Mapping Knowledge about Product Lifecycle Engineering for Ontology Construction via Object-Process Methodology

D. Dori[1], M. Shpitalni[2] (1)
[1]Faculty of Industrial Engineering and Management
[2]Laboratory for CAD and LCE, Faculty of Mechanical Engineering
Technion - Israel Institute of Technology
32000 Haifa, Israel

Abstract

Knowledge mapping is a first and mandatory step in ontology definition. This paper considers the lifecycle of products and systems, and discusses the creation of a knowledge-based ontology. With respect to the life cycle of products and systems, knowledge refers to the processes involved in their creation (design manufacturing and assembly), use and maintenance, and end of life (EOL). Hence, this knowledge should consider what a product is comprised of (its structure), how it operates (its dynamics), and how it interacts with the environment. A clearly defined and consistent mapping of knowledge regarding structure, operation and interaction is necessary to construct an effective and useful ontology. Yet, in order to obtain the required knowledge and to organize it in a consistent and useful form, an appropriate ontology must be used. An interactive, iterative and consistent method is needed to cope with this complex and circular problem. In this paper, the Object-Process Methodology (OPM) approach is considered, along with OPCAT [1], a tool for OPM-based knowledge modeling. OPM is a systems-modeling approach that represents knowledge about systems concurrently and bi-modally through graphics (a set of Object-Process Diagrams, OPDs) and text (Object-Process Language, OPL, a subset of English), yielding a single, unified and consistent view. In this paper we propose a foundational modeling and ontology construction approach for a generic product that incorporates hardware and software components. The ontology can serve as a basis for a knowledge model to cover the entire product lifecycle, from inception to EOL, and can be applied in the VRL-KCiP Network of Excellence.

Keywords

Knowledge management, Lifecycle ontology, Object-process methodology

1 INTRODUCTION

Advances in information technologies along with innovations in production systems have dramatically changed processes and engineering practices related to product development [2]. These changes, examined in terms of research challenges and opportunities by Kimura et al. [3], have enhanced global manufacturing capabilities to a point far exceeding the demand. As a consequence, competition among manufacturers has dramatically increased. At the same time, new issues, most notably environmental consciousness, are becoming increasingly critical, imposing additional constraints on new products.

The need for a shift in paradigm is described by Takata et al. [4], who claim that the scale of industrial activities has already exceeded its limit. Hence, the manufacturing paradigm must be changed, as suggested by Kimura [5]. To a large extent, this change in paradigm is based on the transition from selling of products to selling of services and on accommodating a holistic lifecycle approach [6], as suggested by Alting and Jorgensen [7] for sustainable industrial production and later reexamined by Hauschild et al. [8]. Recent studies [9] indicate that future products and systems will be increasingly knowledge-driven rather than energy-driven, underlining the urgent need to develop the proper tools and environment for extracting existing knowledge and generating new knowledge. This goal can be achieved only if a comprehensive ontology is defined to consider the entire lifecycle of products and systems.

Knowledge mapping is a first and mandatory step in the process of ontology definition. This paper considers the lifecycle of products and systems, and discusses the creation of a knowledge-based ontology.

While systems and products are becoming ever more complex, spanning a growing number of disciplines, development of an ontology that can serve as a basis for a rigorous methodology for lifecycle engineering is still in its evolutionary stages. In particular, the early lifecycle stages of most systems are still the least structured and least coherent. They suffer from ambiguity and lack a clear and direct translation of customer needs/wants into precise product specifications.

All phases comprising this lifecycle, including the end of life (EOL) phase, must be taken into account when mapping the knowledge and defining the ontology. Interaction between the phases must also be considered, bearing in mind that products are becoming more complex, and more software-based components are being integrated into products.

Following the early work by Tipnis [10] on product lifecycle economic models, several methods have been proposed for modeling and/or evaluating all or part of a lifecycle system. Takata et al. [4] proposed flexible means to represent technical information relevant to manufacturing facility management, especially with respect to maintenance. Krause & Kind [11] proposed a reference model to assure supply of appropriate information according to specific requirements pertaining to all lifecycle phases. In 2001, Brissaud & Tichkiewitch [12] proposed a comprehensive product model that can be used to globally optimize the phases comprising the product lifecycle. Concurrently, Westkämper [13] proposed a platform for integrating the management of assembly/disassembly and product lifecycle. Tools to support conceptual design, development, and assessment in terms of environmental cost and impact have been proposed by Park et al. [14], by Kaebernick et al. [15], who proposed a simplified lifecycle assessment for the early

design stages of products, and more recently, by Duflou et al. [16]. Proactive support tools for lifecycle engineering, as well as a simulation system for lifecycle process planning, have been proposed by Takata and Kimura [17]. Zhang et al. have proposed web-based applications, such as a web-based system for reverse manufacturing and product environmental impact assessment, which takes end-of-life dispositions into consideration [18].

In defining ontology, the knowledge must first be mapped. Knowledge, in turn, is about systems: their function, their structure, their dynamics, and their interaction with their environment, a collection of systems in itself.

System architecture is the underlying combination of structure (expressed in terms of objects and their attributes) and behavior (expressed in terms of processes and how they transform objects) that enables a system to attain its intended function. System architecting is therefore of paramount importance to successful design, implementation, and evolution of artificial complex systems in general and products in particular.

Having recognized the commonality among a large variety of systems, product lifecycle engineering combines a host of scientific, technological, ecological, and human aspects in a highly multidisciplinary fashion. Recently, the notion of Product Lifecycle Management (PLM) has started to make headway, not just in academic echelons but also in leading enterprises which have realized the necessity of a comprehensive approach to ensure that systems and products be managed by best industry practices backed by sound methodology. A clear, consistent and comprehensive mapping of knowledge about systems must account for system structure, dynamics, and interaction with the environment.

From cognitive science [19] we know that humans access and process knowledge via two main channels: the visual channel and the non-visual channel. The former relies on graphics and diagramming, while the latter is based on language, both spoken and written. Hence, an ideal knowledge mapping and representation system should be capable of presenting knowledge using graphics as well as at least a subset of natural language text.

In this paper we apply an OPM-based approach to modeling product lifecycle knowledge. In particular, we focus on knowledge organization and ontology creation for the Virtual Research Lab for a Knowledge Community in Production (VRL-KCiP) Network of Excellence (European Community 6th Framework Programme).

2 OBJECT-PROCESS METHODOLOGY

In complex products, structure and behavior, as well as hardware and software, are typically highly intertwined and interdependent. Motivated by this observation, Object-Process Methodology, OPM [20] has been developed as a holistic approach to the study and development of systems, integrating object-oriented and process-oriented paradigms into a single frame of reference. Structure and behavior, the two major aspects exhibited by any system, co-exist in the same OPM model without highlighting one at the expense of the other. Due to its structure-behavior integration, OPM provides a solid basis for modeling complex products. The OPM ontology consists of entities and links. Entities include things and states. Things are stateful objects and processes, both of which can be physical or informatical. Objects exist, possibly at some state, while processes transform objects by generating or consuming them, or changing their state.

Links can be structural or procedural. Structural links express static, time-independent relations between pairs of entities. The four fundamental structural relations are aggregation-participation, generalization-specialization, exhibition-characterization, and classification-instantiation. Procedural links connect entities (objects, processes, and states) to describe the behavior of a system.

System behavior can be manifested in three major ways: (1) processes can transform (generate, consume, or change the state) of objects; (2) objects can enable processes without being transformed by them; and (3) objects can trigger events invoking processes if some conditions are met. Accordingly, a procedural link can be a transformation link, an enabling link, or an event link.

A transformation link expresses object transformation, i.e., object consumption, generation, or state change. An enabling link expresses the need for an object to be present, possibly at a certain state, in order for the enabled process to occur, but the enabled process does not transform the enabling object.

An event link connects a triggering entity (object, process, or state) with a process that it invokes. The event types supported by OPM include state entrance, state change, state timeout, process termination, process timeout, reaction timeout, and external events. External events can be clock events or events triggered by environmental entities, for example a user or an external device.

2.1 The Bimodal Graphic-Text OPM Representation

Two semantically equivalent modalities, one graphic and the other textual, jointly express the same OPM model. The graphical, visual OPM formalism consists of a set of interrelated Object-Process Diagrams (OPDs), showing portions of the system at various levels of detail. Each OPM element is denoted in an OPD by a symbol. The OPD syntax specifies correct and consistent ways by which entities can be connected via structural and procedural links, each having its specific, unambiguous semantics. OPM assigns special graphical symbols to a selected set of relations, as in Unified Modeling Language (UML) class diagrams, but for a larger set of relations [21].

OPCAT [1] is a Java-based software environment that supports OPM system modeling and evolution. As shown in the toolbox at the bottom of OPCAT's GUI in Figure 1, a triangular graphical symbol along the line connecting two items (objects or processes) is assigned to each of the four fundamental structural relations mentioned above, symbolically represented as follows: black triangle for aggregation, white triangle, as in UML, for generalization, black on white triangle for characterization, and black circle in white triangle for instantiation. Like associations in UML class diagrams, other structural relations become textual tags (labels) recorded along the arrow connecting the two entities, such that concatenating the source entity with the tag and then with the destination entity yields a meaningful sentence.

Object-Process Language (OPL) is the textual counterpart modality of the graphical OPD set. OPL is a dual-purpose language, oriented towards humans as well as machines. Catering to human needs, OPL is designed as a subset of English that serves domain experts and system architects jointly engaged in analyzing and designing a system or product. Every OPD construct is expressed by a semantically equivalent OPL sentence or phrase. This dual representation of OPM increases the processing capability of humans, according to the cognitive theory of multimodal learning proposed by Mayer [19]. By catering to the modality principle of this cognitive theory, OPM enables modeling systems to operate both graphically and textually.

2.2 Top-Level View of Product Lifecycle Engineering

Figure 1 presents a snapshot of OPCAT's graphic user interface, showing a top-level view (System Diagram, or SD) of the Product Lifecycle Engineering system considered here. OPCAT translates each OPD construct into its equivalent OPL sentence, yielding an OPL paragraph that is a collection of OPL sentences specifying the knowledge represented graphically in the OPD. Conversely, typing an OPL sentence complements the OPD, such that at any point the graphic and textual representations are completely equivalent and can be reconstructed one from the other. With OPCAT, users can model complex systems and products at all levels of granularity, and express and query knowledge related to these models. In the OPCAT GUI in Figure 1, the hierarchy of diagrams and things (i.e., objects and processes) appears on the left, the graphic (OPD) window at the top right, the text (OPL) window at the bottom right, and the palette with the various OPM entities and links at the bottom. Some OPL sentences, all generated automatically by OPCAT, are shown in the OPL window.

Objects are denoted as rectangles, processes by ellipses, and object states by rounded rectangles inside the object. Most of the objects depicted are physical, as denoted by the shading of their representative rectangles. These include Product and Manufacturer, also expressed in the sentences "Product is physical." and "Manufacturer is physical." The entire OPL paragraph is as follows:

Product is physical.
Product can be **approved**, **distributed**, **used**, **pre-tested**, **rejected**, **stored**, **sold**, or **retired**.
    pre-tested is initial.
    retired is final.
Product is **made** of **Raw Material**.
Product **benefits** **User**.
Manufacturer is physical.
Manufacturer makes & supports **Product**.
Environment is environmental and physical.
Environment consists of **User**, **Market Demand**, **Raw Material**, **Technology**, and **Competition**.
    User is environmental and physical.
    Market Demand is environmental.
    Raw Material is environmental and physical.
    Technology is environmental.
    Competition is environmental.
Product Lifecycle Engineering is physical.
Product Lifecycle Engineering requires **Environment**.
Product Lifecycle Engineering affects **Product**, **User**, and **Manufacturer**.

These sentences show how knowledge that combines structure and behavior can be represented both by intuitive (yet formal) graphics and by human intelligible text, OPL, a subset of English. Users not familiar with OPM graphic notation can verify their specifications by inspecting the OPL sentences generated or edited with each graphic input. For example, the OPL sentence "**Product Lifecycle Engineering** affects **Product**, **User**, and **Manufacturer**." is equivalent to the relevant part of the OPD showing the three effect links (denoted by bidirectional arrows) connecting the objects **Product**, **User**, and **Manufacturer** to the process **Product Lifecycle Engineering**, giving rise to the OPL reserved word "affects". The effect link connects a process (**Product Lifecycle Engineering**) to an object, or in our case, each one of the three affected objects, **Product**, **User**, and **Manufacturer**. States are situations in which an object can exist. The sentence "**Product** can be **approved**, **distributed**, **used**, **pre-tested**, **rejected**, **stored**, **sold**, or **retired**." is a state enumeration sentence listing all possible states of the object **Product** throughout its lifecycle. States can be initial or final, as expressed in the sentences "**pre-tested** is initial." and "**retired** is final." The corresponding graphic symbol is a thick frame for the initial state and a double frame for the final state.

A major problem with most graphic modeling approaches is their scalability. As system complexity increases, the graphic model becomes loaded with shapes and cluttered with links crossing each other in all directions. According to the cognitive principle of limited channel capacity [19], there is an upper limit on the amount of detail humans can process before becoming overwhelmed. OPM addresses this principle and implements it in OPCAT using three abstraction/refinement mechanisms: (1) *unfolding/folding*, for refining/abstracting the structural hierarchy of a thing and applied by default to objects; (2) *in-zooming/out-zooming*, to expose/hide an item's inner details within its frame, applied primarily to processes; and (3) *state expressing/suppressing*, to expose/hide an object's states. These mechanisms enable complexity management by enabling the creation of interrelated OPDs (along with their corresponding OPL paragraphs) that are limited in size, thereby avoiding information overload and facilitating convenient human processing. Flexible combinations of these three mechanisms enable OPM to specify a system to any desired level of detail without loss of legibility or clarity of the resulting specification. The complete OPM system specification is expressed graphically by the resulting set of consistent, interrelated OPDs, and textually by the corresponding OPL script, the union of information expressed in the OPL paragraphs. Like OPD, each OPL paragraph is a collection of sentences spanning no more than a single page, so that humans can comfortably read and digest the knowledge it expresses.

3 OPM AND PRODUCT LIFECYCLE ENGINEERING

The Product Lifecycle Engineering process is in-zoomed in Figure 2 to show subprocesses, from Design to End of Life, executed in a top-to-bottom order.

3.1 The Design Process

The Design process is further in-zoomed in Figure 3 to show subprocesses ranging from Requirement Engineering through Conceptual and Detailed Design to Transfer to Production.

3.2 Simulation through Animation

OPCAT is capable of simulating a system through vivid animation. Figure 4 shows the animated Design process, with Detailed Design in action, generating the Hardware, Software, and EOL models. After Design comes Manufacturing. One of the outcomes of Detailed Design is the Software Model, which, together with Hardware Models and the EOL Model, constitutes the Product Model. The Software Model is the instrument for Software Module Developing, embedded within the Making process. In turn, the Making process is embedded within Manufacturing. Figure 5 shows the details of the Software Module Developing process, revealing Analysis, Design, and Implementation as its three major subprocesses. Software, as part of the entire product, undergoes a "mini lifecycle," similar to the grand product lifecycle, albeit on a smaller scale. When software constitutes a significant and especially a major part of a product, the situation might be reversed. Hardware design and manufacturing, if present, would then be embedded as part of the encompassing software product lifecycle model. Here, however, we consider a case in which software still plays a less significant role in the overall product architecture.

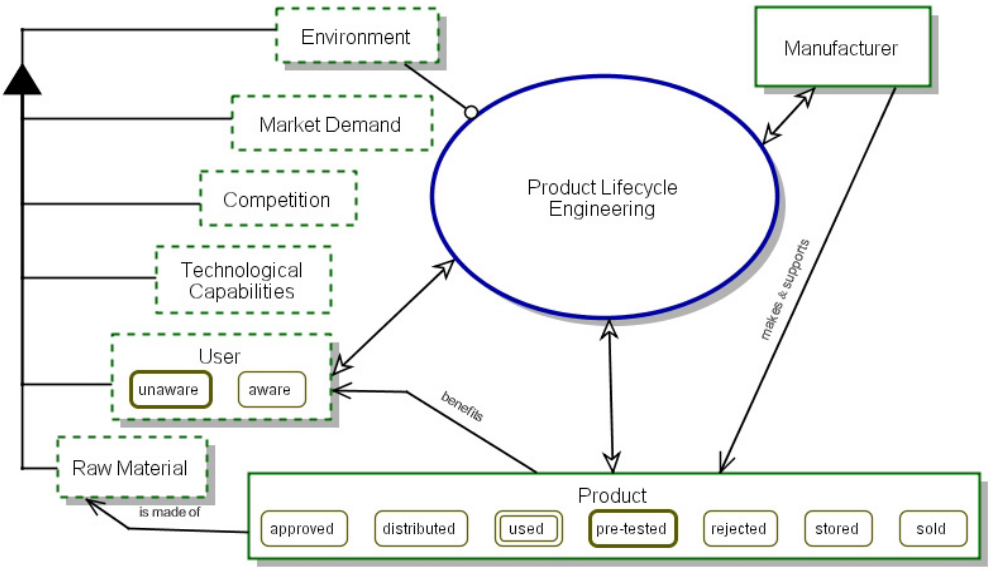

Figure 1: Top-level view (System Diagram, SD) of the Product Lifecycle Engineering system, showing the OPD window (top), part of the corresponding OPL window (bottom), and the OPD tree (left).

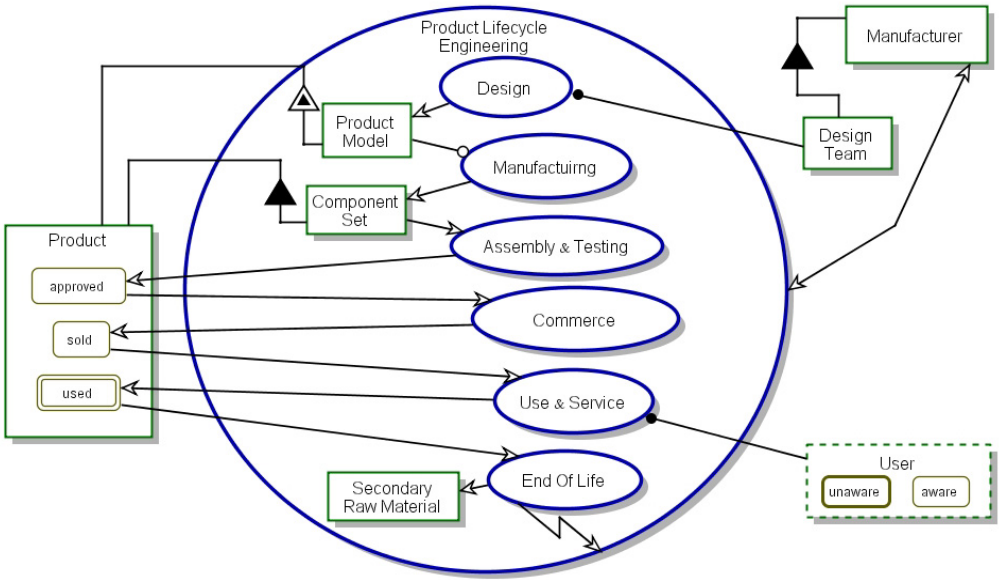

Figure 2: The Product Lifecycle Engineering process is in-zoomed to show subprocesses from Design to End Of Life, executed in a top-to-bottom order.

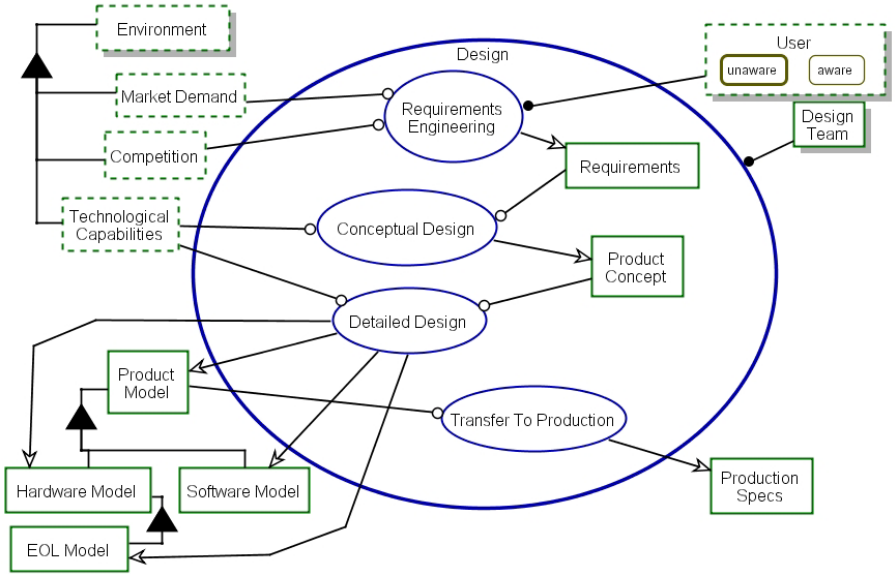

Figure 3: Design process is in-zoomed to show subprocesses from Requirement Engineering to Transfer to Production.

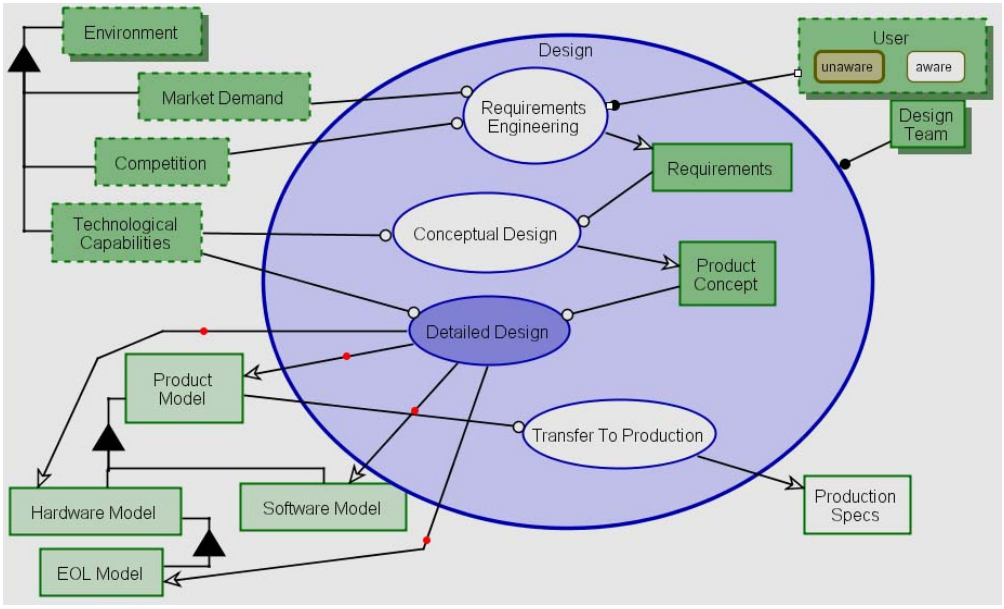

Figure 4: Animated Design process, showing Detailed Design in action, as it generates Hardware, Software, and EOL models.

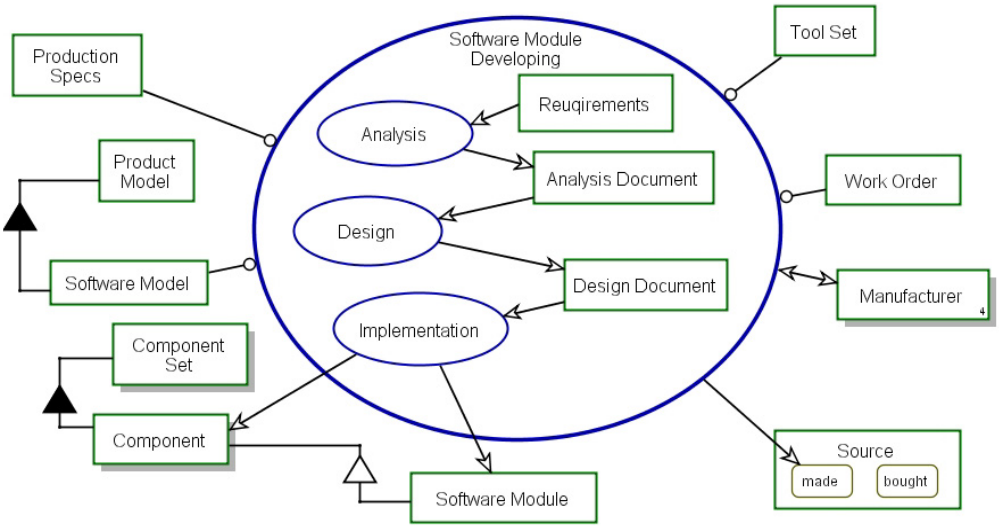

Figure 5: Software Module Developing process, showing Analysis, Design, and Implementation as three major subprocesses.

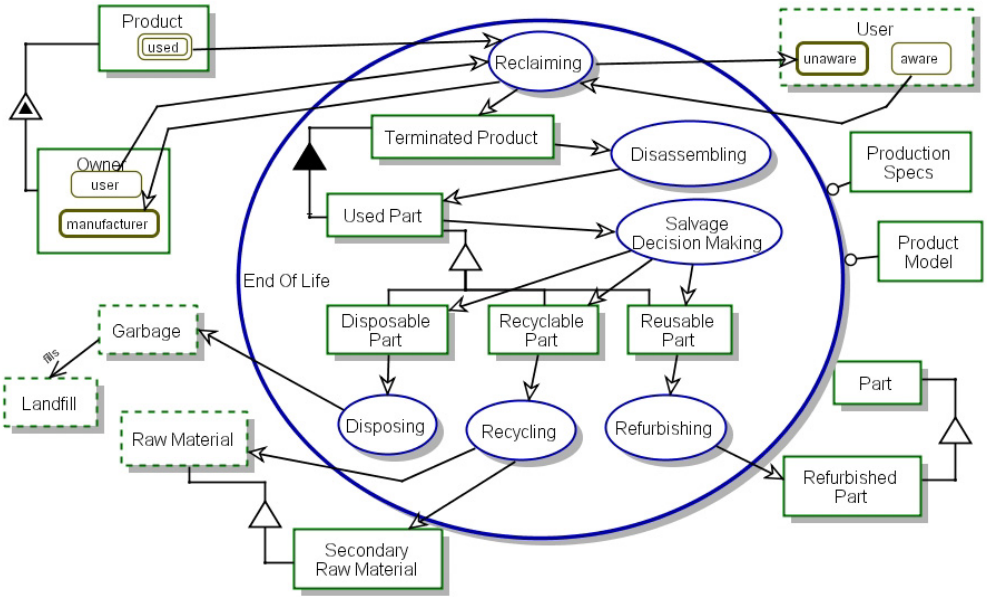

Figure 6: End of Life process, with subprocesses: Reclaiming, Salvage Decision Making, Disposing, Recycling, Refurbishing.

3.3 Commerce, Using, and End of Life

Commerce, the next process in the product lifecycle, comprises **Storing**, **Marketing**, **Distributing**, and **Selling**. The following process, **Use & Service**, zooms into **Using**, **Customer Relationship Management**, **Maintenance**, and **Retiring**. **Using** is the ultimate objective of the entire product lifecycle, for all processes activated before and after it are aimed at making the **Using** process longer, more productive, and more appealing to the User. Using changes the **Benefit** which is a **User** attribute, from potential to materialized.

Retiring is the process that occurs when **Product** is no longer usable, when using it causes more trouble than the benefit from the service it is supposed to provide, or when newer, more attractive and economic alternatives are available. **Retiring** changes the state of **Product** from **used** to **retired**, marking its entry to the **End of Life** process. As described in Figure 6, the **End of Life** process includes the subprocesses of **Reclaiming**, **Salvage Decision Making**, **Disposing**, **Recycling**, and **Refurbishing**.

4 SUMMARY AND FUTURE WORK

Using Object-Process Methodology and OPCAT, the OPM-supporting software tool, we have laid the foundations for ontology of a generic lifecycle product that spans the entire spectrum of the product lifecycle, from **Design** to **End Of Life**, and includes hardware and software components. The ontology specifies the hierarchy of processes and the objects they transform. This ontology can serve as a basis for specialized product lines and their characteristic lifecycle processes. It can also serve as a basis for knowledge mining via technologies such as inexact graph matching used as Web services. As future work, we will examine a number of representative cases typical of manufacturing processes matching the expertise of the research groups in the VRL-KCiP NoE. We will integrate these cases as specializations of the generic product lifecycle model.

5 ACKNOWLEDGMENTS

This research has been supported in part by the Minerva Schlesinger Laboratory for Automated Assembly and the EU VRL-KCiP NoE.

6 REFERENCES

[1] Dori, D., Reinhartz-Berger, I., Sturm, A., 2003, OPCAT – A Bimodal CASE Tool for Object-Process Based System Development, Proc. IEEE/ACM 5th International Conference on Enterprise Information Systems (ICEIS 2003), Angers, France, 286-291. (www.ObjectProcess.org)

[2] Bar Cohen, A., 1995, Mechanical Engineering in the Information Age, Mechanical Engineering, 117/12:66-70.

[3] Kimura, F., Lipson, H., Shpitalni, M., 1998, Engineering Environments in the Information Age: Research Challenges and Opportunities, Annals of the CIRP, 47/2:87-90.

[4] Takata, S., Kimura, F., van Houten, F.J.A.M., Westkämper, E., Shpitalni, M., Ceglarek, D., Lee, J., 2004, Maintenance: Changing Role in Life Cycle Management, Annals of the CIRP, 53/2.

[5] Kimura, F., Suzuki, H., 1995, Product Life Cycle Modelling for Inverse Manufacturing, Life Cycle Modelling for Innovative Products and Processes, Chapman & Hall, 80-89.

[6] Shpitalni, M., 2004, Impact of Life Cycle Approach and Selling of Services on Product Design, Archives of Mechanical Technology and Automation (Special Issue), 24/2:251-261.

[7] Alting, L., Jorgensen, J., 1993, The life cycle concept as a basis or sustainable industrial production, Annals of the CIRP, 42/1:163.

[8] Hauschild, M., Wenzel, H., Alting, L., 1999, Life cycle design - a route to the sustainable industrial culture?, Annals of the CIRP, 48/1:393-396.

[9] Manufuture: A Vision for 2020, Manufuture 2004, http://europa.eu.int/comm/research/industrial_technologies/pdf/manufuture_vision_en.pdf.

[10] Tipnis, V.A., 1991, Product life cycle economic models, Annals of the CIRP, 40/1:463-466.

[11] Krause, F.-L., Kind, Chr., 1995, Potentials of information technology for life cycle oriented product and process development, in Proceedings of the IFIP WG5.3 international conference on life cycle modeling for innovative products and processes, H. Jansen and F.-L. Krause, Eds., Chapman & Hall: Berlin, 14-27.

[12] Brissaud, D., Tichkiewitch, S., 2001, Product Models for Life-Cycle, Annals of the CIRP, 50/1:105-108.

[13] Westkämper, E., 2002, Platform for the Integration of Assembly, Disassembly and Life Cycle Management, Annals of the CIRP, 51/1:33-36.

[14] Park, J.-H., Seo, K.-K., Wallace, D., Lee, K.-I., 2002, Approximate Product Life Cycle Costing Method for the Conceptual Product Design, Annals of the CIRP, 51/1:421-424.

[15] Kaebernick, H., Sun, M., Kara, S., 2003, Simplified Lifecycle Assessment for the Early Design Stages of Industrial Products, Annals of the CIRP, 52/1:25-28.

[16] Duflou, J., Dewulf, W., Sas, P., Vanherck, P., 2003, Pro-active Life Cycle Engineering Support Tools, Annals of the CIRP, 52/1:29-32.

[17] Takata, S., Kimura, T., 2003, Life Cycle Simulation System for Life Cycle Process Planning, Annals of the CIRP, 52/1:37-30.

[18] Zhang, H.C., Li, J., Shrivastava, P., Whitley, A., Merchant, M.E., 2004, Web-Based System for Reverse Manufacturing and Product Environmental Impact Assessment Considering End-of-Life Dispositions, Annals of the CIRP, 53/1:5-8.

[19] Mayer, R. E., 2001, Multimedia Learning, Cambridge University Press.

[20] Dori, D., 2002, Object-Process Methodology-A Holistic Systems Paradigm, Springer Verlag, Berlin, Heidelberg, New York, www.ObjectProcess.org.

[21] Dori, D., 2002, Why Significant Change in UML is Unlikely, Communications of the ACM, 82-85.