An Ontological Model and Method for Obsolescence Resolution and Management

Liyu Zheng[1], Peter Sandborn[2], Janis Terpenny[1], and Nihal Orfi[1*]

[1]Industrial and Manufacturing Systems Engineering
Iowa State University
Armes, IA, 50011, USA

[2]Department of Mechanical Engineering
University of Maryland,
College Park, MD, 20742, USA

ABSTRACT

With today's rapid advancement in technologies, commercial high tech components are rendered obsolete more frequently. Such components are often parts of long-life products. Hence, technology obsolescence can make design changes for systems expensive and result in high life-cycle costs. Several tools have been developed to support obsolescence management. While these tools provide benefits and serve as focal points for information related to components, these tools are limited in many respects due to data conflicts, incompleteness and inconsistency. Further, the lack of communication between these tools compounds the limitations making the proactive management of obsolescence even more challenging. This paper addresses gaps in existing tools, by providing a framework capable of integrating heterogeneous sources of information and knowledge that are required to resolve and manage obsolescence. This is implemented through an obsolescence resolution ontological model developed using the ontology editor Protégé. A reasoning method is used to query obsolescence information and resolution strategies to support decision-making. This paper includes two case studies describing results from the application of the ontological model and method and how these facilitate sharing and exchange of knowledge critical to obsolescence management.

1. INTRODUCTION

Many low volume products and systems, such as those utilized by military and avionics applications, often make use of commercial high-tech components. In the past decade, technology has advanced very rapidly causing such components to have a shortened procurement life span. Newer technologies are being introduced frequently, rendering components obsolete. Yet, custom low volume products and systems such as submarines and aircraft can be in use for decades [1] [2]. There are different types of obsolescence; in this paper the type of obsolescence addressed is referred to as DMSMS (Diminishing Manufacturing Sources and Material Shortages), which is defined as the loss of the ability to procure part (component) or technology from its original manufacturer [3].

Obsolescence mitigation and management strategies are critical to maintaining long field-life products and systems. There are multiple reactive methods for obsolescence mitigation, which consist of reallocating stock, replacing the part, substituting or repairing the part, executing a lifetime buy of the part, reclamation, or redesigning the part or system [4]. In addition to the reactive obsolescence mitigation methods, various proactive and strategic tools are used. The most prominent effort by the Department of Defense (DoD) to address the issue of obsolescence is the launch of the DMSMS Knowledge Sharing Portal (KSP) [5]. At the heart of the KSP is the Shared Data Warehouse, which is a web-based system to identify and solve obsolescence associated problems. Despite progress in similar tools and benefits with information related to components and sources for component acquisition, these tools are limited in many respects due to data conflicts, incompleteness and inconsistency between sources. Additionally, communication between existing tools is lacking, resulting in a significant barrier to efficiently managing obsolescence. To address these problems, an interoperable ontological framework has been developed in which knowledge related to obsolescence management is stored to facilitate the usage of these tools.

To develop a knowledge base, there is a need to capture the abstract, or knowledge used by engineers during

* Corresponding Author: Tel.: (540) 808-9173; E-mail: norfi@iastate.edu

component obsolescence management. This knowledge may include the rationale behind decisions such as when to invest, what technology to invest in, or whether to wait until a future point in time. Ontology provides a formal method for capturing the abstract knowledge and has become a very popular form of knowledge storing and sharing within the last 20 years due to its broad application and many advantages over other forms of knowledge sharing [6]. In this paper, an obsolescence resolution ontology is described that has been developed to capture and manage knowledge used for dealing with the obsolescence of components occurring in long field-life systems.

2. DEVELOPMENT OF THE OBSOLESCENCE RESOLUTION ONTOLOGY

Ontology is used to define a common vocabulary for sharing information in the domain. Some reasons to develop ontologies are: (1) to share common understanding of the structure of information, (2) to enable reuse of domain knowledge, (3) to separate domain knowledge from operational knowledge, and (4) to analyze domain knowledge [7]. In practice, developing ontology includes defining classes that indicate concepts in a domain and arranging them in a taxonomic hierarchy (classes, subclasses, and super classes), defining slots for classes that indicate the properties of classes and creating instances of classes by specifying allowed values for these slots. Accordingly, there are five main steps in ontology development. A brief description of these steps along with the development of the obsolescence resolution ontology is provided next.

Step 1. Determine the domain and scope of the ontology. In this step, motivation, purpose, scope, and requirements for the proposed ontology need to be specified according to intended applications. A variety of modeling language tools could be used to facilitate and visualize the design of the ontology. For example, UML (Unified Modeling Language) [8] is a standardized general-purpose modeling language in the field of object-oriented software engineering. UML for ontology development can start from the design of use cases, which can identify the goal and uses of the system from the users' perspectives. Then classes that support use cases can be enumerated. An obsolescence resolution ontology can then be developed for managing knowledge in the decision-making process. Its development starts with studying the use cases associated with obsolescence.

Step 2. Define the classes and the class hierarchy. Classes are defined for the concepts identified in Step 1. The class hierarchy is one unique characteristic of the ontology, which arranges abstract concepts on the top and specific concepts on the bottom. There are three possible approaches in developing the class hierarchy: top-down, bottom-up and a combination or blended approach to top-down and bottom-up approaches [9]. A combination development process is often used to develop the class hierarchy, since it combines the advantages of the other two approaches: A top-down development starts with the definition of the most general class and subsequent specialization of the classes. On the contrary, a bottom-up development starts with definition of the most specific classes with subsequent grouping of these classes into more general concepts.

Step 3. Define the slots of the classes. There are two types of slots. One type of slot has data type value (e.g., integer, float, Boolean, string, date, etc.). The other type has a value that is another class. If a slot's value is an instance of another class, this slot represents a relation that links the two classes together.

For steps 2 and 3 in the developed ontology: obsolescence resolution class is defined as a subclass of resolution class, and it inherits the slots "ID", "name" and "description" from the resolution class. Resolution class is a more abstract class that has a similar role as other abstract classes such as object, criterion and constraint. Obsolescence resolution class can cooperate with other classes to support a broader problem via resolution class. In addition to the slots "ID", "name" and "description" that are for identifying the specific resolution method and and providing information of that method, obsolescence resolution class has the following slots, as shown in Table.

Table 1. Obsolescence Resolution Class Slots.

Slot(s)	**Usage**
Issue	To describe the obsolescence problem
Criteria and Constraints	To determine the obsolescence resolution
Person	To designate who executes the obsolescence resolution
Date	To identify when the resolution takes place
Cost	To state the cost of performing the selected resolution
Result	To describe outcome after performing the resolution
Limitation	To describe the limitation of the selected resolution
System	To describe the long field-life system that has the obsolescence problem (e.g. aircraft, ship...etc.)
Component	To state the obsolete component to which the obsolescence resolution is applied

Among these slots, the value type of the slots “ID”, “name”, “description’’, “issue”, “criteria”, “constraints”, “result”, and “limitation” are string, the value type of the slots “system” and “component” are instance, and the value type of the slots “date” and “cost” are float. In addition, the slot “ID” can only have single (unique) value, since it is used to identify the resolution.

Step 4. Create the instances. Instances of classes can be created by specifying values of slots and thus allows the wide application of knowledge captured. In the developed ontology, obsolescence resolution classes are organized into four categories defined by the method of obtaining the new component for replacing the obsolete component. In each category, there are four specific resolutions that are organized in the hierarchical structure. These four categories are:

Category 1. Same Component From The Same Source. This category of resolutions is always the first choice. Using the same component from the same source helps avoid problems caused by a new component that might be incompatible with the system. There are several resolutions in this category:

1. Controlled Inventory: This is the inventory that the organization has always maintained control over. It is divided into two subclasses: Existing Stock, where the inventory is in place prior to the obsolescence event; and Buy and Hold, where the inventory is obtained as a result of a purchase precipitated by the obsolescence event (i.e., and lifetime buy or bridge buy).
2. Reclamation/Salvage: Using components that were used in a previous product – noting that the previous product does not have to be the same product.
3. Aftermarket: New (finished or complete) components that are obtained from an approved aftermarket source.
4. Negotiate with the Source: Negotiate with the original source to continue offering the component.

Category 2. Same Component From A Different Source. In many cases, it is difficult to get the same component from the same source. Two main resolutions to be considered in this category:

1. Retarget Source: A source that will specifically “remake” the component. Two subclasses are considered: Design Information Available, where information can be obtained to support the source; and Reverse Engineering.
2. Non-Specified or Non-Qualified Source: This is a source of the component that has not been previously used (i.e., not specified in the original specification of the system).

Category 3. Different Component. If the same component is not available, a different component that will be substituted in place of the original component is needed. This category includes:

1. Same Component (With Administrative Change): The new component is same as the original component but the “part number” or some other administrative change has been made by the source.
2. Same Source Similar Component: A slightly different version of the component provided by the same source.
3. Different Component: A completely new component.

Category 4. More Than A Component Change. Differing from the resolutions in the other three categories that focus on an individual component, the resolutions in this category are carried out on a system basis. These resolutions are considered to be a way of strategically managing the obsolescence problem to achieve an optimal solution. This category includes the following resolutions:

1. Design Refresh: Design refresh targets the same performance and functionality as the original system. The challenge is when to schedule refresh activities and their content in order to minimize cost.
2. Redesign: Targets a change in performance and/or functionality of the system.

Qualification is a necessary part of obsolescence management. Qualification is the testing of a component and/or the product that the component goes into to insure that it performs as specified where “as specified” means both functionally and from a reliability standpoint. Qualification can be performed at many different levels. The requirement for qualification differs by the criticality of the application, and not simply by the part. Qualification resolutions include:

1. Drop-In: No special qualification of the component is necessary.
2. Source Evaluation/Qualification: Perform an evaluation of the source and qualify its capability and the quality of components that it produces.
3. Component-Level Testing and Re-Qualification: Test the component to make sure it is qualified for use.
4. Subsystem-Level Testing and Re-Qualification: A subsystem needs to be qualified.
5. System-Level Testing and Re-Qualification: The whole product or system needs to be qualified.

Step 5. Evaluate the developed ontologies. From the view of logic, ontologies are required to be consistent, complete, and concise. From the view of application, ontologies are required to be valid, usable, reliable, and effective [10]. Methods used to evaluate the ontologies include expert judgments, experimental testing, and so on. Finally, developing an ontology is an iterative process. Ontologies need to be continuously revised and refined by going through these development steps.

2.1. REASONING FOR OBSOLESCENCE RESOLUTION

The obsolescence resolution class taxonomy that has been introduced builds a knowledge base. Reasoning mechanisms are then used to search the obsolescence resolution knowledge base. There are two main reasoning methods, backward-chaining and forward-chaining. The backward chaining starts with a list of goals and works backwards to find proof for the goals. An inference engine for the backward chaining would search the inference rules until it finds one which has a ***Then*** clause that matches a desired goal. In contrast, the forward chaining starts with the available data and uses inference rules to draw conclusions. An inference engine for the forward chaining searches the inference rules until it finds one where the ***If*** clause is known to be true. Once found, it can conclude, or infer, the ***Then*** clause, resulting in the addition of new information to its dataset [11].

In the approach presented here, forward chaining is used for obsolescence resolution reasoning. The rules are written using SWRL (Semantic Web Rule Language). SWRL rules have a format that is consistent with the principle of the forward chaining. It is in the form of an implication between an antecedent (***If*** clause) and a consequent (***Then*** clause). An antecedent and a consequent are both conjunctions of atoms. Atoms are basic units that SWRL rules consist of, and 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, instances or data values. Some example rules for obsolescence resolution reasoning are provided in Table 2.

Table 2. Rules for obsolescence resolution reasoning.

Rules	Function
Obsolete Component (?x) ∧ sameAs (?x, ?y) ∧ New Component (?y) ∧ Obsolescence Resolution (?z) ⟶Same Component (?z) Obsolete Component (?x) ∧ differentFrom (?x, ?y) ∧ New Component (?y) ∧ Obsolescence Resolution (?z) ⟶Different Component (?z) Obsolete Component Source (?x) ∧ sameAs (?x, ?y) ∧ New Component Source (?y) ∧ Obsolescence Resolution (?z) ⟶Same Source (?z) Obsolete Component Source (?x) ∧ differentFrom (?x, ?y) ∧ New Component Source(?y) ∧ Obsolescence Resolution (?z) ⟶Different Source (?z)	Determine same component, different component, same source, different source
Same Component From A Different Source (?z) ∧ Different Component (?z) ∧ More Than A Component Change (?z) ∧ Obsolescence Resolution (?z) ⟶ Qualification (?z)	Determine whether qualification is needed

3. CASE STUDIES

Two case studies are presented as examples to illustrate the application of the obsolescence resolution ontology in the management of product obsolescence. The ontology is developed in the ontology editor Protégé.

3.1. CASE STUDY I: SIMPLE RESOLUTION FOR COMPONENT OBSOLESCENCE IN AN AIRCRAFT ECU

Aircraft (military and commercial) are in service for long time periods. According to the recorded life spans of defence systems, the B-52, a long-range, subsonic, jet-powered strategic bomber, has a service life of more than 85 years (1955-2040+), and the KC-135, an aerial refueling military aircraft, has a service life of more than 60 years (1957-2017+). For other aircraft such as the F-15 and UH-1, service life are at least 35 years (1975-2010+) and 45 years (1959-2004+) respectively. There is also a high likelihood that service life spans of current aircraft will be extended, as new aircraft are not yet available.

Obsolescence of subsystems and components is common in aircraft because of long life cycles. This example considers the ECU (engine control unit) subsystem. The ECU is an electronic subsystem used to control the engine of aircraft to produce a determined amount of thrust by monitoring engine parameters. The ECU consists of three hardware boards (I/O, CPU, EMI) populated with hundreds of electronic components (e.g. microprocessor, memory, etc.). Electronic component obsolescence has been identified as a serious problem affecting the maintenance of aircraft. Figure 1(b) shows the average introduction rates for new generations of commercial integrated circuits. The shortest rate is only 9 months with none greater than 10 years - a fraction of the aircraft life span.

(a)ECU

Device Category	Average Introduction Rate
Logic Families	6 years
Memory Families	9 months
Microprocessors	2 years
Digital Signal Processor (DSP)	3 years
Programmable Logic Device (PLD)	1 year
Linear Interfaces	8 years
Gate Arrays	2 years

(b)Electronic components' average introduction rate

Figure 1. ECU and electronic components' average introduction rate [3] [8].

Obsolescence of individual electronic components in the ECU subsystem could be managed using the simple resolutions from the first three categories: Same Component From The Same Source, Same Component From A Different Source, and Different Component. The knowledge of a resolution used to solve the obsolescence problem of a component is captured as an ontology instance. The instance is created in Protégé, with the following details:

- **ID:** Obsolescence Resolution OR000123
- **name:** Substitute an obsolete microcircuit in the ECU using different component
- **description:** The microcircuit AB-1053 is obsolete in the ECU that is used to control the engines of the aircraft. See the file AB-1053 for data information of the microcircuit AB-1053. *(The data file provides detail information about the component including features, applications, diagram, etc.)*
- **issue:** The microcircuit AB-1053 is obsolete in the ECU A380
- **system:** ECU A380
- **component:** Microcircuit AB-1053. *(The images of the system and component could be displayed in Protégé to help visually learn the object that has the obsolescence problem.)*
- **criteria**: Minimize cost, maximize reliability
- **constraints:** Compatible interface, same function, source, budget
- **result:** The total number of 10 obsolete microcircuit components AB-1053 need to be dealt with. The microcircuit DE-452 is used to substitute the microcircuit AB-1053. See the file DE-452 for the data information of the microcircuit DE-452.
- **cost:** $1000.0
- **limitation:** Qualification is needed to make sure the microcircuit DE-452 work normally in the system
- **person (responsible for):** Tenold Cole
- **date:** 2012 *(The format of (start date — ending date) could be used to clarify the time length.)*

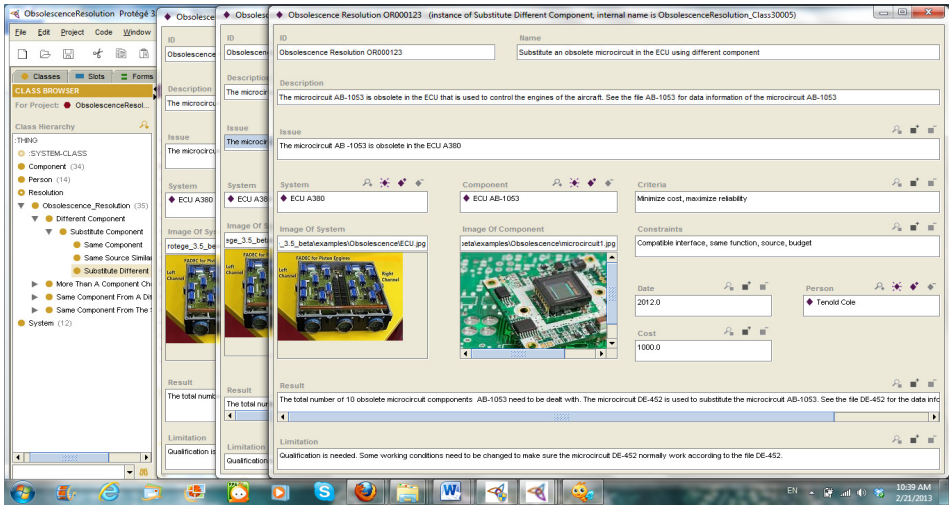

Figure 2. The obsolescence resolutions for the obsolete component in the ECU.

Figure 2 shows how the ontology can be used to sort the information, showing the different solutions to the problem cascaded next to each other as individual instances. This helps compare different obsolescence resolutions and determine the most appropriate resolution to apply to an obsolete component. The decision making process is shown in Figure 3. A pool of obsolescence resolutions that could be used to manage a component's obsolescence have been represented as ontology instances. The decision priority is given to the resolutions from the category of Same Component From The Same Source, since these resolutions will not cause incompatibility between component and system (no qualification required). SWRL rules in Table 1 are used to search this category of resolutions. There are three possible query results. The first result is that only one Same Component From The Same Source is found, and then this is the best resolution for the obsolete component. The decision making process is completed. The second result is that several Same Component From The Same Source resolutions are found. Further, a decision needs to be made on this subset of resolutions to determine the best resolution. The third result is that no Same Component From The Same Source resolution is found. In this case, other categories of resolutions need to be considered. For the latter two results, further decisions need to be made using other criteria such as lower cost until the best resolution is determined.

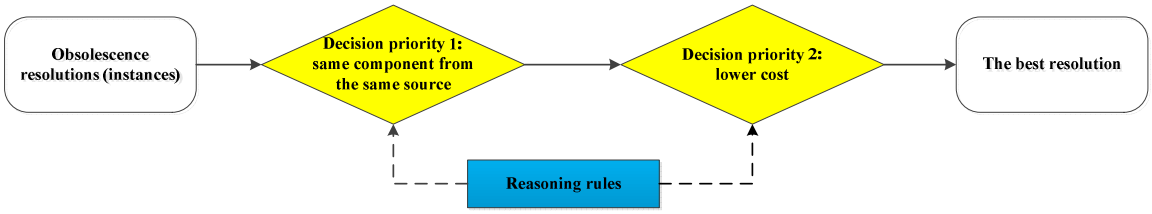

Figure 3. Determine the best resolution for an obsolete component.

The obsolescence resolutions discussed in this case study for the aircraft ECU are on a component-specific basis. The best resolution is determined from the aspect of a single component, but not from the whole system (i.e., the whole ECU). In many cases, individual component resolutions that optimize the obsolescence management for the whole system are required. How ontology is used to capture knowledge in the development of such kinds of resolution will be discussed in the next case study.

3.2. Case Study II: Complex Resolution for Nuclear Power Plant I&C System Obsolescence

An aging and obsolete instrumentation and control (I&C) system is often the main obsolescence problem in nuclear power plants [9].The I&C system consists of control equipment, sensors and transmitters, wire systems, process to sensor interfaces, and other devices and components. It controls all facets of plant operation to realize the plant's safe and stable performance. Operating nuclear plants still use a great deal of analog I&C equipment designed and built decades ago. The equipment has components that are no longer manufactured or even available in secondary markets. I&C system obsolescence is exacerbated by the fact that most nuclear power plants are now planning to extend their operating licenses by 20 years. As a result, nuclear power plants will need to take steps to ensure that each I&C system will be able to perform its intended function until the plant is retired. I&C system obsolescence in the nuclear power plant needs to be managed with a strategy that is comprehensive and dynamic in order to manage existing and future plant control systems.

The strategy used to manage the obsolescence of an I&C system in the nuclear plant could be classified as the refresh resolution (Category 4). A methodology for design refresh planning has been proposed by Singh and Sandborn [10]. This methodology determines the best dates for performing design refreshes, and the optimum mixture of actions to take at those design refreshes. A design refresh planning model that could be used to manage the I&C system obsolescence in the nuclear power plant is summarized in Figure 4. These models generate design refresh plans that are a group of zero, one or more design refreshes that will be performed on the I&C system during its lifetime (zero is possible as no-refresh solution is a viable alternative). When a design refresh is encountered, long-term mitigation solutions (e.g., substitute part, emulation, uprate similar part, etc.) are applied until the end of the system life or possibly until some future design refresh. Qualification may be needed if design change occurs. During each time period between two contiguous 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.

Input:

- The bill of material (BOM) information of the system including components' procurement costs and forecasted obsolescence dates
- System and part demand over time

Goal:

- Minimize life-cycle cost
- Maximize "value" (performance, reliability, etc.)

Output:

- When to design refresh
- What obsolete system components should be replaced at a specific design refresh (versus continuing with some other obsolescence mitigation strategy)
- What non-obsolete system components should be replaced at a specific design refresh

Flowchart:

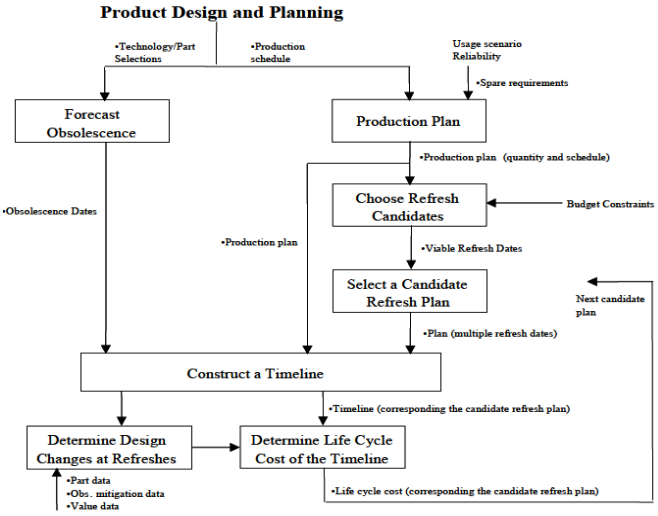

Figure 4. Summary of the design refresh planning model [12].

The result of design refresh planning for the nuclear power plant I&C system is a group of design refreshes scheduled at different times during the system's lifetime: 2010, 2018, 2026, 2036, and 2045 (study performed in 2008). At each design refresh, different obsolescence resolutions will be applied on different types of components in the system. For example, at the first design refresh in 2010, for components 1 and 2, the resolution of substituting with different components is used; for component 3, the resolution of existing stock is used; and so on. The specific obsolescence resolution applied on a component utilizes information in the same manner as introduced in the Case Study I. The resolutions applied on the components could be displayed as ontology instances on the timeline, as shown in Figure 5. This case study depicts one of the advantages of ontology; that it captures the progress of design refresh planning applied on the system with the update of the obsolescence resolution information as t time moves on. This makes it possible to view and compare different obsolescence resolutions applied on the component at different points in time.

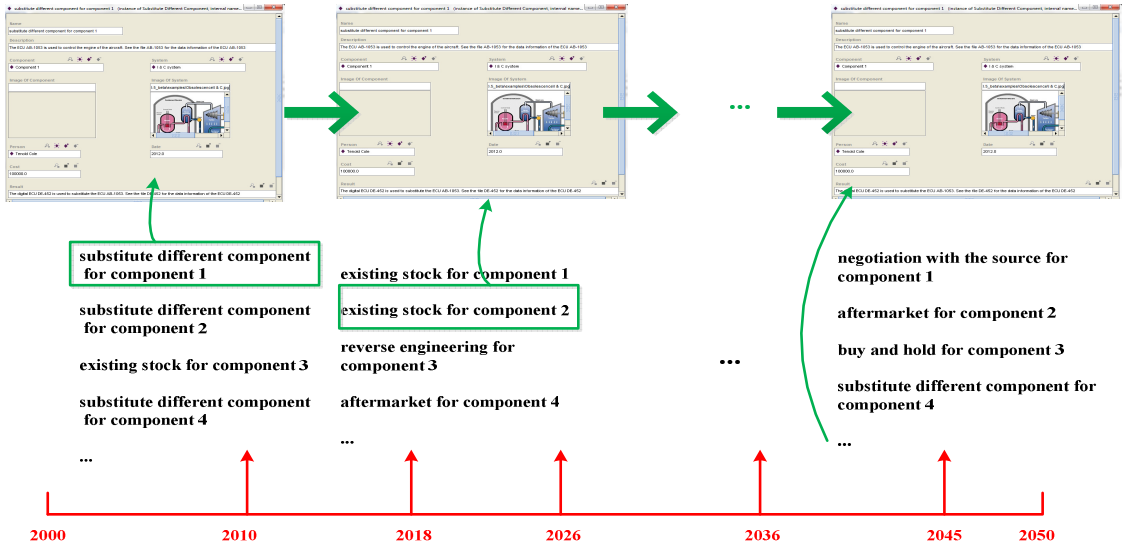

Figure 5. Obsolescence resolutions in a design refresh plan display on timeline.

4. CONCLUSIONS AND FUTURE WORK

Component obsolescence is a significant problem facing long field-life systems. Current tools are lacking in needed information and knowledge and provide limited support for the obsolescence management. To address this problem, an ontology-based approach has been utilized to capture the knowledge of resolutions used in the obsolescence management and represent it in an explicit and consistent manner.

In this paper, an obsolescence resolution ontology is developed for product obsolescence management. Obsolescence resolutions are represented using ontology class taxonomy. The capability of the obsolescence resolution ontology has been demonstrated with two case studies. Case study I demonstrates the recording and comparison of obsolescence resolutions applied to each individual obsolete component in the system. Case study II demonstrates the tracking of progression of a group of obsolescence resolutions applied to strategically maintaining an obsolete subsystem as a function of time. The developed obsolescence resolution ontology is capable of working together with other ontologies such as component and assembly ontologies, thereby providing a more extensive knowledge base to support obsolescence management more broadly.

In the future, with the development of technology, new and advanced obsolescence resolutions will be added to enrich the current knowledge base. The knowledge base needs to be maintained periodically to ensure it is not outdated. For the purpose of commonly sharing knowledge for better obsolescence management, the obsolescence resolution could be developed with OWL (web ontology language) and be published on the web in XML format.

ACKNOWLEDGMENTS

This work was funded by the National Science Foundation through Grant Nos. 0928530, 0928628, and 0928837. Any opinions, findings, and conclusions or recommendations presented in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] C. Josias, J. Terpenny and K. McLean, "Component Pbsolescence Risk Assessment," in Industrial Engineering Research Conference (IERC), 2004.

[2] P. Sandborn, "Trapped on Techonlogy's Trailing Edge," IEEE Spectrum, pp. 42–45, 2008.

[3] P. Sandborn, V. Prabhakar and O. Ahmad, "Forecasting Electronic Part Procurement Lifetimes to Enable the Management of DMSMS Obsolescence," Microelectronics Reliability, vol. 51, pp. 392–399, 2011.

[4] R. Stogdill, "Dealing with Obsolete Parts," IEEE Design and Test of Computers, vol. 16, no. 2, pp. 17–25, 1999.

[5] J. McDermott, "Reducing the Impact of Obsolescence in Military Systems," in DMSMS , New Orleans, Louisiana, 2002.

[6] P. Witherell, S. Krishnamurty and I. Grosse, "Ontologies for Supporting Engineering Design Optimization," ASME Journal of Computing and Information Science in Engineering, vol. 7, pp. 141–150, 2007.

[7] N. Noy and D. McGuiness, "Ontology Development 101: A guide to creating your first ontology," in Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, 2001.

[8] P. Kogut, S. Cranefield, L. Hart, M. Dutra, K. Baclawski, M. Kokar and J. Smith, "UML for Ontology Development," The Knowledge Engineering Review, vol. 17, no. 1, pp. 61–64, 2002.

[9] M. Uschold and M. Gruninger, "Ontologies: principles, methods and applicaitons," Knowledge Engineer Review, vol. 11, no. 2, 1996.

[10] G. Guida and G. Mauri, "Evaluating Performance and Quality of Knowledge-Based Systems: Foundation and Methodology," IEEE Transaction on Knowledge and Data Engineering, vol. 5, no. 2, pp. 204–224, 1993.

[11] B. Coppin, Artificial Inteliigence Illuminated, Jones and Barlett, 2004.

[12] P. Singh and P. Sandborn, "Obsolescence Driven Design Refresh Planning for Sustainment-Dominated Systems," The Engineering Economist, vol. 51, no. 2, pp. 115–139, 2006.