December 2004

An Ontological Approach to Support Domain Knowledge Learning in Information Systems Development

Lan Cao
Georgia State University

Han Woo
Georgia State University

An Ontological Approach to Support Domain Knowledge Learning in Information Systems Development

Lan Cao
Department of Computer Information Systems
Georgia State University
lcao@cis.gsu.edu

Han G. Woo
Department of Computer Information Systems
Georgia State University
hwoo@stuent.gsu.edu

ABSTRACT

This research focuses on providing integrated, structured, and user- independent domain knowledge for system analysts and designers. We take an ontological approach to representation of domain knowledge to support the process of obtaining domain knowledge in information system development. Domain knowledge learned by system analysts is their evolving perceptions of the application domain which is relatively stable and consistent. This research intends to capture the changing perceptions of application domain under the guidance of a consistent domain ontology. Another objective of this research is to link domain ontology with all design components to facilitate change management during ISD. A technique is proposed to apply ontology to modeling both the relative consistent reality of the world and the evolving understanding of an application domain, and to supporting change management during ISD process. We developed a prototype to demonstrate the technique using a sample ontology for apparel manufacturing.

Keywords

Ontology, domain knowledge, information system development, problem space, design space.

INTRODUCTION

Limited domain knowledge results in errors and deficiencies in understanding user requirement (Curtis et al., 1988), which in turn, have disastrous effects on subsequent design and implementation phases and on the quality of the resulting information systems. The deep application-specific knowledge required to successfully build an information systems is thinly spread through many software development staffs. Although individual staff members understand different components of the application, the integration of various knowledge domains required to integrate the design of a large, complex system is a scarcer attribute. As a result, software development requires a substantial time commitment to learning the application domain. Moreover, system analysts with varying degrees of domain knowledge interview customers and users, and then write specifications for software designers based on their understanding of the domain. In this process, relevant domain information is inevitably lost (Kraut and Streeter, 1995). Research in ISD has proposed techniques to help analysts to obtain domain knowledge, e.g., user participation in ISD process has been used as one way to obtain knowledge of application domain (Ives and Olson, 1984), however, it is surprising to find out that sometimes users assigned to the development team also lack knowledge of the domain (Schmidt et al., 2001). *This research focuses on an ontological approach for providing integrated, structured, and user- independent domain knowledge to system analysts and designers.*

The primary objective of this research is to propose a technique of representing problem domain by an ontological approach to support learning of domain knowledge in ISD. Domain knowledge learned by system analysts is their evolving perceptions of the application domain. One objective of this research is to capture the changing perceptions of application domain under the guidance of relative stable domain ontology. The learning of domain knowledge exists during the entire ISD process, not only during the system analysis stage. The changing business environment requires more learning effort. Another objective of this research is to link domain ontology with all design components to facilitate change management during ISD.

RELATED RESEARCH

To deal with the interaction with the changing business environment, prior research in ISD has focused on various process models such as spiral model (Boehm, 1988), layered behavior model (Curtis et al., 1988) and recently, the Rational Unified Process (RUP). These models and techniques emphasize the interactions between system analysts and the business environment, however, they do not describe how system analysts and developers learn domain knowledge from the interactions. Domain modeling has been recognized as a technique to model the procedures of an application domain that can be used for a variety of operational goals in support of specific software engineering tasks or processes (Iscoe et al., 1991). However, the domain models are representations of a piece of the application domain for a specific system on particular purposes. They do not provide a consistent description of a domain. This results in integration difficulties among different systems in one domain.

Domain knowledge and domain modeling

A problem domain defines the real world in which the envisioned system will operate. Developing an information system requires understanding of domain knowledge which is usually informal, implicit, ad hoc, and modeled only incompletely and indirectly in terms of problem-specific languages (Iscoe et al., 1991). A case study (Curtis et al., 1988) shows that developers have limited domain-specific knowledge at the early stage of a project. A thin spread of application domain knowledge typically leads to fluctuating and conflicting requirements which in turn cause a breakdown in communication. The cost of learning an application area is a significant corporate expense. The time estimated for a new developer to become productive ranged from six months to a year. Major changes in the business application or in the underlying technology required additional learning. Purao and colleagues (Purao et al., 2002) find that designers iteratively revisit the problem space during design process. While current techniques focus almost exclusively on design, there are very few techniques and tools to facilitate designer exploration of the problem space.

For large projects, diverse interests of multiple stakeholders cause conflicts between them. A win-win situation can be reached by resolving these conflicts through negotiation and renegotiation (Boehm, 1995). However, conflicts identification and resolution are difficult because of the lack of a common domain ontology which provides a consistent semantic framework for win conditions.

Domain modeling is an approach to formalize the domain knowledge in ISD processes (Gomaa, 1995; Iscoe et al., 1991). A domain model generally includes knowledge components such as a lexicon, thesaurus, ontology, taxonomy, templates, relationships, products, events, and actions (Tracz, 1994). Sugumaran et al. (2000) developed a framework for domain models to support reuse of domain knowledge in system analysis. The framework includes components in three levels: domain, processes, objectives (domain level), actors, actions (process level), data files and flows (data level), along with their relationships, with processes and actions as links between levels. System analysts can browse and select components in domain model repository and generate requirements for a new system. While the research supports using domain models to provide structured domain knowledge necessary to ISD, its purpose is to automate conceptual design by reusing domain knowledge. It is not designated to facilitate the learning of domain knowledge in information system development in general.

Ontology and Ontology in ISD

In philosophy, ontology is the study of the kinds of things that exist (Chandrasekaran et al., 1999). In AI, ontology refers to explicit representation of domain concepts (Gruber, 1992). Each ontology is a system of concepts and their relations, in which all concepts are defined and interpreted in a declarative way (Devedzic, 2002). In (Bench-Capon and Visser, 1997), the motivation for using ontologies in information systems is described as following: 1) knowledge sharing; 2) verification of a knowledge base; 3) software engineering considerations; 4) knowledge acquisition; 5) knowledge reuse; and 6) domain-theory development.

Ontology can be at a very high level and can be used across domains such as Sensus and WordNet, or it can be domain specific such as a chemical ontology (Lopez et al., 1999). No matter in which level, in this research we view ontology as an objective description of the real world.

Chandrasekaran et al. (1999) described the role of ontology in information systems development. Ontology is important for knowledge representation and knowledge sharing, which leads to build specific knowledge base and increase the potential for knowledge reuse. Ontology has been applied to ISD in area such as database design (e.g., Storey et al., 1998; Sugumaran and Storey, 2002; 2003), object-oriented system design, and knowledge–base systems building (e.g., Bench-Capon and Visser , 1997).

Storey et al. (1998) developed a database design ontology and incorporated it to an expert system. Sugumaran and Storey (2003) proposed an ontology-based approach to support ER modeling. These research use ontological approach to facilitate database design, however, their focuses are on improvement of conceptual design by comparing terms entered by users with ontological knowledge bases, rather than on learning domain knowledge in the whole ISD process.

Object-oriented design of software systems also depend on appropriate domain ontology. Objects, their attributes and procedures more or less mirror aspects of the domains that are relevant to the application (Chandrasekaran et al., 1999). Devedzic (2002) argues that the process of object-oriented analysis and design is the process of building a domain ontology that is specific to the application. In this research, ontology and OO components are considered as being different. Ontology is defined as the conceptualization of domain reality, which is developed by domain experts and provides structured, user-independent domain knowledge to system analysts; while OO components such as objects, classes, and hierarchies are the reflection of the domain ontology and are the designers' perception of domain reality. These design components belong to design domain. This research proposes an ontological approach to bridge the problem domain and design domain by linking the domain ontology to the design components.

ONTOLOGICAL APPROACH FOR BRIDGING PROBLEM DOMAIN AND DESIGN DOMAIN

In this section, a technique to bridge problem domain and design domain is proposed. The technique has two major purposes: 1) help designers to obtain domain knowledge by explore/search/retrieve domain ontology, and support designer's behavior of exploring the problem domain; 2) manage changes in problem domain and system design by linking design components and problem domain. Figure 1 shows the architecture of the technique.

Problem Domain and Design Domain

In this research, problem domain refers to a subset of the real world with which a computer-based information system is concerned, while design domain refers to the design solutions describing the system itself (Guidon, 1990). Unified Modeling Language (UML) proposed by Rational™ has been accepted as the de facto standard for modeling and implementing information systems. UML can create multiple perspectives of information systems artifacts to be built, which include functional, structural, behavioral and dynamic views. These techniques are used to analyze and design a system in design domain. In this research, UML is not the focus of the research, and the proposed techniques can work with other design components in design domain.

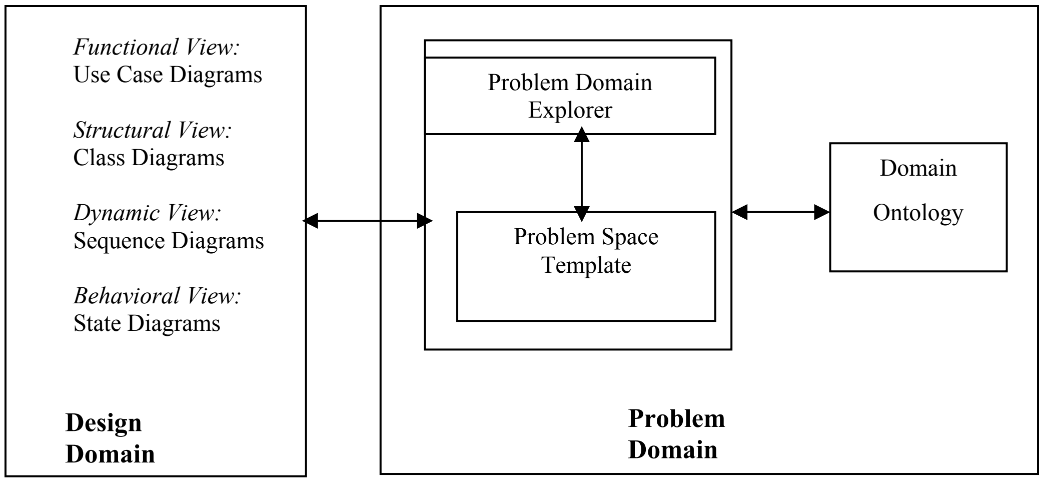

Figure 1. Architecture for Bridging Problem Domain and Design Domain

Domain Ontology

Domain ontology includes general knowledge of a domain. Domain is described in (Sugumaran et al., 2000) as "a region or a space that capture various features of the area under study". Examples of domain include apparel manufacturing, banking, and health insurance. The envisioned system may deal with only a part of the domain, e.g., a task scheduling system for apparel manufacturing, or the whole domain, e.g., an ERP system for apparel manufacturing. The constructs used in domain ontology in this research follow the ontological framework developed by Bunge (1977; 1979) and described in (Wand et al. , 1999). Those constructs are: things, properties and attributes, class and kind, laws and natural kinds, composites and the part-of relationship. In general, these constructs provide both semantic network of a world reality, and domain knowledge embedded in the structural constructs such as laws and relationships. Even though developing domain specific ontology is resource intensive, prior research has proved that such ontologies can be developed to facilitate ISD. For example, a law ontology in Dutch is developed and used to support the design of legal information systems (Bench-Capon and Visser, 1997).

In this research, we developed a sample domain ontology for apparel manufacturing. Figure 2 shows a part of the sample ontology.

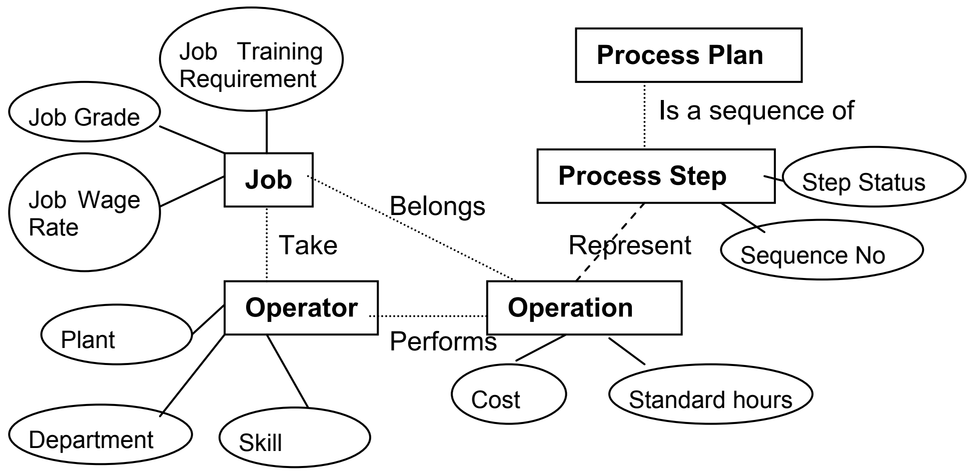

Figure 2. Partial Apparel Manufacturing Domain Ontology

Problem Domain Explorer

The Problem Domain Explorer is middleware between the UML design space and the domain ontology. It provides the interface to access the domain ontology. Analysts and designers explore the domain ontology to gain a basic understanding of the domain. They can also search the domain ontology to find specific topics within the domain and retrieve domain knowledge to understand the problem domain. For example, in developing a system to management resources in apparel manufacturing, the domain ontology provides the resources involved (e.g, *workstation* and *operator*), and the properties of each resource, and the relationships between the resources and other terms. (e.g., one *operator* has multiple skills on performing multiple *operations*, and can be assigned to multiple *workstations*. The capacity of a *workstation* depends on the *operation* it performs.) The above information can be easily retrieved by browsing or searching the domain ontology.

Problem Domain Explorer also provides a framework which analysts use to build a process model specific for the application. In this research, domain ontology is not the only resource of domain knowledge. It is not aimed to replace learning from users, customers, and documentations. Instead, it is the first step in understanding the domain. A process model is developed based on a user's domain knowledge using the *Problem Space Template* which is described in section 3.4.

Problem Space Template

A Problem Space Template is a place where system analysts model their perceptions of the problem domain based on domain knowledge obtained from the domain ontology and other resources. It includes description of the business (or manufacturing) processes, actors and activities, other stakeholders' roles, and resources etc. The difference between Problem Space Template and system requirements is that Problem Space Template records the understanding of an application domain, while requirements are how the system should respond to the application domain. For example, in developing a resource management system for apparel manufacturing, the description of all kinds of equipment, their capacity, and the process of job assignments are captured in the Problem Space Template. On the other hand, the system requirements specify how a user will interact with the system to assign a job to a workstation and update its workload.

Problem Space Template integrates the relatively stable domain knowledge such as the structure of processes, data, and entities with activities in business processes in detail, showing dynamic information such as the invocation of processes, activities and events. Figure 3 shows our initial, minimal meta-model for the Problem Space Template.

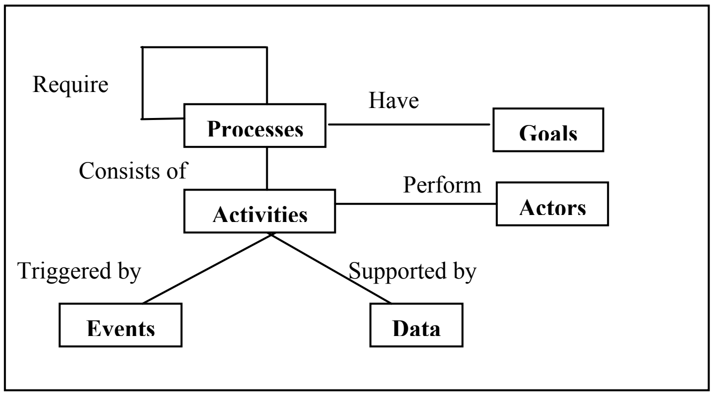

Figure 3. Problem Space Template Meta-Model

Problem Space Template is different from domain ontology in scope, nature, and focus. First, Problem Space Template is application specific while the domain ontology is more general. For example, a problem space template of a task scheduling system contains the rules and processes of a scheduler to break orders into pieces and assign them to the working floor, while domain ontology covers everything of apparel manufacturing. Secondly, Problem space template is dynamic while domain ontology is relatively stable. Problem Space Template is continually updated and refined to reflect changes in the design and in the business environment. Domain ontology remains stable during a period of time to provide consistent and stable domain knowledge. This consistency supports knowledge sharing and reuse, more importantly, the interoperatebility and integration of different systems within a domain. Finally, Problem Space Template is a process model which focuses on modeling the workflow of business process, while domain ontology does not contain the process information.

Even though they are different, Problem Space Template and domain ontology are linked to each other. In problem space template, the domain knowledge retrieved from domain ontology through Problem Domain Explorer is indexed and can be traced back to domain ontology. Problem space template is also linked to the design components. When there are changes in problem space template or system design, the changes can be traced to the other side, as well as to the domain ontology. This is similar to requirements traceability (Ramesh and Jarke, 2001) which links rationale for design decisions, requirement and system components.

PROTOTYPE IMPLEMENTATION

A prototype of our approach has been developed using Rational Rose® and Microsoft Access®. The sample domain ontology we developed is stored in a Microsoft Access Database. Problem Domain Explorer, a Rose Add-in program implemented in Visual Basic, provides an interface for user to explore, search, and retrieve a domain ontology. Problem Space Template is implemented using UML Activity Diagrams.

Browsing domain ontology through Problem Domain Explorer

We developed a sample domain ontology for apparel manufacturing based on Apparel Manufacturing Architecture (AMA) developed by Georgia Institute of Technology (1994). AMA is a comprehensive set of specifications for apparel enterprises. AMA has been used as a basis for several information systems developed by different developers and at different times. Those information systems are seamlessly integrated with each other. Even though AMA itself is not a domain ontology, it contains most of the information to derive an ontology for apparel manufacturing.

Domain ontology is presented with a tree-like structure as shown in Figure 4. A user can browse the ontology as she or he navigates through the tree structure. A user can also search the ontology by entering key works and synonyms. The related terms of a selected term is also displayed.

Building Process Model in Problem Space Template

The process model of the application domain is built in Problem Space Template using UML activity diagrams. UML activity diagrams consist of several constructs such as activity, state, transition, object, swimlane (actors) and decisions that can be used to represent the constructs in the Problem Space Template Meta-Model. *Activities* define the performance of a task. A *transition* indicates that an activity will perform certain specified actions when a specified event occurs or when certain conditions are satisfied. *Transition* is mapped to *Event* in Problem Space Template Meta-Model. *Object* represent something accompanying the *activities* such as the input and output of the *activities and is mapped to Data.* Figure 4 shows the process model of Resource Management and its link to the domain ontology.

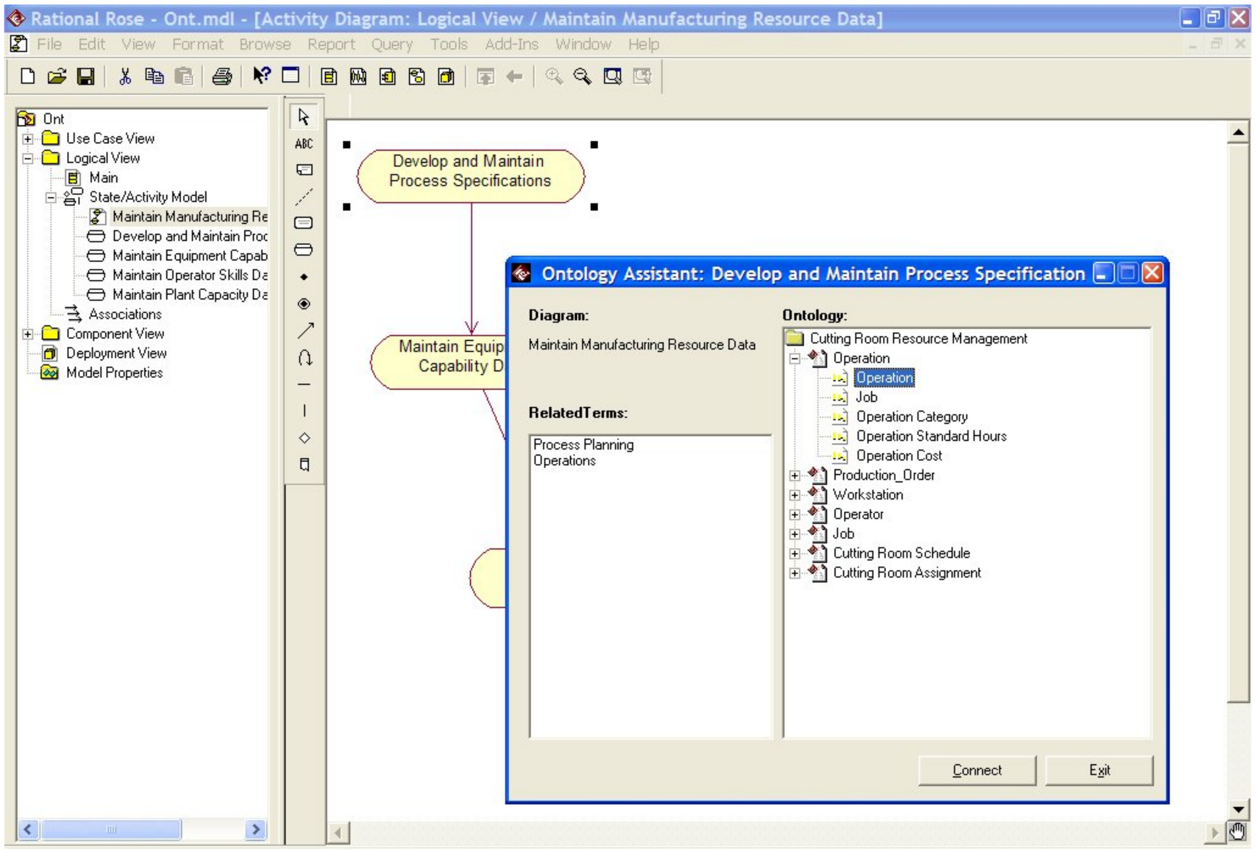
Rational Rose - Ont.mdl - [Activity Diagram: Logical View / Maintain Manufacturing Resource Data]
File Edit View Format Browse Report Query Tools Add-Ins Window Help
Ont
Use Case View
Logical View
Main
State/Activity Model
Maintain Manufacturing Re
Develop and Maintain Proc
Maintain Equipment Capab
Maintain Operator Skills Da
Maintain Plant Capacity Da
Associations
Component View
Deployment View
Model Properties
Develop and Maintain Process Specifications
Maintain Equip Capability D
Ontology Assistant: Develop and Maintain Process Specification
Connect
Exit

Diagram:
Maintain Manufacturing Resource Data

RelatedTerms:
Process Planning
Operations

Ontology:
- Cutting Room Resource Management
  - Operation
    - Operation
    - Job
    - Operation Category
    - Operation Standard Hours
    - Operation Cost
  - Production_Order
  - Workstation
  - Operator
  - Job
  - Cutting Room Schedule
  - Cutting Room Assignment

Figure 4. Problem Space Template of Resource Management

Each activity is linked to the domain ontology by right click the activity. Users can select the related terms from the ontology, or view the details of previously selected terms. Each activity is also linked to design components such as use cases and classes in class diagrams through the build-in mechanisms in Rose®. For example, Note is used to link an Use Case in Use Case diagrams to a Problem Space Template in Activity Diagrams.

CONCLUSION

The primary contribution of this research is a technique that facilitates the learning of domain knowledge in ISD. Limited domain knowledge causes many problems in ISD processes, even though techniques such as domain modeling have been developed, their focuses are more on knowledge reuse/sharing and design automation, and most of them are stand alone tools without integration with other design activities. This research focuses on domain knowledge learning process during ISD and provides a technique for designers to learn domain knowledge by exploring the problem domain.

The second contribution is the combination of ontological approach and problem space template to represent domain knowledge. Domain knowledge has two-fold characteristics: on one side, it reflects the domain reality which is structured and stable such as the terms and basic rules. On the other side, domain knowledge evolves as a result of learning process and changing business environment. The approach in this research integrates the two conflicting nature of domain knowledge.

Practitioners are expected to benefit from the research. Being supported by the tool, system analysts will understand the application domain as a whole and obtain necessary domain knowledge more precisely and quickly. The time and cost in learning domain knowledge will be largely reduced.

Research in both ontology and ISD can gain insights from this study. As discussed in (Guarino and Welty, 2002), ontology in computer science is "normally taken as synonymous with knowledge engineering in AI, conceptual modeling in database and domain modeling in OO design". However, ontology itself is beyond those applications. This research combines rigorousness of ontology and flexibility of its application into one technique. On the other hand, this research is a starting point for researchers to further investigate tools and techniques that assist system analysts to learn domain knowledge through the design process.

REFERENCES

1. Bench-Capon, T. J. M. and P. R. S. Visser (1997) Ontologies in Legal Information Systems, *Proceedings of International Conference on Artificial Intelligence and Law*, Melbourne, Australia.
2. Boehm, B. (1995) Software Requirements Negotiation and Renegotiation Aids, *Proceedings of 17th International Conference on Software Engineering*, Seattle, Washington.
3. Boehm, B. (1988) A Spiral Model of Software Development and Enhancement, *Computer*, 11 (4).
4. Bunge, M. (1977) Treatise on Basic Philosophy: *Vol 3: Ontology I: The Furniture of the World*, New York, NY: D. Reidel Publishing Co., Inc.
5. Bunge, M. (1979) Treatise on Basic Philosophy: *Vol 3: Ontology II: World of Systems*, New York, NY: D. Reidel Publishing Co., Inc.
6. Chandrasekaran, B., J. R. Josephson, and V. R. Benjamins (1999) What Are Ontologies, and Why Do We Need Them? *IEEE Intelligent Systems*, 14(1), 20-26.
7. Curtis, B., H. Krasner, and N. Iscoe (1988) A Field Study of the Software Design Process for Large Systems, *Communications of the ACM*, 31 (11).
8. Devedzic, V. (2002) Understanding Ontological Engineering, *Communications of the ACM*, 45 (4).
9. Gomaa, H. (1995) Domain Modeling Methods and Environments, *Proceedings of 7th International Conference on Software Engineering on Symposium on Software Reusability*, Seattle, WA, 256-258.
10. Gruber, R. R. (1992) Ontolingua: A Mechanism to Support Portable Ontologies, *Technical Report*. California, USA: Knowledge Systems Laboratory, Stanford University.
11. Guarino, N. and C. Welty (2002) Evaluating Ontological Decisions with Ontoclean, *Communications of the ACM*, 45 (2).
12. Guidon, R. (1990) Designing the Design Process: Exploiting Opportunistic Thoughts, *Human-Computer Interaction*, 5.
13. Iscoe, N., G. B. Williams, and G. Arango (1991) Domain Modeling for Software Engineering, *Proceedings of International Conference on Software Engineering*, Austin, Texas, United States.
14. Ives, B. and M. H. Olson (1984) User Involvement and MIS Success: A Review of Research, *Management Science*, 30 (5).
15. Kraut, R. E. and L. Streeter (1995) Coordination in Software Development, *Communications of the ACM*, 38 (3).
16. Lopez, M. F., A. Gomez-Perea, and J. P. Sierra (1999) Building a Chemical Ontology Using Methontology and the Ontology Design Environment, *IEEE Intelligent Systems*, Jan/Feb.
17. Purao, S., M. Rossi, and A. Bush (2002) Towards an Understanding of the Use of Problem and Design Spaces During Object-Oriented System Development, *Information and Organization*.
18. Ramesh, B. and M. Jarke (2001) Toward Reference Models for Requirement Traceability, *IEEE Transaction on Software Engineering*, 27 (1).
19. Schmidt, R., K. Lyytinen, M. Keil, and P. Cule (2001) Identifying Software Project Risks: An International Delphi Study, *Journal of Management Information Systems*, 17 (4).
20. Storey, V. C., D. Dey, H. Ullrich, and S. Sundaresan (1998) An Ontology-Based Expert System for Database Design, *Data and Knowledge Engineering*, 28.
21. Sugumaran, V., M. Tanniru, and V. C. Storey (2000) Supporting Reuse in Systems Analysis, *Communications of the ACM*, 43 (11).
22. Sugumaran, V., and V. C. Storey (2002), Ontologies for Conceptual Modeling: Their Creation, Use, and Management, *Data and Knowledge Engineering*, 42(3).

23. Sugumaran, V., and V. C. Storey (2003) Supporting Database Designers in Entity-Relationship Modeling: An Ontology-based Approach, *Proceedings of 24th International Conference on Information Systems*, Seattle, WA.
24. Tracz, W. (1994) Domain-Specific Software Architecture (DSSA) Frequently Asked Questions, *Software Engineering Notes*, 19 (2).
25. Wand, Y., V. C. Storey, and R. Weber (1999) An Ontological Analysis of the Relationship Construct in Conceptual Modeling, *ACM Transactions on Database Systems*, 24 (4).