, 284(5), 34-43.

Blumenfeld, S. (2007). Predicting the end. Broadcast Engineering, 90.

Brickley, D., Guha, R.V. (2004). RDF vocabulary description language 1.0: RDF schema, W3C working draft. http:// www.w3.org/TR/PR-rdf-schema.

Bureau of Transportation Statistics (2006). Transportation statistics annual report. Washington, D.C.: U.S. Department of Transportation, Research and Innovative Technology Administration.

Carbone, J. (2006, December 14). RoHS means more electronic component obsolescence. Purchasing, 25-28.

Chang, X. (2008). Ontology development and utilization for knowledge management in product design. Dissertation. Virginia Tech.

Chang, X., Sahin, A., Terpenny, J. (2008). An ontology-based support for product conceptual design. Robotics and Computer-Integrated Manufacturing, 24(2008), 755-762.

Chang, X., Terpenny, J. (2009). Ontology-based data integration and decision support for product e-Design. Robotics and Computer-Integrated Manufacturing, 25(2009), 863-870.

Charlesworth, K. (2007, July). Turn a tidy profit from any old iron. Printing World, 27.

Cranefield, S. (2001). Networked knowledge representation and exchange using UML and RDF. Journal of Digital Information, 1(8), article no. 44.

De Grip, A., Van Smoorenburg, M., Borghans, L. (1997). The Dutch observatory on employment and training, ROA-W-1997/2E, Research Centre for Education and the Labour Market, Maastricht.

Denny, M. (2004). Ontology editor survey results. http://www.xml.com/2004/07/14/examples/Ontology_Editor_Survey_2004_Table_-_Michael_Denny.pdf.

Department of Defense. Department of Defense Regulation 4140.1-R, DoD supply chain management regulation.

Drake, M (2003). Sharing obsolescence information at Raytheon with the component obsolescence and reuse tool. Proceedings of the Annual Reliability and Maintainability Symposium.

Cunningham, J. R. (2002). Demonstration of physics-of-failure interconnect assessment of an avionic controller. Dissertation. The University of Maryland.

Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F. (2001). OIL: An ontology infrastructure for the Semantic Web. IEEE Intelligent Systems, 14(1), 37-46.

FOLDOC (2001). Unified Modeling Language. http://foldoc.org/UML.

Gašević, D., Djurić, D., Devedzić, V. (2006). Model driven architecture and ontology development. Berlin: Springer-Verlag.

Gašević, D., Djurić, D., Devedzić, V., Damjanović, V. (2004). Converting UML to OWL ontologies, Proceeding of the 13th International World Wide Web Conference on Alternate Track Papers and Posters, NY, 488-489.

Gómez-Pérez, A., Corcho, O. (2002). Ontology languages for the Semantic Web. IEEE Intelligent Systems, 17(1), 54-60

Goggins Jr., P.W., (2007, June). Time to replace an aging fleet of commercial and industrial boilers. Boiler Systems Engineering, 7.

Gruber T.R. (1993). A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2), 199-220.

Hendler, J., McGuinness, D. (2000). The DARPA agent markup language. IEEE Intelligent Systems, 15 (6), 72-73.

Henke, A.L., Lai, S. (1997). Automated parts obsolescence prediction. Proceeding of the DMSMS Conference.

Hillier, F.S., Lieberman, G.J. (1990). Introduction to Operation Research, 5th Edition, McGraw-Hill International.

Hoorickx, P.F. (2008). Sustaining engineering challenges in long field medical electronics. Proceedings of the SMTA Medical Electronic Symposium, Anaheim, CA.

Horridge, M. (2009). A practical guide to building OWL ontologies using Protégé 4 and CO-ODE tools. The University of Manchester.

Horrocks, I. (2002). DAML+OWL: a reasonable web ontology language. Proceedings of EDBT 2002, Lecture Notes in Computer Science 2287, 2-13.

Jiang, L., Qu, F. (2006). Study on construction and application of product requirement design ontology. International Technology and Innovation Conference, 2006.

Johnson, W.J. (2002). Defence logistics agency component obsolescence management. Proceedings of Sixth Joint FAA/DoD/NASA Ageing Aircraft Conference.

Josias, C., Terpenny, J.P. (2004). Component obsolescence risk assessment. Proceedings of the 2004 Industrial Engineering Research Conference (IERC).

Kashyap, V., Sheth, A. (1997). Semantic heterogeneity in global information systems: the role of metadata, context and ontologies. Cooperative Information Systems, 139-178.

Kim, W., Seo, J. (1991). Classifying schematic and data heterogeneity in multidatabase systems. IEEE Computer, 24(12):12-18.

Kim, K., Manley, D.G., Yang, H. (2006). Ontology-based assembly design and information sharing for collaborative product development. Computer-Aided Design, 38 (2006), 1233-1250.

Koutsoukis, N.S., Mitra, G. (2003). Decision modeling and information systems: the information value chain. Kluwer Academic Publisher, ORCS 26, 1-4020-7560-X.

Lassila, O. (1998). Web metadata: A matter of semantics. IEEE Internet Computing, 2(4), 30-37.

Lassila, O., Swick, R. (1999). Resource description framework (RDF) model and syntax specification. http://www.w3.org/TR/REC-rdf-syntax/.

Li, F., Feng, Y., Wang, C., Yang, X. (2008). Research on product information organization based on ontology in the network. Wireless Communications, Networking and Mobile Computing, 2008.

Livingston, H. (2000). GEB1: Diminishing manufacturing sources and material shortages (DMSMS) management practices. Proceedings of the DMSMS Conference.

Lohse, N., Valtchanov, G., Ratchev, S., Onori, M., Barata, J. (2005). Towards a unified assembly system design ontology using protégé. 8th Intl. Protégé Conference, Madrid, Spain.

Manola, F., Miller, E. (2004). RDF primer, W3C recommendation. http:// www.w3.org/TR/REC-rdf-syntax.

McDermott, J., Shearer, J., Tomczykowski, W. (1999). Resolution cost factors for diminishing manufacturing sources and material shortages. ARINC. Retrieved February 21, 2006, http://smaplab.ri.uah.edu/dmsms98/papers/trunnell.pdf. Supplemental Report, Resolution cost factors for diminishing manufacturing sources and material shortages. ARINC, 2001.

McGuinness, D.L., Fikes, R., Rice, J., Wilder, S. (2000). An environment for merging and testing large ontologies. Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000), San Francisco, CA.

McGuinness, D. L., Wright, J. (1998). Conceptual modeling for configuration: a description logic-based approach. Artificial Intelligence for Engineering Design, Analysis, and Manufacturing-special issue on Configuration.

Meixell, M., Wu, S. D. (2001). Scenario analysis of demand in a technology market using leading indicators. IEEE Transactions on Semiconductor Manufacturing, 14 (1), 65-78.

Mishra, S. (1997). Visual Modeling and Unified Modeling Language (UML): introduction to UML. Rational Software Corporation.

Mitra, G. (1988). Models for decision making: an overview of problems, tools and major issues. Mathematical Models for Decision Support, NATO ASI Series, Springer-Verlag.

Musen, M. A. (1992). Dimensions of knowledge sharing and reuse. Computers and Biomedical Reseach, 25, 435-467.

Mutschler, A. S. (2008, January 14). Used semiconductor equipment market on the rise. Electronic News, 53.

Nelson III, R., Sandborn, P. (2010). Managing coupled hardware and software obsolescence using constraint-driven design refresh planning. Proceedings of DMSMS conference, Las Vegas, NV.

Nelson III, R., Sandborn, P. (2011). Strategic management of component obsolescence using constraint-driven design refresh planning. International Journal of Product Lifecycle Management (Submitted).

Neumann, S., Weiss, A. (1995). On the effects of schooling vintage on experience-earnings profiles: theory and evidence. European Economic Review, Vol.39, No.5, pp.943-55.

Noy, N.F., McGuiness, D.L. (2001). Ontology development 101: a guide to creating your first ontology. Stanford Knowledge Systems laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880.

Noy, N. F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R.W., Musen, M.A. (2001). Creating semantic web contents with Protégé-2000. IEEE Intelligent Systems.

O'Connor, M. J., Das, A. K. (2009). SQWRL: a query language for OWL. Proceedings of OWL: Experiences and Directions 2009 (OWLED 2009), Fifth International Workshop, Chantilly, VA.

OMG ODM RFP (2003). OMG Ontology Definition Metamodel Request for Proposal, OMG document: ad/2003-03-40. http:// www.omg.org/cgi-bin/doc?ad/2003-03-40.

Pecht, M., Solomon, R., Sandborn, P., Wilkinson, C., Das, D. (2002). Life cycle forecasting, mitigation assessment and obsolescence strategies. CALCE EPSC Press.

Power, D.J. (2002). Decision support systems: concepts and resources for managers. Westport Conn., Quorum Books.

Prophet, G. (2002). Guarding against component obsolescence. EDN, 47(25), 63-73.

Rai, R., Terpenny, J. (2008). Principles for managing technological product obsolescence. IEEE Transactions on Components and Packaging Technologies, 31(4), 880-889.

Sandborn, P. (2007). MOCA user's guide. ESCML 07-04.

Sandborn, P., Mauro, F., Houston, J., Singh, P. (2003). Optimum technology insertion into systems based on the assessment of viability. IEEE Transaction on Components and Packaging Technologies, 26, 734-738.

Sandborn, P., Jung, R., Wong, R., Becker, J. (2007). A taxonomy and evaluation criteria for DMSMS tools, databases and services. Proceedings of the Aging Aircraft Conference, Palm Springs, CA.

Sandborn, P., Mauro, F., Knox, R. (2007). A data mining based approach to electronic part obsolescence forecasting. IEEE Trans. On Components and Packaging Technologies, 30(3), 397-401.

Singh, P., Sandborn, P. (2006) Obsolescence driven design refresh planning for sustainment-dominated systems. The Engineering Economist, 51(2), 115-139.

Smith, M.K., Welty, C., McGuinness, D.L. (2004). OWL Web Ontology Language Guide, W3C. http://www.w3.org/TR/owl-guide.

Solomon, R., Sandborn, P., Pecht, M. (2000). Electronic part life cycle concepts and obsolescence forecasting. IEEE Trans. On Components and Packaging Technologies, 707-717.

Stogdill, R.C. (1999). Dealing with obsolete parts. IEEE Design and Test of Computers, 16(2),17-25.

Stuckenschmidt, H., van Harmelen, F. (2005). Information sharing on the semantic web. Berlin: New York: Springer.

Terpenny, J. (1997). An Integrated System for Functional Modeling and Configuration in Conceptual Design. Proceedings of the Seventh International Flexible Automation and Intelligent Manufacturing Conference, Middlesbrough, U.K., 69-80.

Terpenny, J., Strong, S., Wang, J. (2000). A methodology for knowledge discovery and classification. Proceedings of the Tenth FAIM 2000 - Flexible Automation And Intelligent Manufacturing Conference, University of Maryland, College Park, Maryland.

Terpenny, J., Sandborn, P., Rai, R. (2009). Collaborative research: knowledge representation and design for managing product obsolescence. Proposal Submitted to National Science Foundation (NSF Grants 0928530, 0928628, 0928837, July 2009-July 2011).

Tomczykowski, W. J. (2003). A study on component obsolescence mitigation strategies and their impact on R and M. Proceedings of the Annual Reliability and Maintainability Symposium, 332-338.

The University of Massachusetts Amherst Center for e-Design. Ontology downloads. http://edesign.ecs.umass.edu/?page_id=52.

Uschold, M., Gruninger, M. (1996). Ontologies: principles, methods and applications. Knowledge Engineering Review 11(2).

US Air Force (2003). Air force handbook for the 108th congress, first session.

Van Loo, J., De Grip, A., De Steur, M. (2001). Skills obsolescence: causes and cures. International Journal of Manpower, Vol.22, No.1/2, 121-137.

Vegetti, M., Henning, G.P., Leone, H.P. (2005). Product ontology definition of an ontology for the complex product modeling domain. 2nd Mercosur Congress on Chemical Engineering, 4th Mercosur Congress on Process System Engineering.

Wolberg, J. (2006). Data analysis using the method of least squares: extracting the most information from experiments. Berlin, New York, Springer C2006.

Wright, M., Humphrey, D., McCluskey, P. (1997). Uprating electronic components for use outside their temperature specification limits. IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part A, 20(2), 252-256.

W3C (2004). SWRL: a semantic web rule language combining OWL and RuleML. http://www.w3.org/Submission/SWRL/.