Ontology-based Support for Variability Management in Product and Service Families

Kannan Mohan and Balasubramaniam Ramesh
kmohan@cis.gsu.edu, bramesh@gsu.edu
Department of Computer Information Systems
Georgia State University.

Abstract

Product/service family engineering, which encourages the development of a common product platform, plays a key role in facilitating large-scale and planned reuse in the development of customized products. In product family designs, managing points of variability is critical to achieve product variety. Variation points are points at which one among the several possible variations of a feature of a system can be selected to achieve different configurations of a product. We are developing an ontology that catalogues the different concepts associated with variability. This ontology is used to define the elements characterizing the knowledge elements necessary for managing variability in product/service families. We have also developed a knowledge management system integrated with an ontology development tool to facilitate knowledge capture and retrieval for variability management.

1. Introduction

Customers prefer products and services that are specifically tailored to suit their needs, but are no more expensive than generic products [1]. Product and service development firms are increasingly using family-based development approach to satisfy these conflicting requirements of distinctiveness of the products/services and low costs. Mass customization attempts to addresses this issue by facilitating provision of tremendous variety and individual customization at prices comparable to standard products and services [2]. Techniques used in mass customization can be used to achieve service variety by exploiting the commonality among the variety of services/products demanded by the customers [2]. Distinctiveness in products (or services) can be achieved by designing a set of products (or services) that have a lot of common functionalities or features, but have sufficient number of points where variations can be introduced. Such a set of products, called as a product family, share a common, managed set of features that satisfy the specific needs of a particular market segment and are developed from a common set of core assets in a prescribed way [3]. Proliferation of variety becomes possible by delaying design decisions, thereby introducing variability in the family of developed products and services. A service platform is defined as the common set of design variables around which a family of services can be developed [4]. A platform is built using core assets that are used to build products that are part of the service family. Prior research suggests that building a platform-based architecture and adding customer specific attributes on top of the platform leads to reduction in development costs [5]. Platform based development is increasingly being recognized as a strategy for maximizing innovation and dramatically increasing the value of a firm's product and service by using it along with complements [6].

E-Services are defined as internet-based applications that fulfill service needs by seamlessly bringing together distributed, specialized resources to enable complex transactions [7]. Services that are delivered electronically, typically through the Internet, are referred to as e-services. We define a service family as a set of services that share certain common aspects and have predicted variability. Services are clustered as a family based on their commonality, as it is easier to analyze, design and manage the family as a related set of elements rather than concentrating on each member of the family separately [8]. A family-based approach focuses on managing both the commonality and differences among family members. Dijkstra and

Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)
0-7695-1874-5/03 $17.00 © 2002 IEEE

Parnas, [9, 10] pioneers in structured programming and program families, suggest that each new alternative solution for a related problem be considered a member of a family. Coplien et al. [11] argue that commonality and variability analyses help software engineers achieve a systematic way of thinking about and identifying the family of services.

Flexible design and development of service families largely depend on reusability of assets that are produced during domain engineering. Reusability can be facilitated by a common service platform architecture. Service variety is achieved by plugging in different components that satisfy customer-specific requirements to the common core. This approach involves determining relationships among various components and modules [4]. Further, this requires effectively modeling the various ways in which different configurations of the services vary from each other and the development of mechanisms to implement these variabilities.

Some of the common issues associated with variability management in product families are as follows [12]:

- Dependency management
  - Assessing the impact of changes in some artifacts on others generated during the development life cycle of software product lines
  - Knowledge about dependencies among artifacts is key to identifying the appropriate parts of the architecture that have an impact on the development of specific variants. Bosch et al. [12] reiterate that managing such dependency knowledge related to variations is a complex task and needs extensive tool support.
- Evolution of variability
  - Adding new variation points
  - Changing attributes of existing variation points
  - Adding new variants
- Modeling and implementing variability
  - Selection of a wrong phase in the development life cycle to introduce and model variability
  - Selection of an inappropriate mechanism to implement variations
  - Scattering of variability over various parts of the system
- Interactions among features
  - Gurp et al. [13] refer to variability in terms of variant features. They stress the need to model interactions among these features.
- Choices for binding variations
  - A variation point can be bound to different product variants depending on the choice made according to the customer requirements. Bachmann and Bass [14] refer to the existence of such choices as alternatives for binding variation points to specific variants and the delay in selecting one necessitating the capture of all the alternatives.
  - Capture of rationale related to the various alternatives aids the development team by providing a set of solutions to use.

The objective of this research is to address these issues through the complementary use of an ontology for variability and a knowledge management system that captures traceability knowledge associated with variability in product and service families. In this paper, we discuss the use of the ontology that we are developing as a means to classify and conceptualize the knowledge captured about variability in a family of software systems. As part of this ontology, we are also capturing various variability modeling mechanisms thereby aiming to provide support for mechanism selection. We illustrate, through examples from a case study, how we use this ontology to facilitate knowledge reuse.

In section 2, we define and describe variability. We describe the development and use of an ontology for variability in section 3. In section 5 we provide a discussion on related literature. Then we enumerate the contributions of our research and provide directions for future research.

2. Managing Variability in Product/Service Families

Variability refers to the points where the behavior of the system can be changed [15]. It is defined using differentiation index, which measures where differentiation occurs within the process flow [16]. Coplien et al. [11] describe variability as an assumption that is true for some elements in a set of objects, or an attribute with different values for at least two different elements from the set of objects. Variations are triggered by customer-specific requirements that could indicate changes in the environment or lead to algorithmic changes in the domain [17]. Bachmann et al. [14] argue that the existence of a collection of alternatives that provide a list of potential solutions to the development team is a cause for variability. Feature-based recognition of variability, proposed by Lee et al [17], suggests that variability should be analyzed in terms of product/service features. Since it is

very difficult to elicit all variable requirements completely from the customer, the platform should be designed in such a manner so as to accommodate changes necessitated by evolving requirements.

Following a process that lends itself to the anticipation and identification of all possible family members during the early phases of the development life cycle would be an ideal method for product/service family development [8]. Bachmann and Bass [14] categorize and describe variability at the architectural level, and prescribe architectural solutions for the various types of variations. We argue that variability is multi-faceted, and that it is important to recognize the importance of conceptualizing the various knowledge elements associated with variability and use appropriate mechanisms for modeling them. A primary thesis of our work is that the development of an ontology of variability and the mechanisms to implement them, will facilitate knowledge reuse in service family development. An ontology should encompass the various types of variability, its attributes, different mechanisms that can be used to model variability and other concepts related to variability. We further propose the development of a knowledge management system to manage knowledge about the sources and uses of variability to facilitate reuse of variability mechanisms across product families.

3. Ontology for Variability

An ontology is a set of concepts or terms and their relationships that describe some area of knowledge or build a representation of it [18]. Ontologies can then be built by identifying the following components: 1) basic terms and relationships between terms; 2) rules to combine terms; and 3) definitions of such terms and relationships [19]. It would also include simple rules of inference and logic for a particular domain. Ontology development aims at capturing domain knowledge in a generic way and providing a commonly agreed understanding of a domain that may be reused and shared across applications and groups [20]. We are developing an ontology to represent various aspects of variability in product and service families. We use Methontology [21] as methodology for our ontology development due to the suitability of the intermediate representations prescribed by this method to model variability. The usability and simplicity of Methontology also make it suitable for use by designers of service families. The scope of our ontology was defined after examining various scenarios that arise in a product/service family development organization. Several interviews with domain experts involved in the development of a family of Warehouse management systems helped in the initial identification of concepts, their properties and their relationships with other concepts. Structured interviews were used to evaluate the conceptual model and to verify the accuracy of domain definitions. Concepts from the resulting domain dictionary were used to develop a domain independent ontology for variability. Intermediate representations suggested by Methontology were developed based on an extensive survey of literature on managing variability in product and service families. The ontology that we are developing varies from high domain dependence to domain independence. We have also mapped concepts from the generic ontology to those that pertain to the domain specific ontology. In order to capture and store this ontology, we have developed a tool based on Microsoft Access. This tool facilitates management of intermediate representations of ontologies. Figure 3 shows a screen shot of our tool illustrating the conceptual hierarchy of our ontology, an intermediate representation suggested by Methontology.

4. Integrating the Ontology with a Knowledge Management-Centric Approach

We view product family development as a knowledge intensive process. This involves managing variations among different members of the product family by identifying common and variable aspects in the domain under consideration. The issues identified in section 1 emphasize the knowledge intensiveness of the product line engineering process. We suggest that a flexible knowledge management system that facilitates knowledge reuse will lead to effective product family development. We suggest the capture of various knowledge elements related to variability. This knowledge will be useful in understanding the source of variability and identifying appropriate mechanisms to implement it. Further, this knowledge can be used to implement variability in other related product families. The first step in this approach is the definition of the knowledge elements that must be captured to represent the context in which variability is introduced and implemented in a product family. To answer this question, we turn to the ontology for variability developed from a case study in product and service family development. This ontology identifies variability-related concepts at varied levels of granularity that are to represent the contextual knowledge about variability in the knowledge management system. Figure 1 identifies the elements of our approach to integrating the ontology for variability with our knowledge management system. The oval elements in the figure represent processes and the two-sided boxes represent knowledge fragments.

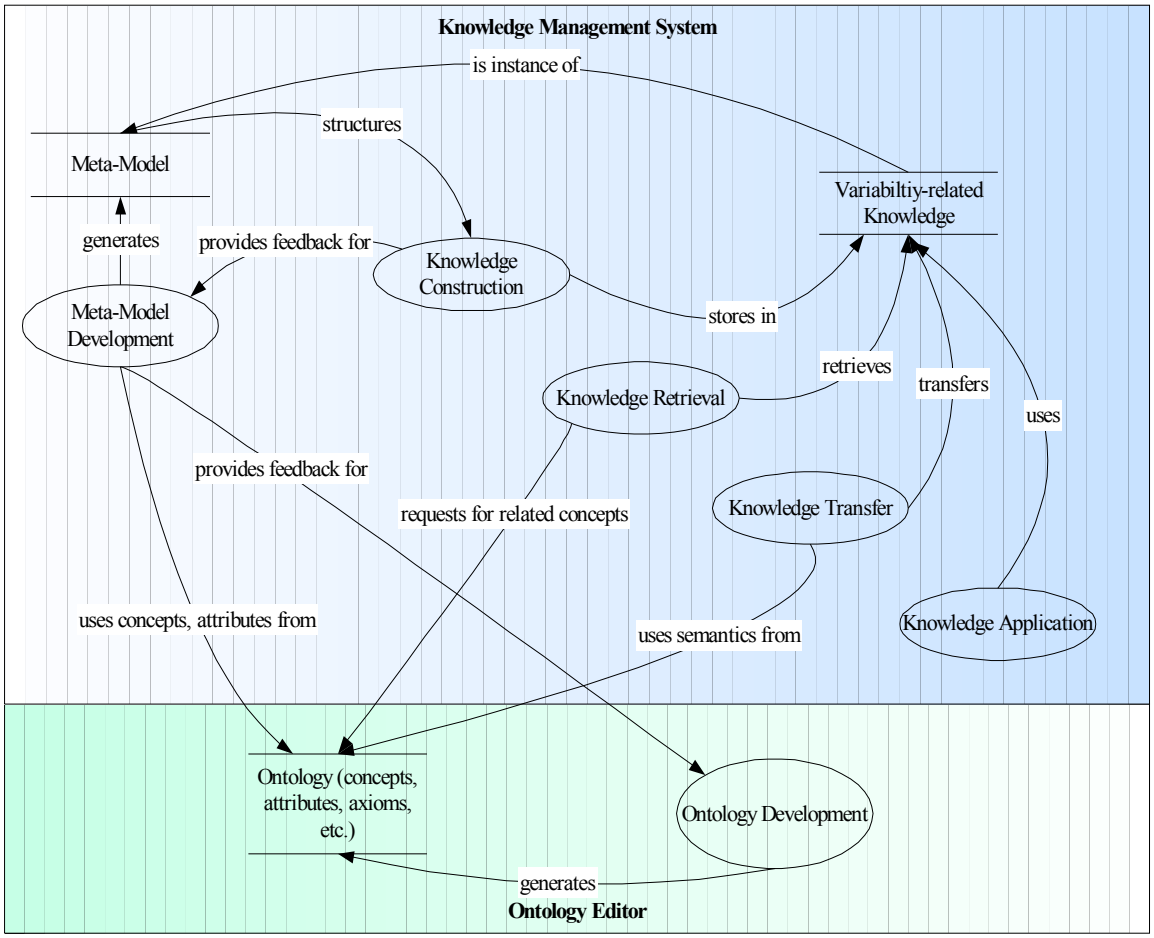

Figure 1: Integrating Ontology and Knowledge Management for variability management

Figure 1 shows how ontology development guides the development of a meta-model specifying concepts related to variability, and the various knowledge management processes. Our knowledge management system is capable of handling the four sets of knowledge management processes suggested by Alavi and Leidner [22]: Knowledge Construction, Storage and Retrieval, Transfer and Application. Figure 1 shows how the ontology complements these knowledge processes, consistent with the approach proposed by Staab et al. [23] for ontology-based knowledge management. The meta-model development process uses various concepts from the ontology, at different levels of granularity. We describe the functionalities of our KMS to facilitate knowledge construction and knowledge retrieval using the ontology.

4.1. Knowledge Construction

The meta-model developed is used as a schema that structures the knowledge construction process. The designer can use our knowledge management system to define the meta-model or the schema for knowledge capture. In this process, the designer can choose concepts from the ontology to be entities in the meta-model. Also, elements in the ontology can be used to define attributes of knowledge elements in the meta-model. After the meta-model has been defined according to the project specific needs, the designer can use the system to construct knowledge elements by instantiating the meta-model elements. This constructed knowledge is stored in a knowledge repository.

Figure 2 shows a snapshot of our knowledge management system illustrating the knowledge construction process. It shows the available schema elements as tool bar buttons. Here, 'stakeholder' is a knowledge element that has been already added to the meta-model. To add further elements to the meta-model, the designer opens the 'Set Node and relations' user interface (shown in Figure 2). From this interface, the designer could invoke an interface to the ontology (shown in Figure 3) using the 'Open Ontology Browser' button.

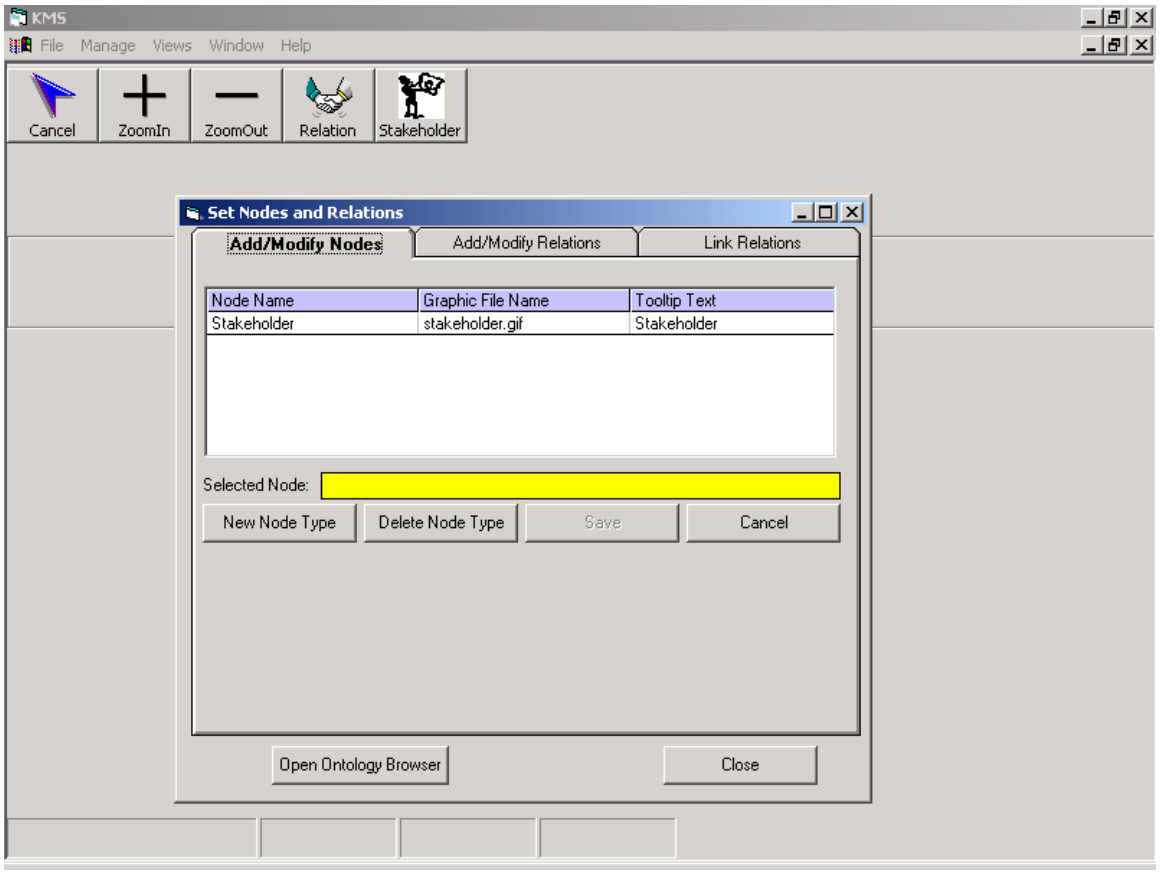

Figure 2: Meta-Model Definition in KMS

Concepts from this ontology can be selected and included in the meta-model. Also, the attributes of these concepts can be imported from the ontology to the knowledge schema.

4.2. Knowledge Retrieval

Retrieval of stored knowledge is facilitated by a sophisticated querying mechanism that exploits the associations established in the ontology. Consider the situation when a designer is interested in identifying mechanisms used in past projects or in other members of a product family that implement a specific type of variability. The knowledge base is first queried to see if there is a direct match between the user's query. In the absence of a direct match, the ontology is used to construct other queries to represent semantically similar situations. Semantic distances established among various elements in the ontology can be used to identify similar concepts to construct such queries. Even when it is not possible to construct a semantically equivalent situation, it is possible to identify 'similar' situations using such an approach. Then, relevant mechanisms used to address the variability problem can be retrieved from the knowledge base. This process helps maximize knowledge reuse. Examples of reusable knowledge fragments include mechanisms used to implement variability, appropriate phases in which specific types of variability has to be modeled or the rationale related to crucial variability-related design decisions made.

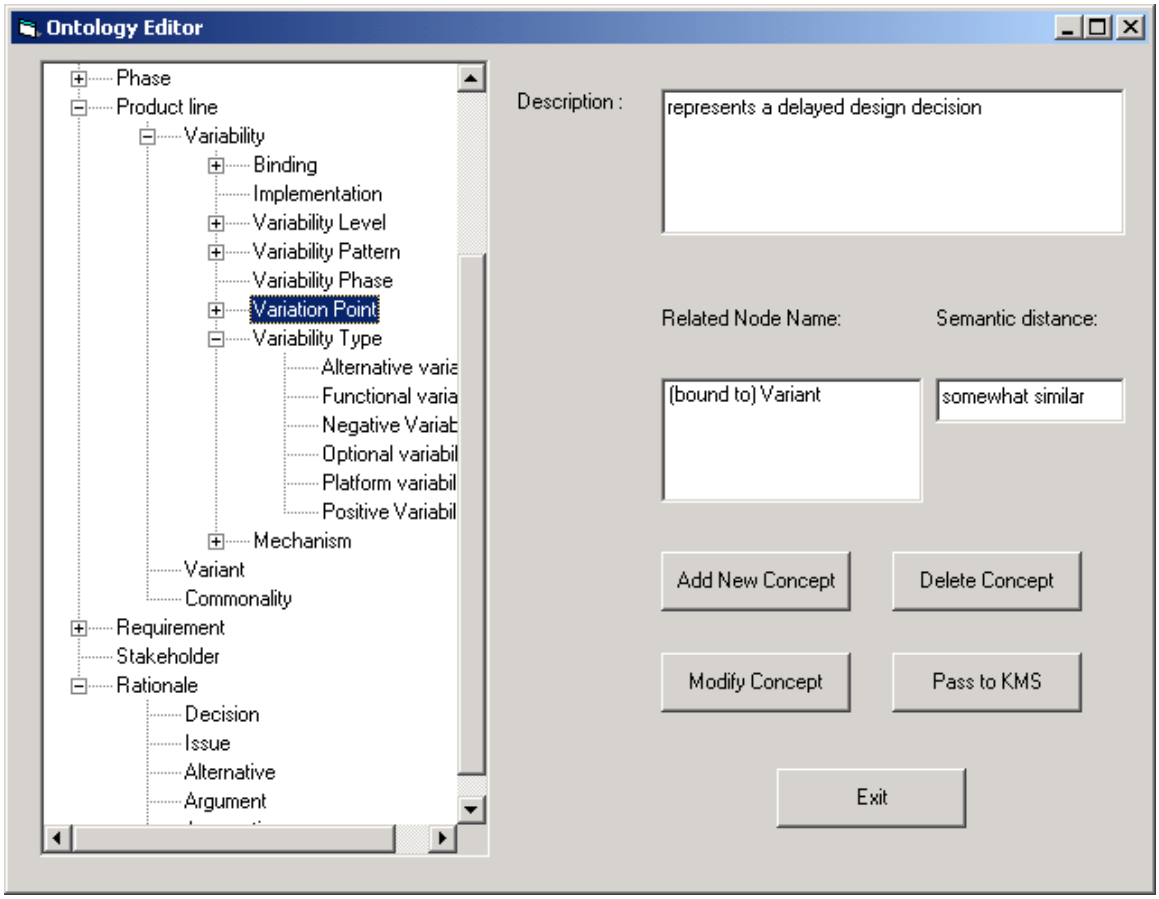

Figure 3: Ontology Editor showing fragments of Ontology

Consider the scenario when a user of our KMS is interested in identifying concepts related to a specific variability point. Our KMS communicates with the ontology repository (illustrated in Figure 3) to identify associated concepts that have not been captured in a given model. After identifying such related concepts, the KMS can identify various instances of these concepts and prompt to user to check for similar concepts that might be of use to the current scenario. Such identification is facilitated by computing the conceptual distance between the similar knowledge elements. The various attribute values of concepts play a key role in facilitating the calculation of conceptual distances. The retrieval of related concept instances will lead to knowledge reuse as illustrated below with an example from the case study.

For example, consider a product family for warehouse management. A customer can request a variety of operations to be performed on received goods: : (a) They can be sent to a quality audit as soon as they arrive; (b) They can be sent to the inventory to be stored; or (c) They can be shipped directly to the customer. This set of operations varies depending on the customer-specific requirements. A customer may also request functionality to support additional operations at any time. The variation required here is the set of operations that can be performed on a particular object, in this case, the received good. When the designer attempts to design a solution to this variability need, s/he could begin creating nodes in the KMS to document the solution. As soon as the designer instantiates some concepts like variability, variation point and variability type, s/he requests the KMS to retrieve related concepts. The KMS will communicate with the ontology repository to identify related concepts and then search its knowledge repository to identify instances of these related concepts. Such identification is facilitated with the computation of conceptual distance measures. Assuming that this type of variability had been handled before and had been documented in the KMS, the documented solution will have instantiated concepts related to variability modeling mechanisms. These instances, for example, may refer the designer to the use of visitor pattern to address this variation. The designer can choose to reuse the past solution along with the desired parts of the past documentation tailored to match the current scenario.

Knowledge transfer implies the transfer of knowledge from where it is stored to where it is needed. Inconsistencies in semantic understanding between the knowledge constructors and the users hampers effective knowledge transfer. By providing a semantic mapping, the ontology helps alleviate this problem.

5. Related research

Prior research has addressed the various key issues related to variability identification and modeling. Feature-Oriented Domain Analysis focuses on the systematic discovery and exploitation of commonality and variability in related software systems [24]. FODA is primarily done to identify distinct features in the domain. Capabilities in a domain are modeled as features in FODA [25]. These features depict common as well as different aspects of a domain. FODA uses the techniques of generalization/specialization, parameterization and aggregation to model variations. In FODA, variations are modeled at a much higher level than what is discussed in this research. Our approach differs in the granularity at which variability is addressed. FODA is aimed at analyzing and modeling the various types of features required in the system. Our approach can complement FODA in that the architectural modeling phase can use the techniques discussed in this paper. Koala is a component model that suggests the separation of component and configuration development. It is similar to COM (Component Object Model) [26] in that the Koala components have interfaces like their COM counterparts. This model is to introduce component-orientation in a restrictive domain. Variations are modeled using Koala by configuring the various Koala components. The configuration information is independent of the components that are part of the configuration and also, the components are designed in such a manner that they make no assumptions about the configurations. Koala also addresses problems at a different level of design when compared to our research. In broad terms, it is a product family architecture representation technique that can be used to capture variation points while the techniques discussed in this paper are more implementation-oriented. In fact, our solution strategy can be used within the design of Koala components to implement internal variations.

Mae [27] is similar to Koala in its objectives. It is again used to provide an architectural representation [28]. It differs from Koala in a few aspects. Instead of a switch design pattern, the Mae uses a new type of variant component. Mae is also tightly integrated with a configuration management system so as to document architecture evolution.

A primary difference between these studies and ours is the use of an ontology for identifying and matching variability needs and types with appropriate mechanisms that can be used to model and implement these variations.

Prior research in the area of ontologies has focused on multiple aspects of developing and using them. Ontology engineering has been a challenging area and has gained extensive focus [29]. Several ontology editors, ontology representation languages and ontology development methodologies have been proposed in the past [30]. Holsapple and Joshi [31] suggest a collaborative approach to ontology development.

Ontologies have been applied for widely diverse purposes. Storey et al. [32] discuss the use of common sense knowledge in automating the process of database design. They use ontologies to enable their common sense business reasoner to semantically understand the various terms and relationships. They also use a distance function that provides a measure of how close the functions are, as a key to knowledge reconciliation. Ontologies have been used in many projects, viz., Cyc [33], shared ontologies for knowledge reuse [34, 35], taxonomic reasoning for conceptual database design [36] and heterogeneous databases [37]. Goldstein and Storey [38] discuss the adaptation and implementation of an ontology used for classifying terms in English text in the context of database design. Domingue et al. [39] discuss the use of ontology as tools in communities of practice and how it can be used as a basis for knowledge services. Crampes and Ranwez [40] discuss their profile-based and ontology-driven approach to conceptual web navigation. Everett et al. [41] have developed a system supported by ontologies to identify conceptually similar documents. Such a system would form a basis for large-scale knowledge extraction from documents. Prior research also discusses the various benefits of using ontologies for different purposes [42]. Prior research has also attempted to map distribute ontologies to one another through the identification of semantic similarity among concepts [43]. The GLUE system [43] applies machine-learning techniques to create semantic mappings among ontologies. They also discuss the use of various practical similarity measures.

The capability of an ontology to facilitate reuse has been emphasized in prior literature [44]. Knowledge reuse has been identified as one of the key applications of ontologies [45]. Devedzin [46] explains how ontologies and software patterns are complementary in sharing common issues in knowledge reuse. Prior research has also emphasized the role of ontologies in structuring meta-data [47]. Stuckenschmidt [48] argues that ontologies specification and meta-data generation

are supplementary processes. They propose a semi-automatic generation of meta-data models based on ontologies. In short, ontologies have been found to be useful in a variety of domains. Here, we leverage the usefulness of ontologies in facilitating knowledge management and reuse.

6. Contributions and Future Research

The primary contribution of our research is use of ontologies in variability management for software families. Knowledge reuse is facilitated through the use of ontologies by our knowledge management system. Specifically, this research addresses the following issues:

- Using ontologies to define the scope of knowledge elements related to variability
- Knowledge reuse through the integration of a knowledge management system and ontologies

Our tool can be used to create intermediate representations for any ontology. This approach of knowledge reuse can be domain independent, i.e., we can use this knowledge management and ontology-centric approach to reuse knowledge for varied purposes.

Our ontology is by no means exhaustive. We expect this ontology to evolve based on further identification of concepts associated with variability. We are conducting further case studies in order to expand the ontology. We are also investigating the formalizing the ontology so as to assist in automated inferences. Future research will also focus on evaluating our ontology [49] and our current knowledge reuse approach of using an ontology for variability, in various product/service family development environments.

7. References

[1] T. Smith and J. Reece, "The Relationship of Strategy, Fit, Productivity, and Business Performance in a Services Setting," *Journal of Operations Management*, vol. 16, pp. 3-26, 1999.

[2] J. Pine, *Mass Customization: The new Frontier in Business Competition*. Boston, MA: Harvard Business School Press., 1993.

[3] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*. Upper Saddle River, NJ: Addison-Wesley, 2002.

[4] T. W. Simpson, "A Concept Exploration Method for Product Family Design," in *Mechanical Engineering*. Atlanta, GA: Georgia Institute of Technology, 1998.

[5] B. Kahn, "Variety: From the Consumer's Perspective," in *Product Variety Management*, T.-H. Ho and C. S. Tang, Eds. Norwell, MA: Kluwer Academic Publishers, 1998.

[6] M. A. Cusumano and A. Gawer, "The Elements of Platform Leadership," *Sloan Management Review*, vol. 43, pp. 51-58, 2002.

[7] P. Seybold, "Preparing for the e-Services Revolution," Patricia Seybold Group, Boston Customers.com Report April, 30 1999.

[8] G. Cugola and C. Ghezzi, "Program Families: Some Requirements Issues for the Process Languages," presented at 10th International Software Process Workshop, Dijon, France, 1996.

[9] E. W. Dijkstra, *Notes on Structured Programming*. London: Academic Press, 1972.

[10] D. L. Parnas, "On the Design and Development of Program Families," *IEEE Transactions on Software Engineering*, pp. 1-9, 1976.

[11] J. Coplien, D. Hoffman, and D. Weiss, "Commonality and Variability in Software Engineering," *IEEE Software*, vol. 15, pp. 37-45, November/December 1998.

[12] J. Bosch, G. Florijn, D. Greefhorst, J. Kuusela, H. Obbink, and K. Pohl, "Variability Issues in Software Product Lines," presented at Proceedings of the Fourth International Workshop on Product Family Engineering PFE-4, Bilbao, Spain, 2001.

[13] J. v. Gurp, J. Bosch, and M. Svahnberg, "Managing Variability in Software Product Lines," presented at LAC (Landelijk Architectuur Congres), Amsterdam, 2000.

[14] F. Bachmann and L. Bass, "Managing Variability in Software Architectures," *ACM SIGSOFTSoftware Engineering Notes*, vol. 26, pp. 126-132, 2001.

[15] J. Bosch, *Design and Use of Software Architectures*: Addison-Wesley, 2000.

[16] M. Martin, W. Hausman, and K. Ishii, "Design for Variety," in *Product Variety Management*, T.-H. Ho and C. S. Tang, Eds. Norwell, MA: Kluwer Academic Publishers, 1998.

[17] K. Lee, K. C. Kang, E. Koh, W. Chae, B. Kim, and B. W. Choi, "Domain-Oriented Engineering of Elevator Control Software: A Product Line Practice," in *Software Product Lines: Experience and Research Directions*, P. Donohue, Ed. Norwell, MA: Kluwer Academic Publishers, 2000.

[18] W. Swartout, "Ontologies," *IEEE Transactions on Intelligent Systems*, pp. 18-19, 1999.

[19] R. Neches, R. E. Fikes, T. Finin, T. R. Gruber, T. Senator, and W. R. Swartout, "Enabling technology for knowledge sharing," *AI magazine*, vol. 2, pp. 36-56, 1991.

[20] A. G. Perez and V. R. Benjamins, "Overview of Knowledge Sharing and Reuse Components," presented at Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden, 1999.

[21] M. F. Lopez, A. G. Perez, J. P. Sierra, and A. P. Sierra, "Building a Chemical Ontology Using Methontology and the Ontology Design Environment," *IEEE Intelligent Systems and their Applications*, 1999.

[22] M. Alavi and D. E. Leidner, "Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues," *MIS Quarterly*, vol. 25, pp. 107–136, 2001.

[23] S. Staab, R. Studer, H. P. Schnurr, and Y. Sure, "Knowledge processes and ontologies," *IEEE Intelligent Systems*, vol. 16, pp. 26 -34, 2001.

[24] K. C. Kang, S. G. Cohon, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-Oriented Domain Analysis (FODA): Feasibility Study," Software Engineering Institute, Carnegie Mellon University, Pittsburgh CMU/SEI-90-TR-21,, November 1990.

[25] A. Mili and S. M. Yacoub, "A Comparative Analysis of Domain Engineering Methods: A Controlled Case Study," presented at Proceedings of the Software Product Lines: Economics, Architectures and Implications - Workshop #15 at 22nd International Conference on Software Engineering, Limerick, Ireland, 2000.

[26] R. van Ommering, F. van der Linden, J. Kramer, and J. Magee, "The Koala Component Model for Consumer Electronics Software," *IEEE Computer*, vol. 33, pp. 78-85, 2000.

[27] A. v. d. Hoek, M. Mikic-Rakic, R. Roshandel, and N. Medvidovic, "Taming Architectural Evolution," presented at Proceedings of the 8th European Software Engineering Conference, Vienna, Austria, 2000.

[28] E. M. Dashofy and A. v. d. Hoek, "Representing Product Family Architectures in an Extensible Architecture Description Language," presented at Proceedings of the International Workshop on Product Family Engineering (PFE-4), Bilbao, Spain, 2001.

[29] J. C. Arpírez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez, "WebODE: A Scalable Workbench for Ontological Engineering," presented at International Conference on Knowledge Capture, Victoria, British Columbia, Canada, 2001.

[30] F. Lopez, "Overview of Methodologies For Building Ontologies," presented at IJCAI-99 Workshop on Ontologies and Problem Solving Methods, Stockholm, Sweden, 1999.

[31] C. W. Holsapple and K. D. Joshi, "A Collaborative Approach to Ontology Design," *Communications of the ACM*, vol. 45, 2002.

[32] V. C. Storey, R. H. L. Chiang, D. Dey, R. C. Goldstein, and S. Sundaresan, "Database Design with Common Sense Business Reasoning and Learning," *ACM Transactions on Database Systems*, vol. 22, pp. 471-512, 1997.

[33] D. Lenat and R. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project*. Redwood City, CA.: Addison-Wesley Publishing Co., Inc., 1990.

[34] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications.," *Knowledge Acquisition*, vol. 5, pp. 199-200, 1993.

[35] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swartout, "Enabling Technology for Knowledge Sharing," *AI Magazine*, vol. 12, pp. 36-56, 1991.

[36] S. Bergamaschi and C. Sartori, "On Taxonomic Reasoning in Conceptual Design," *ACM Transactions on Database Systems*, vol. 17, pp. 385–422, 1992.

[37] C. H. Goh, S. E. Madnick, and M. D. Siegel, "Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment.," presented at Third International Conference on Information and Knowledge Management, Gaithersburg, MD, 1994.

[38] R. Goldstein and V. Storey, "Common Sense Reasoning in Database Design," presented at IEEE Conference on Artificial Intelligence Applications, Los Alamitos, CA, 1995.

[39] J. Domingue, E. Motta, S. B. Shum, M. Vargas-Vera, Y. Kalfoglou, and N. Farnes, "Supporting Ontology Driven Document Enrichment Within Communities of Practice," presented at International Conference on Knowledge Capture, Victoria, British Columbia, Canada, 2001.

[40] M. Crampes and S. Ranwez, "Ontology-supported and ontology-driven conceptual navigation on the World Wide Web," presented at Conference on Hypertext and Hypermedia, San Antonio, Texas, 2000.

[41] J. O. Everett, D. G. Bobrow, R. Stolle, R. Crouch, V. d. Paiva, C. Condoravdi, M. v. d. Berg, and L. Polanyi, "Making Ontologies Work for Resolving Redundancies Across Documents," *Communications of the ACM*, vol. 45, 2002.

[42] T. Menzies, "Cost benefits of ontologies," *Intelligence*, vol. 10, 1999.

[43] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, "Learning to Map Between Ontologies on the Semantic Web," presented at Eleventh International Conference on World Wide Web, Honolulu, Hawaii, USA, 2002.

[44] X. Wang, C. W. Chan, and H. J. Hamilton, "Design of Knowledge-based Systems with the Ontology-Domain-System Approach," presented at 14th international conference on Software engineering and knowledge engineering, Ischia, Italy, 2002.

[45] M. Gruninger and J. Lee, "Ontology Applications and Design," *Communications of the ACM*, vol. 45, pp. 39-41, 2002.

[46] V. Devedzic, "Ontologies: Borrowing From Software Patterns," *intelligence*, vol. 10, 1999.

[47] P. C. Weinstein, "Ontology-based Metadata: Transforming the MARC Legacy," presented at Third ACM Conference on Digital Libraries, Pittsburgh, Pennsylvania, United States, 1998.

[48] H. Stuckenschmidt and F. v. Harmelen, "Ontology-based Metadata Generation from Semi-Structured Information," presented at international conference on Knowledge capture, Victoria, British Columbia, Canada, 2001.

[49] A. Gomez-Perez, "Some ideas and examples to evaluate ontologies," *11th Conference on Artificial Intelligence for Applications*, pp. 299 -305, 1995.