Ontology-Based User Requirement Analysis for Banking Application

Ruey-Shun Chen, Chan-Chine Chang, Chia-Chen Chen and Isabel Chi
Institute of Information Management, National Chiao Tung University, Taiwan
rschen@iim.nctu.edu.tw

Abstract

Leaning toward user requirements and constructing a standard method to build up a high-quality application in limited developing time, and enhancing the model flexibility for user to change their requirements in time are main objectives of this research. In this study, we created ontology to represent user domain in a deposit system, including operation procedure, entity, interaction and action from deposit business process with classes and relationships in tree maps. It classifies and reorganizes these domains by using the ontology technology. Though ontology is created from user requirements, the results show it can construct a user-driven software for defending frequent requirement changes.

1. Introduction

When developing a new software application, user requirements are listed in a written statement by an experienced domain expert or old-timely system analyst. Each expert or analyst describes user requirements and system figuration in different templates and document patterns. It is difficult for a new person to comprehend the user requirements in a short time. Usually we describe the user requirements by focusing on the core of the user needs, such as specific process patterns or some important business rules. If we are not familiar with this domain knowledge, it is impossible to collect the user requirements. For an application developing company, it is not practical or workable to focus on one business domain all the time. We have to find a way to standardize our developing process in a formularized form and through the developing process extracting knowledge from user requirements in this domain. The purpose is taking the least time to know what a user needs are and to help system analyst to realize user requirements deeply.

The deposit system is one of the banking applications from past developing experience. We review how an application links a knowledge-extraction method with ontology to achieve continuous system developing knowledge support and guide an analyst to develop a new application with standard templates. It provides this knowledge in a machine-readable format that will be maintained in a requirement analysis knowledge base [11]. The process from knowledge extraction is further enhanced using a standard template that provides extended software engineering technology and ontology terminology.

2. Literature Review

2.1 Ontology

Ontology is a system of categories for classifying and talking about the things that are assumed to exist. Ontology conceptualizes a domain into a machine-readable format. We use the term ontology to denote a specification of a conceptualization. That is, a ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set-of-concept definitions. We have been designing ontology for the purpose of enabling domain knowledge sharing and reuse with user.

The subject of ontology is the study of the categories of things that exist or may exist in some domain. An informal ontology may be specified by a catalog of types that are either undefined or defined only by statements in a natural language. A formal ontology is specified by a collection of names for concept and relation types organized in a partial ordering by the type-subtype relation [1].

For banking software, deposit system, we created a ontology to represent user domain, including operation procedure, entity, interaction and action, from deposit business process with classes and relationships in tree maps. We classify and reorganize them using the ontology as an implementation guide. We try to find a way to get closer to users, and let users realize what we have designed for them earlier. Though ontology is created from user requirements, we hope to construct a user-driven software for defending frequent requirement changes.

2.2 Application analysis

Because software, like banking software or application, is embodied knowledge, and because that knowledge is initially dispersed, tacit, latent and incomplete in large measure, application development is a social learning process. The process is a dialogue in which the knowledge that must become the software is brought together and embodied in the software. The process provides interactions between users and designers, between users and evolving tools, and between designers and evolving technology. It is an iterative process in which the evolving tool itself serves as the medium for communication, with each new round of the dialogue eliciting more useful knowledge from the user involved.

Three different viewpoints, as shown in Figure 1, are used frequently when planning about work systems that use IT extensively [6].

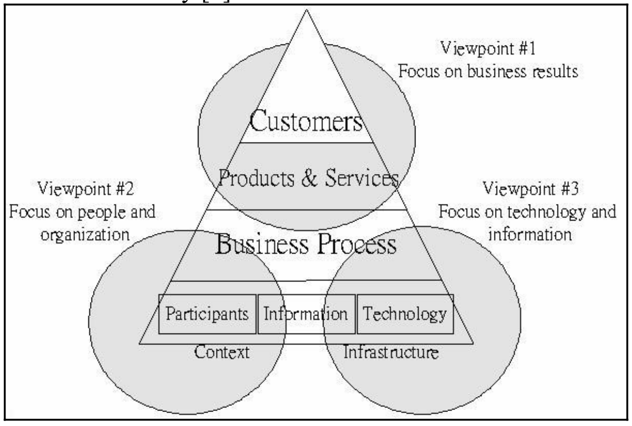


Figure 1. Three viewpoints for thinking about a system in an organization

To solve actual problems in a bank or in an organization, an application engineer or a team of engineer must incorporate a development strategy. The strategy is often referred to as a process model or a software engineering paradigm. A process model for software engineering is chosen based on the nature of the application, the methods and tools to be used, and the controls and deliverables that are required. In this study, we use the linear sequential model as basis for a development of the application process. The linear sequential model sometimes called the waterfall model suggests a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing, and support[5].

2.3 Banking case study

Many information systems in bank are very critical to support bank daily operations. Most of them exist in the system that data entry from a transaction processing. A transaction processing system collects and stores data about transactions and sometimes controls decisions made as part of a transaction. A transaction is a business event that generates or modifies data stored in an information system. They are used widely in bank, including writing a check, use a credit card, or pay a bill sent by a bank [2]. This kind of information system are designed based on detailed specifications for how the transaction should be performed and how to control the collection of specific data in specific data formats and in accordance with rules, policies, and goals of the organization. A well-designed process transaction system can minimize data entry errors by automatically filling in data such as customer name or unit price once the user has entered the customer id or product number. Data entry is very important for users to input transaction data. It is an efficient way for input process and to keep data consistent with other data in the database.

Most of software systems or applications in bank are transaction type systems. We choose a deposit system as an example. There are three major modules including Basic module, Process module, Report module, and 17 sub-modules and 43 user interfaces.

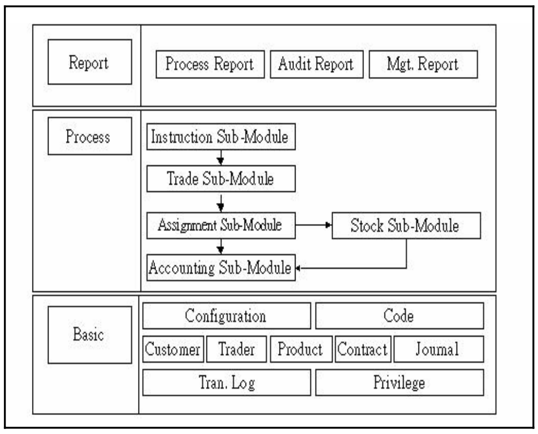


Figure 2. Deposit system function relational diagram

3. Knowledge extraction method

What knowledge exists in software requirement analysis for the process of application development? Software requirements engineering is a process of discovery, refinement, modeling, and specification. Models of the required data, information and control flow, business process, and operational behavior are created. We are pushed to design some classes to support the same type of application, transaction processing type system, from banking software. One of important reasons and benefits is reuse. For example, the analysis of requirements for a new application indicates that 100 classes are needed. Two teams are assigned to build the application. Each will design and construct a final product. Team A does not have access to a class library, and therefore, it must develop all 100 classes from scratch. Team B uses a robust class library and fins that 55 classes already exist. The result is that team B will finish the project much sooner than team A and the cost of team B product will be lower than the cost of team A product. Moreover, team B deliveries a fewer defect product than team A. Where did the robust class library come from? To answer the questions, the organization that created and maintained the library has to apply domain and requirement analysis [8]. In other words, the ideas to design a common class library are from the knowledge base of user requirement and business domain.

Requirements analysis is an application development task that bridges the gap between system level requirements engineering and software design. These activities result in the specification of application operational characteristics, data, function and behavior, indicate application interface with other system elements, and establish constraints that software must meet. Requirements analysis provides the software designer with a representation of information, function, and behavior that can be translated to data, architectural, interface, and component-level class designs [7].

To realize the operational concepts and their relation to automated capabilities is an important thing. Only from these processes, we can know what the users really want and get an accurate pattern for integrated information blueprint. We assume DOD architecture framework defines a common approach for Department of Defense architecture description development, presentation, and integration. The framework is intended to ensure that architecture descriptions can be compared and related to organizational boundaries, including joints and multinational boundaries.

The Framework defines three related views of architecture: operational (OV), systems (SV), and technical standards (TV) as Figure 3[3, 4]. Each view is composed of sets of architecture information that are depicted via graphic, tabular, or textual products. The All-DoD Core Architecture Data Model (CADM) defines the data structure and relationship for architecture information. The purpose of the DoD Architecture Framework is to provide guidance for describing architectures. The Framework provides the rules, guidance, and product descriptions for developing and presenting architecture descriptions that ensure a common denominator for understanding, comparing, and integrating architectures.

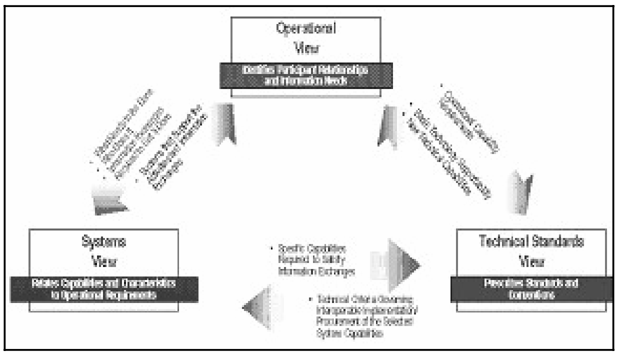

Operational View
Identifies Participant Relationships and Information Needs
Systems View
Relates Capabilities and Characteristics to Operational Requirements
Technical Standards View
Prescribes Standards and Conventions
Specific Capabilities Required to Satisfy Information Exchanges
Technical Criteria Governing Interoperable Implementation/ Procurement of the Selected System Capabilities

Figure 3. Linkages among views.

In the framework, there are three major perspectives(view), that logically combine to describe an integrated architecture. These include the operational view(OV), system view(SV), and technical standards view(TV). Each of the views depicts certain architecture attributes. The OV is a description of the tasks and activities, operational elements, and information exchanges required to accomplish DoD missions. The SV is a description, including graphics of systems and interconnections providing or supporting DoD functions. For the individual system, the SV includes the physical connection, location, and identification of key hardware and software. The SV associates resources to the OV and its requirements per standards defined in the TV. The TV is the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements whose purpose is to ensure that a conformant system satisfies a specified set of requirements.

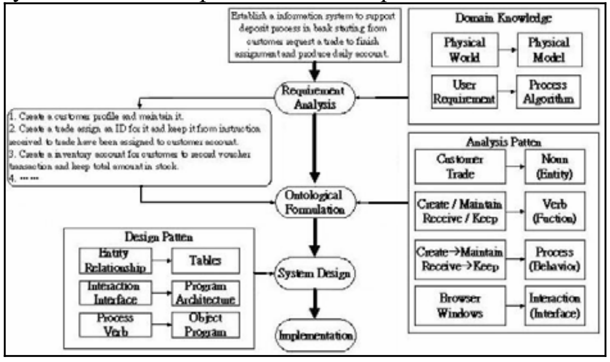

Establish a information system to support deposit process in bank starting from customer request a trade to finish assignment and produce daily account.
Domain Knowledge
Physical World
Physical Model
User Requirement
Process Algorithm
Requirement Analysis
1. Create a customer profile and maintain it.
2. Create a trade assign an ID for it and keep it from instruction received to trade have been assigned to customer account.
3. Create a inventory account for customer to record voucher transaction and keep total amount in stock.
4. ......
Analysis Patten
Customer Trade
Noun (Entity)
Ontological Formulation
Create / Maintain Receive / Keep
Verb (Fuction)
Design Patten
Entity Relationship
Tables
Create→Maintain Receive→Keep
Process (Behavior)
Interaction Interface
Program Architecture
System Design
Process Verb
Object Program
Browser Windows
Interaction (Interface)
Implementation

Figure 4. An example of knowledge extraction

To extract relationships from user requirements between a pair of entities, we need domain-specific knowledge, which we can infer from the ontology and use to determine required and expected relations between the entities. Figure 4 [1] shows the extraction process from the application target "*Establish a information system to support deposit process in bank starting from customer request a trade to finish assignment and produce daily account.*" By the application developing process direct access to the ontology concepts and relations.

4. Knowledge extraction

4.1 Domain knowledge

To provide some of the guidance missing from the general process, following the principle-based system analysis method and DOD_AF [3, 4] principle, a practical approach business professionals can use from analyzing requirements. The method can be used in a number of ways. First, to help organize the analysis when business professionals and domain experts must build their own small physical model for the application or information system using exposed and accumulated domain knowledge. Second, it is a way to create an initial understanding of a situation and even a tentative recommendation before starting a collaboration whit IT professionals. Third, it is a way to make sure that an ongoing collaboration between business and IT professionals balances business issues and computer system details. Fourth, it is a way domain and IT professionals can make sure that the process model coming out from their experience and user requirements is workable and that they have an adequate understanding of the business situation.

Detailed user requirement, operation scenario, system aspect, technical platform included, will be prepared after physical domain description and process algorithm constructing in user requirement statement document. Base on the requirement statement, we continue the next step for system analysis.

4.2. Pattern analysis

To extract the knowledge from detailed user requirement statement, we have to use predefined analysis pattern to reduce system variations among relations defined in the ontology. These include entity, function, process or algorithm, and interface.

At first, we mark the noun and verb in every sentence in a requirement statement. It analyzes each paragraph syntactically and semantically to identify relevant knowledge [13, 15]. The Apple Pie Parser groups phrase that the syntactical analysis determines to be grammatically related. Using semantic analysis, the tool then locates a sentence main components, subject, verb, object, and so on, and identifies names entities using some tools, like GATE and WordNet. This application does not use any tool to aid semantic analysis in the requirement statement. We just mark and analyze each sentence in the requirement statement. And then we assemble these verbs in the statement, and map them to the process algorithm; we draw a map tree to present their relationships and procedures in Figure 5.

In Figure 5, we focus on the branch P1 Basic Module and M1 Customer Sub-Module. Each node on this branch was composed by a noun and a verb and its parent is presented by a service. For example, in a requirement statement, one list is "Create a customer profile and maintain it." The "customer" is a noun meaning a person. The extracted synonyms for the verb sense include "create" and "maintain". The word "maintain" means to maintain customer profiles and records including delete, update and query. The four actions, create, delete, update, query construct one service –S1 Customer Data prepared by the system. The noun "customer" also means a entity and the verbs "create" and "maintain" mean the function that will be provided in a deposit system. We give the entity, customer, a ID number E1 and also give an ID for each function: create (F1), delete (F2), update (F3), and query (F4).

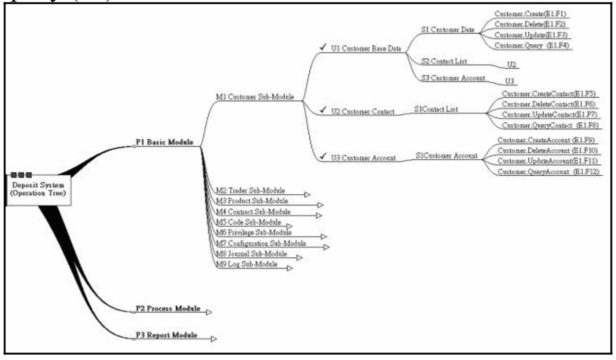


Figure 5. Analysis for deposit system – operation process tree

Let us show the details about Entity (E1-Customer) of function F1 (Customer Create) in Figure 6. Each entity provides a lot of functions for the specific application. We describe it with the function tree. For example, entity customer provides 12 functions for a deposit system. The function tree presents the relationships between processes and functions with these different trees. When we talk about the business process, please check the process tree. Each branch presents the business process. This application will support to write in requirement statements and each node presents the activities around the domain entity. These activities are supported in the application by entity functions. One of important concepts is that each point behind one branch means a decision point in a domain. Which branch system will be provided for users in the right time? One decision point means a system configuration parameter decides the system dynamic flow rule. In Figure 5, service S1, S2 and S3 provide customer basic data maintenance, customer contact list maintenance and customer account maintenance. In the process tree show S2 Contact List and S3 Customer Account will be triggered by a user login to the interface U2 or U3 when one user logins to the interface Customer Base Data . These branches, nodes, and decision points construct the application and represent the behavior in a machine-readable format.

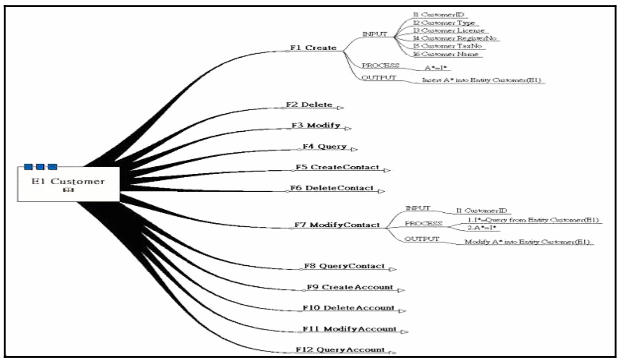
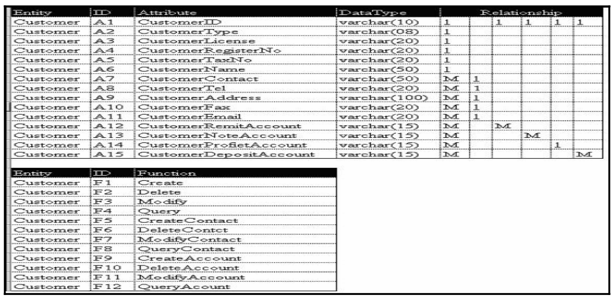


Figure 6. Analysis for deposit system – function tree

Figure 6 shows all of functions are defined by process tress. Each function is decomposed into 3 parts Input, Process, and Output, and will be edited by a program. Just like function specification, it is essential to describe the input parameter, process condition and the function output. For example, the function F1 Create defined by entity (E1 Customer), shows in Figure 7, we know some attributes customer ID, customer type, customer license, customer register no, customer tax no, customer name are input parameters. We give these input parameters some fixed variable numbers I1, I2, I3, I4, I5, I6, after users input data in these parameters, assign them into another variable A*(A1-A6), prepared insert into the table mapping to entity customer. These functions are defined by a function tree and will be performed through a user interface.



Entity	ID	Attribute	DataType	Relationship					
Customer	A1	CustomerID	varchar(10)	1		1	1	1	1
Customer	A2	CustomerType	varchar(08)	1					
Customer	A3	CustomerLicense	varchar(20)	1					
Customer	A4	CustomerRegisterNo	varchar(20)	1					
Customer	A5	CustomerTaxNo	varchar(20)	1					
Customer	A6	CustomerName	varchar(50)	1					
Customer	A7	CustomerContact	varchar(50)	M	1				
Customer	A8	CustomerTel	varchar(20)	M	1				
Customer	A9	CustomerAddress	varchar(100)	M	1				
Customer	A10	CustomerFax	varchar(20)	M	1				
Customer	A11	CustomerEmail	varchar(20)	M	1				
Customer	A12	CustomerRemitAccount	varchar(15)	M		M			
Customer	A13	CustomerNoteAccount	varchar(15)	M			M		
Customer	A14	CustomerProfietAccount	varchar(15)	M				1	
Customer	A15	CustomerDepositAccount	varchar(15)	M					M

Entity	ID	Function
Customer	F1	Create
Customer	F2	Delete
Customer	F3	Modify
Customer	F4	Query
Customer	F5	CreateContact
Customer	F6	DeleteContct
Customer	F7	ModifyContact
Customer	F8	QueryContact
Customer	F9	CreateAccount
Customer	F10	DeleteAccount
Customer	F11	ModifyAccount
Customer	F12	QueryAcount

Figure 7. Analysis for deposit system – entity form

Figure 7 and Figure 8 present the detailed description about entity Customer. Figure 7 shows the data model of entity customer that consists of three interrelated pieces of information: the entity (data object), the attributes that describe the data object, and the relationships between these attributes and the function list provided by this entity. Figure 8 shows the relationships that connect data objects to one another [5]. In the system specification, we have to describe defined data elements, data type, and their relationships [9]. To define these data elements, we follow the entity type and relationship in business domain.

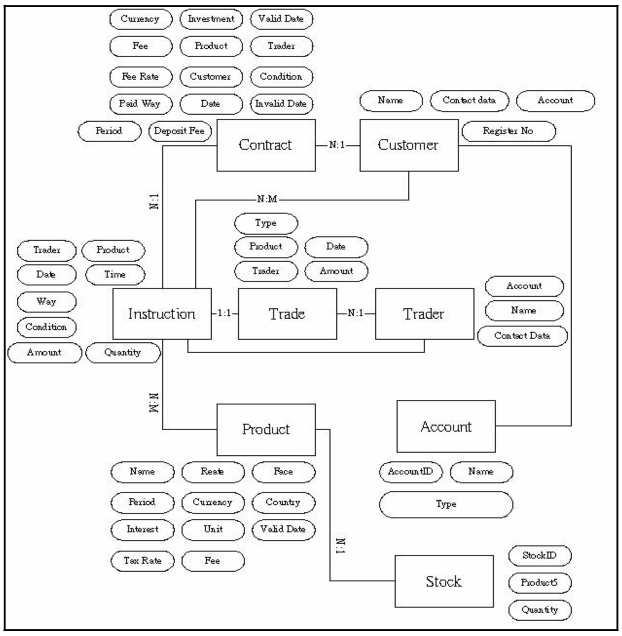


Figure 8. Analysis for deposit system–entity relational diagram

4.3. Pattern design

Design has been described as a multi-step process in which representations of data and program structure, interface characteristics, and procedural detail are synthesized from user requirements analysis in ontological formulation present. Design is information driven. Software and application design methods are derived from consideration of each of the three domains of the analysis model. The data, functional, and behavioral domains serve as a guide for the creation of the software design. We start from the architectural design that is the preliminary blueprint from which application is constructed. The deposit system performs in the IT architecture as Figure 9.

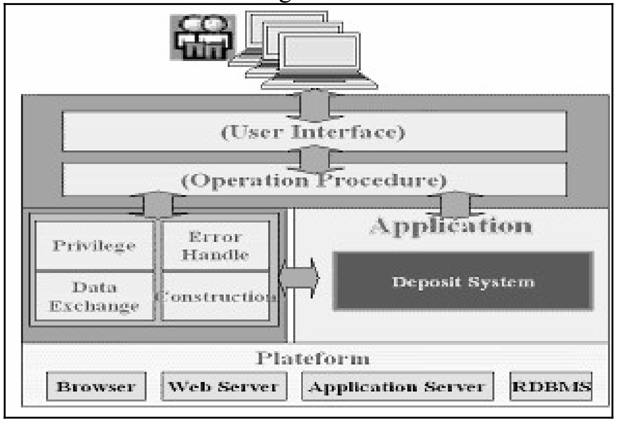


Figure 9. Deposit system architecture.

5. Conclusions

Today leading businesses are increasingly aware that the knowledge of their employee is one of their primary assets. In an application developing company that relies heavily on unique competencies and methods, knowledge has more competitive significance than physical assets because the physical assets can be replaced or replenished more easily. The companies with the best results to date stitch technologies together into a system that operates effectively and that is genuinely supported by the culture and application development. Learning from the user requirement and constructing our method to build up a high-quality application in limited developing period with the flexibility for users to change their requirements in a time.

For a deposit application developing in banking software, developing in an initial experiment, running deposit application to search and extract information from our customer, we set up the knowledge extraction way to select applications from the past system. These developing documents include system requirement statement, system design specification, and running program. In this study, analysis deposit application requirement includes 42 functions. The extraction process identified one type method for transaction process type system developing, but it is most application in general business. Four patterns, operation tree, function tree, entity form and entity relationship diagram are for software analysts to present their system figuration and three patterns are for system designers to present the system in machine-readable type, included program structure, common objects and table lists.

References

[1] Harith Alani, Sanghee Kim, David E.Millard, Mark J.Weal, Wendy Hall, Paul H.Lewis and Nigel R.Shadbolt, "Automatic ontology-based knowledge extraction from web documents", *IEEE Intelligent Systems* vol.18 (1), 2003, pp.14-21.

[2] "BASE REPAIR SOP—Level5", Military of army, 2000

[3] "DoD Architecture Framework—Volumn I: Definitions and Guidelines", Department of defines, 2003

[4] "DoD Architecture Framework—Volumn II: Product Description", Department of defines, 2003

[5] Roger s. Pressman, *Software Engineering – A Practitioner Approach*, McGraw-Hill Higher Education, 2001

[6] Steven Alter, *Information System – The Foundation of E-Business*, Prentice Hall, 2002.

[7] S. Handschuh, S. Staab, and F. Ciravegna, "SCREAM Semi-Automatic Creation of Metadata, Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web", , *Lecture Notes in Artificial Intelligence, no. 2473*, Springer-Verlag, 2002, pp.358–372.

[8] M. Vargas-Vera et al., "Knowledge Extraction Using an Ontology-Based Annotation Tool", *Workshop on Knowledge Markup & Semantic Annotation, ACM Press*, 2001, pp. 5–12**.**

[9] L. Rutledge et al., "Generating Presentation Constraints from Rhetorical Structure", *Proc. 11th ACM Conf. Hypertext and Hypermedia, ACM Press*, 2000, pp.19–28.

[11] R. Yangarber and R. Grishman, "Machine Learning of Extraction Patterns from Unannotated Corpora: Position Statement", *Workshop on Machine Learning for Information Extraction, 14th Eur. Conf. Artificial Intelligence, IOS Press*, 2000, pp. 76–83.

[12] S. Kim et al., "Artequakt: Generating Tailored Biographies with Automatically Annotated Fragments from the Web", *Workshop Semantic Authoring, Annotation & Knowledge Markup, 15th Eur. Conf. Artificial Intelligence, IOS Press*, 2002, pp. 1–6.

[13] H. Alani et al., "Managing Reference: Ensuring Referential Integrity of Ontologies for the Semantic Web", *Lecture Notes in Artificial Intelligence, no. 2473, Springer-Verlag,* 2002, pp. 317–334.

[14] D.T. Michaelides et al., "Auld Leaky: A Contextual Open Hypermedia Link Server", *Lecture Notes in Computer Science, no. 2266, Springer-Verlag,* 2001, pp. 59–70.

[15] S. Staab, A. Maedche, and S. Handschuh, "An Annotation Framework for the Semantic Web", *Proc. 1st Int Workshop MultiMedia Annotation* , 2001.