Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2006 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2006

Exploring Quality Dependencies among UML Artifacts Developed by Novice Systems Analysts

Narasimha Bolloju
City University of Hong Kong

Exploring Quality Dependencies among UML Artifacts Developed by Novice Systems Analysts

Narasimha Bolloju
Department of Information Systems
City University of Hong Kong
narsi.bolloju@cityu.edu.hk

ABSTRACT

Capturing and representing information systems requirements using quality artifacts is an important step in successful implementation. Better quality artifacts in the early stages of systems development help in early detection and correction of errors. Although UML is widely used for modeling systems requirements, it is often difficult for novice systems analysts to develop quality UML artifacts. This paper presents an application of the conceptual model quality framework proposed by Lindland et al (1994) for identifying quality dependencies among the UML artifacts developed by novice systems analysts in an undergraduate course on object-oriented systems analysis and design. Findings from this study offer directions for enhancing teaching and learning of systems analysis techniques with UML.

Keywords

UML artifacts, novice systems analysts, Quality, teaching and learning

INTRODUCTION

Capturing information systems requirements and representing those requirements using appropriate models is an important part of information systems development process. Such models also assist communication among stakeholders and other systems developers. Many information systems development project failures have been linked to problems associated with the requirements capturing (Chaos Report, 1994). Highlighting the importance of quality conceptual modeling, Wand and Weber (2002) state that better quality models facilitate early detection and correction of errors. During this process of conceptual modeling, novice systems analysts encounter difficulties, when compared to experienced analysts, in the areas of domain-specific knowledge, problem-structuring and cognitive process (Schenk et al., 1998). Further, the absence of established validation procedures (Shanks et al., 2003) makes the requirements specification a complex task to perform efficiently and effectively for novice analysts.

Nowadays, different techniques available in the Unified Modeling Language (UML, 2004) are being widely used for modeling systems requirements. UML 2.0 provides 12 types of artifacts for documenting the system requirements from different perspectives. A typical systems analyst is expected to be familiar with many of these techniques. Many researchers (e.g., Agarwal and Sinha, 2003; Siau and Cao, 2001) observe difficulties and complexities associated with using UML. Several practitioners in this field offer recommendations and guidelines (e.g., Ambler, 2003; Cockburn, 2001) and some suggest employing commonly used patterns (e.g., Fowler 1997; Adolph and Bramble, 2003) for effective and efficient use of various modeling techniques available in UML.

Despite the availability of such guidance and recommendations, typical novice systems analysts fail to get maximum benefit due to the cognitive overload associated with those guidelines and recommendations. Investigating specific difficulties frequently encountered by novice analysts from the quality perspective can be helpful in identifying a smaller set of problems and relationships among those problems. Such investigations can help in developing better training procedures and help providing enhanced support in the modeling process.

This study aims to apply the conceptual model quality framework proposed by Lindland *et al* (1994) to identify errors frequently committed by novice systems analysts while developing some commonly used UML artifacts, and then to identify intra- and inter-artifact quality dependencies. It is expected that a better understanding of such dependencies help in addressing specific problems associated with the modeling process and possibly help in developing methods and tools dealing with those problems.

The following section reviews the conceptual model quality framework proposed by Lindland et al (1994) and discusses its suitability for evaluating quality of UML artifacts. The third section details the method used for analyzing UML artifacts for identifying frequent errors and intra- and inter-artifact error clusters. Thereafter, the findings and implications of this research are presented.

LITERATURE REVIEW

Use-case driven modeling is a commonly employed approach in systems development based on object-oriented methods. In this approach, use case models (comprising use case diagrams and use case descriptions) are developed first to capture the system requirements, and then these models drive the development of remaining artifacts such as domain models (class diagrams) and dynamic models (interaction diagrams). Since a set of existing artifacts are used to develop the next set of artifacts, it is important to ensure the quality of artifacts at each and every stage of development, especially in the initial stages. The focus this study is limited to the UML artifacts commonly used in the initial stages (i.e., use case diagrams and descriptions, class diagrams and sequence diagrams).

Quality of these UML artifacts can be evaluated following approaches similar to those used for evaluating conceptual model quality. Wand and Weber (2002) describe a framework for research on conceptual modeling comprising four elements – grammar, method, script and context. According to their framework, scripts are the models or artifacts produced using a grammar (e.g., UML), following a method (e.g., use case driven modeling) in a given context (e.g., developing activity diagrams for business process redesign).

A wide range of studies on conceptual model quality have been surveyed by Wand and Weber (2002). Among these studies, frameworks for conceptual model quality provide a systematic structure for evaluation. Genero and Piattini (2002) review four major frameworks for this purpose and describe the key elements of each framework. Many of these frameworks offer mostly subjective ways of evaluating the quality of conceptual models.

Lindland et al (1994) have argued about the need for a framework addressing both process and product for treatment of quality and proposed a framework, borrowing three linguistic concepts viz. syntax, semantics and pragmatics, suitable for categories quality of conceptual models. Then they applied these concepts to four aspects of modeling viz. language, domain, model and audience participation (see Figure 1).

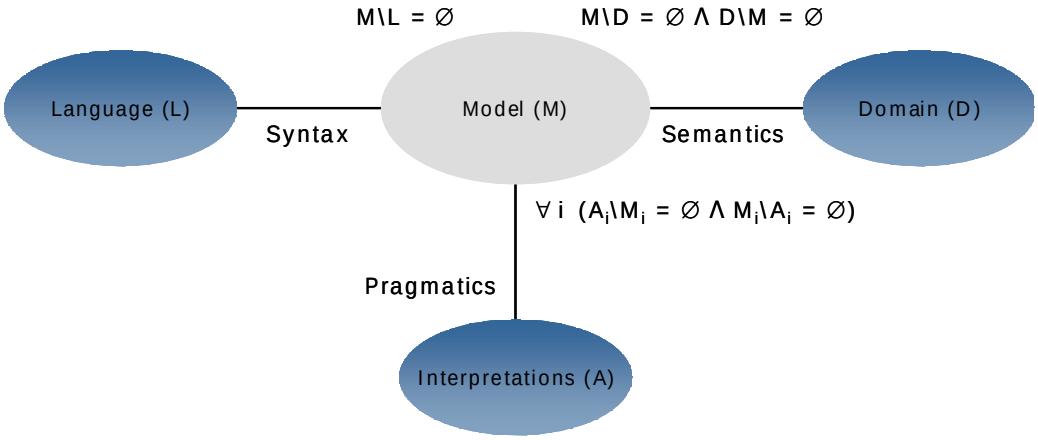

Figure 1. Lindland et al's Framework for Conceptual Model Quality

Syntactic quality: Syntactic correctness of a model (M) implies that all statements in the model are according to the syntax of the language (L), i.e., M\L = ∅. This category captures how a given model adheres to the language rules (i.e., syntax). Therefore, less number of errors and deviations from the rules indicate better syntactic quality.

Semantic quality: This category captures the quality of a model in terms what the model lacks something that is present in the domain (D) and what the model includes something that is not present in the domain. Semantic quality is described in terms of validity and completeness goals. The validity goal specifies that all statements in the model are correct and relevant to the problem domain; i.e., M\D = ∅. The completeness goal specifies that the model contains all statements about the problem domain that are correct and relevant; i.e., D\M = ∅. It is, however, possible that these two goals cannot be achieved realistically unlike the syntactic correctness.

Pragmatic quality: This quality category addresses the comprehension aspect of the model from the stakeholders' perspective. Pragmatic quality captures how the model has selected "from among the many ways to express a single meaning" and it essentially deals with making the model easy to understand. The comprehension goal specifies that all audience members (or interpreters) completely understand the statements in the model that are relevant to them (i.e., the model projections).

The three above categories address different aspects of quality that require increasingly more effort and expertise to achieve. This framework has been applied by researchers in different situations such as evaluating the quality of information models (Moody et al 2003) and comparing quality of different ontology languages and tools (Su and Ilebrekke 2005). Extensions to this framework have also been proposed (e.g., Krogstie et al 1995).

Since UML artifacts produced in early phases are more conceptual in nature, this framework can be applied effectively to evaluate the quality of such artifacts. In addition, considering the fact that many of the UML artifacts model the system requirements from different perspectives it would also be helpful to study and analyze any dependencies in intra- and inter-artifact quality. Results from such analysis are expected to contribute towards better understanding of frequent errors that lead to quality problems and thereby help in developing methods and tools for modeling systems requirements effectively and efficiently.

METHOD – ANALYSIS OF UML ARTIFACTS

This section provides details of specific sets of artifacts analyzed, and the methods used for identifying common quality problems and relationships among those problems. As described earlier in Section 2, use case models comprising of use case diagrams and use case descriptions, static models represented by class diagrams and dynamic models represented by sequence diagram form the focus of this study. Artifacts from a set of 15 semester-long projects completed by teams of students of an object-oriented analysis and design course were used for this purpose. This set of artifacts is a good representative of typical models created by novice analysts because (a) the final year undergraduate students have had prior experience with structured systems analysis and design method in a prerequisite course, and (b) the students work in teams of size 3 or 4 on semester-long projects which involved gathering requirements of different types of business applications.

Each student team worked on a project covering business applications such as banking, hotel reservations, movie ticketing, and airline reservations. The project topics were randomly assigned at the beginning of the course and project proposals developed by student teams describing the application scope were reviewed and approved by the instructor. This process has ensured that the projects were of comparable complexity in terms of the modeling skills and effort required. All the teams have used MS-Visio for drawing UML diagrams and MS-Word for writing use case descriptions using a template provided for this purpose. On average, the use case diagrams included about 6 actors and 16 use cases with 3 or 4 important use cases described in detail. The class diagrams, on average, included 14 classes and up to 50 attributes and 23 operations across all classes. The sequence diagrams, corresponding to the use case descriptions, included 6 objects and 14 messages on average.

Evaluation of various artifacts using the framework of Lindland et al (1994) required developing a detailed coding scheme containing errors belonging to different quality categories. Two researchers, the course instructor (the author) and an experienced tutor (a Ph. D. candidate), were involved in developing the coding scheme and the subsequent identification of errors in each project. Two copies of each project report were used by these researchers to independently prepare a list of errors observed and then categorize those errors into 5 groups: syntactic, semantic – validity; semantic correctness; pragmatic – expected missing; pragmatic – unexpected is present. The researchers then consolidated the two sets of categorized errors and described the errors. The final coding scheme has 13, 14, 35 and 23 error codes for use case diagrams, use case descriptions, class diagrams and sequence diagrams respectively.

Prior to coding the set of artifacts, the researchers using one of the project reports tested the coding scheme, and that project report was excluded from the rest of the analysis. This initial testing has helped in achieving a better understanding of the

coding process and final set of refinements to the coding scheme. The researchers independently examined each artifact of 14 remaining projects, on separate copies, for presence of errors from the coding scheme. Only the first occurrence of each type of error was noted and any multiple occurrences of the same error belonging to the same project were ignored. Then a process of verification was performed, again independently, after exchanging the lists of errors identified by each researcher in each project report. A total of 380 errors were identified in the 14 projects with an overall inter-rater agreement of 75% after this verification. Considering the large number of possible error codes (85) in the coding scheme, the complexity of the highly subjective process of finding those errors in various artifacts and the exclusion of errors not identified from the calculation of inter-rater agreement, this appeared to be an acceptable inter-rater agreement. However, to reach a complete inter-rater agreement the researchers discussed and resolved the remaining differences. This process has finally resulted in 100% inter-rater agreement with a total of 350 errors in 14 project reports. Developing the coding scheme and identifying errors in various artifacts is a highly interpretive and subjective process. By taking elaborate precautions and multiple rounds of independent coding by experienced researchers, the effect of subjectivity has been minimized.

Intra- and inter-artifact quality dependencies were identified based on specific errors that occurred together in groups or clusters. Since many syntactic errors could be eliminated using CASE tools, only the semantic and pragmatic errors were considered in this analysis. To identify how errors within and across different types of artifacts are related, PermuCLUSTER (Spaans and van der Kloot, 2004), a hierarchical clustering technique implemented as an add-on to SPSS, was used. PermuCLUSTER is claimed to deal with the phenomenon of input order instability, which makes the cluster solutions dependent on order of rows and columns of proximity matrix, by repeating the cluster analysis by permuting the rows and columns. In addition this technique offered two goodness-of-fit measures (normalized sum of squared differences and cophenetic correlation coefficient) for selecting the optimal clustering solution.

FINDINGS

In this section, we first present descriptive statistics obtained from the errors identified by the two researchers and then present the dependencies observed among errors within and across the three types of artifacts addressed in this study. The total numbers of errors observed under various categories for each type of artifact are presented in Table 1. Relative differences in these error counts indicate differences in difficulties associated with creating corresponding artifacts.

	Syntactic	Semantic	Pragmatic	Total
Use Case Diagrams and Descriptions	29	62	56	147
Class Diagram	26	32	33	91
Sequence Diagram	38	41	33	112
Total	93	135	122	350

Table 1. Numbers of errors observed in different artifact types

Analysis of number of errors of three quality categories for all artifacts of various projects indicated significant correlations between total number of semantic errors and the total errors (0.683 significant at the 0.01 level – 2-tailed), and between the total number of syntactic errors and total errors (0.631 significant at the 0.05 level – 2-tailed) were observed. Finally, the total numbers of semantic and pragmatic errors are negatively correlated (though not significant).

Intra-artifact quality

Specific errors that have been frequently observed in different types of artifacts along with frequency of occurrence are listed in Table 2. Commonly committed errors in use case descriptions are mostly associated with step descriptions. Most errors in class diagrams are related to the specification of operations. And, errors related to flow control (structuring) are frequent in sequence diagrams.

Use case diagrams and descriptions. Frequent errors observed in use case descriptions are largely related to the description of steps in use cases. On the other hand, frequent errors observed in use case diagrams are related to identification of proper

relationships (viz. includes, extends and generalization) between use cases. Excessive use of low level steps in use case descriptions is possibly due to the tendency of team members in beefing up the use case description without attempting to acquire or gather sufficient domain knowledge pertaining to the underlying use case scenario. It appears that the novice analysts have tried to compensate for their lack of domain expertise by introducing unnecessary steps or splitting simple steps.

Artifact	Error Code	Error Description	Frequency
Use case diagrams and descriptions	UPu3e	Excessive use of manual step(s) in use case description	10
	USy2b	Improper notation in use case diagram	9
	USy4	Use case name mismatch between diagram and description	9
	USc1c	Missing step in use case description	9
	USc2a	Ambiguous step description	9
	UPu3f	Excessive implementation details (ui; database; programming style)	9
	USv1b	Invalid relationship between use cases	9
	USv1f	Invalid extension(s) to a step	8
	UPu1	Manual operations are listed as use cases	8
	UPu3d	Excessive splitting of step(s) in use case description	8
Class diagrams	CSy4	Non-implicit operations present in sequence diagram are not included	9
	CSy5	Implicit operations are listed	7
	CSv1b	Wrong range for association cardinality (or multiplicity)	7
	CPe3	Insufficient distinction among subclasses in a generalization hierarchy	6
	CSv2a	Wrong location of attribute(s)	5
	CSv2b	Wrong location of operation(s)	5
	CSc3	Operation cannot be realized using existing attributes and relationships	5
	CPu4	Derived (or redundant) attribute	5
Sequence diagrams	SSy1a	Improper flow control - initial trigger is missing	11
	SSc2	Message parameters are missing (completely or occasionally)	10
	SSy1c	Improper flow control - return to an object different from calling object	9
	SSv3	Message parameters are used before their values are available	9
	SPu1a	Improper delegation of responsibility to a wrong object	8
	SSy5	Object identifier does not belong to the class diagram	7
	SSc1	Some essential (required) classes/objects left out of sequence diagram	7

Table 2. Frequently observed errors in three artifact types

Class diagrams. The team members' prior experience with entity-relationship modeling appears to have contributed to the overall quality in different ways. Many errors were related to identifying operations and assigning those operations to

appropriate classes. Errors related to association specification especially the cardinality details (wrong range of values) were also observed frequently. The majority of errors observed under pragmatic quality category (e.g., derived or redundant attribute) can be attributed to prior experience of these analysts with data base design and implementation. We have also noticed instances where the subclasses in class hierarchies having insufficient distinction which may be either due to the urge to use this feature or due to lack of depth in requirements specified in use cases.

	Semantic	**Pragmatic**
Use case diagrams and descriptions	USv1b - Invalid relationship between use cases USv1f - Invalid extension(s) to a step USc2a - Ambiguous step description USc1c - Missing step in use case description	UPu3e - Excessive use of manual step(s) in use case description UPu3f - Excessive implementation details (UI; database; programming style) UPu3d - Excessive splitting of step(s) in use case description UPu1 - Manual operations are listed as use cases
Class diagrams	CSv1b - Wrong range for association cardinality (or multiplicity) CSc1c - One or more major operation(s) are missing CSc1b -One or more major attribute(s) are missing CSc3 - Operation cannot be realized using existing attributes and relationships	CPe3 - Insufficient distinction among subclasses in a generalization hierarchy CPu2a - Excessive use of generalization(s) CPu2b - Excessive use of primary key concept
Sequence diagrams	SSv3 - Message parameters are used before their values are available SSv1 - Wrong target (class/object) for a message SSc2 - Message parameters are missing (completely or occasionally) SSc1 - Some essential (required) classes/objects left out of sequence diagram SSc5 - Operation cannot be realized using attributes and/or links of an object SSc3 - One or more missing iterations	SPe4 - Poor structuring of actions (particularly sequence & iteration) SPu1a - Improper delegation of responsibility to a wrong object

Table 3. Error Clusters Affecting Intra-Artifact Quality

Sequence diagrams. The majority of errors can be attributed to the inexperience of novice analysts in problem-structuring skills such as decomposition and to difficulties in applying object-oriented concepts related to distribution of responsibilities across participating objects. Many syntactic errors are related to message flow control such as missing initial trigger messages, returning control to objects other than the calling object, etc. Pragmatic errors included improper delegation of responsibility (often to a wrong object/class) and/or making a class/object perform computations that can be delegated to other objects.

Table 3 shows the three optimal clusters of errors (selected based on the cophenetic correlation coefficient calculated by PermuCLUSTER) affecting quality of the three types of artifacts. Analyzing the errors in the first cluster, it can be noticed that quality of use case step descriptions contributes significantly to the overall quality of use case models. The second cluster indicates the importance of proper understanding and application of the concepts of object-orientation and capturing various types of relationships (especially generalization-specialization) in developing quality class diagrams. The cluster corresponding to sequence diagrams highlights the importance of problem structuring skills and object-oriented concepts related to delegation of responsibility to appropriate objects. These three error clusters can be used for both addressing

problems in teaching and learning of techniques for modeling associated artifacts, and for implementing intelligent support mechanisms into CASE tools.

Inter-artifact quality

Three optimal clusters, one for each pair of artifact types, have also been identified (Table 4). First two of these clusters, contrary to expectations, do not provide any interesting patterns related to quality dependencies. However, the third cluster highlights the dependence between use case models and dynamic models. Since sequence diagrams are often created using corresponding use case descriptions, any problems in use case descriptions (particularly step descriptions) directly affect the quality of sequence diagrams.

	Semantic	**Pragmatic**
Use case diagrams and descriptions vs. Class diagrams	USv1b - Invalid relationship between use cases USv1f - Invalid extension(s) to a step CSv1b - Wrong range for association cardinality (or multiplicity) USc1c - Missing step in use case description USc2a - Ambiguous step description	CPe3 - Insufficient distinction among subclasses in a generalization hierarchy UPu3e - Excessive use of manual step(s) in use case description UPu3f - Excessive implementation details (user interface; database; programming style) UPu3d - Excessive splitting of step(s) in use case description
Class diagrams vs. Sequence diagrams	CSv1b - Wrong range for association cardinality (or multiplicity) SSv1 - Wrong target (class/object) for a message SSv3 - Message parameters are used before they are available SSc1 - Some essential (required) classes/objects left out of sequence diagram SSc2 - Message parameters are missing (completely or occasionally)	SPu1a - Improper delegation of responsibility to a wrong object
Use case diagrams and descriptions vs. Sequence diagrams	USv1b - Invalid relationship between use cases USv1f - Invalid extension(s) to a step SSv3 - Message parameters are used before their values are available USc1c - Missing step in use case description USc2a - Ambiguous step description SSc2 - Message parameters are missing (completely or occasionally)	UPu3e - Excessive use of manual step(s) in use case description UPu3f - Excessive implementation details (user interface; database; programming style) UPu3d - Excessive splitting of step(s) in use case description SPu1a - Improper delegation of responsibility to a wrong object

Table 4. Error Clusters Affecting Inter-Artifact Quality

CONCLUSION

An application of Lindland et al's conceptual model quality framework for analyzing the quality of UML artifacts and for identifying intra- and inter-artifact quality dependencies is presented in this paper. By training novice systems analysts with methods and techniques to minimize such errors, we can expect better quality models representing system requirements from

them. Although this framework has been applied for identifying quality problems in different types of UML artifacts, using two experienced researchers for identifying errors in each artifact has minimized any possible bias. While most previous applications of Lindland et al's framework have focused mostly on entity-relationship, extended entity-relationship or object models, this research has applied it for three different types of UML artifacts. During this application, some difficulties were observed especially in classifying certain errors into semantic and pragmatic categories (e.g., operation cannot be realized using existing attributes and relationships; wrong location of operations; ambiguous step description).

Based on the intra-artifact quality dependencies identified in this study, the following observations can be made: (a) by gathering required domain knowledge novice systems analysts can create quality step descriptions which in turn contribute towards creating quality use case models, (b) although prior experience with entity-relationship modeling is useful for developing quality domain models (class diagrams), it is essential for novice analysts to understand the application of object-oriented concepts particularly those related to generalization and identifying operations, and (c) skills related to problem structuring and assigning operations (particularly delegation of responsibilities) to proper objects are necessary for developing quality dynamic models (interaction diagrams).

The frequently observed errors and the clusters or errors in different types of UML artifacts shed light on specific difficulties encountered by novice systems analysts and on the associated training requirements. Training programs designed specifically to meet these requirements would be helpful in enhancing teaching and learning effectiveness of systems analysis to students.

Using projects completed by teams of students, though these students had prior experience with structured systems analysis and design method, is a major limitation of this study. Considering the difficulties in accessing the work produced by experienced systems analysts in real life projects, the limited experience of subjects and semester-long duration of team projects may be considered a suitable alternative. Another limitation of the study is related to types of artifacts considered. Though some of the widely used UML artifacts are considered in this study, other artifacts such as activity diagrams and collaboration diagrams may be relevant for such studies. Finally, only one type of audience (i.e., experienced analysts) was considered for evaluating the quality of artifacts. The conceptual framework of Lindland et al (1994) suggests using different types of audiences such as end-users, project managers, designers and programmers.

Further research investigations and studies are required for identifying causal relationships among various types of errors identified in different types of UML artifacts, for incorporating some of these findings into CASE tools to provide intelligent and adaptive support to novice systems analysts, and for identifying the effect of quality of analysis artifacts on that of design artifacts.

ACKNOWLEDGMENTS

Funding for this research was received from the Strategic Research Grant # 7001871 of City University of Hong Kong.

REFERENCES

1. Adolph, S. and Bramble, P. (2003) *Patterns for Effective Use Cases*, Boston: Addison Wesley.
2. Agarwal, R. and Sinha, A.P. (2003) Object-oriented modeling with UML: A study of developers' perceptions. *Communications of the ACM*, 46(9), 248-256.
3. Ambler, S.W. (2003) *The Elements of UML Style*. Cambridge University Press.
4. Chaos Report (1994), The Standish Group. (http://www.standishgroup.com/sample_research/chaos_1994_1.php)
5. Cockburn, A. (2001) *Writing Effective Use Cases*, Addison-Wesley.
6. Fowler, M. (1997) *Analysis Patterns: Reusable Object Models*. Addison Wesley, Menlo Park, CA.
7. Genero, M., and Piattini, M. G. (2002) Quality of Conceptual Modelling in *Information and Database Quality* by Piattini, M.G., Calero, C., and Genero, M. (eds), Kluwer Academic, 13-44.
8. Krogstie, J., Lindland, O.I., and Sindre, G. (1995) Towards a Deeper Understanding of Quality in Requirements Engineering. *Proceedings of the 7th International Conference on Advanced Information Systems Engineering*, Finland, 82-95.
9. Lindland, O. I., G. Sindre, and Sølvberg, A. (1994) Understanding quality in conceptual modeling. *IEEE Software*, 11(2), 42-49.

10. Moody, D.L., Sindre, G., Brasethvik, T., and Sølvberg, A. (2003) Evaluating the Quality of Information Models: Empirical Testing of a Conceptual Model Quality Framework. *Proceedings of the 25th International Conference on Software Engineering*, 295-305.
11. Schenk, K.D., Vitalari, N.P., and Shannon Davis, K. (1998) Differences Between Novice and Expert Systems Analysts: What Do We Know and What Do We Do?. *Journal of Management Information Systems*, 15(1), 9-50.
12. Shanks, G., Tansley, E., and Weber, R. (2003) Using ontology to validate conceptual models, *Communications of the ACM*, 46(10), 85-89.
13. Siau, K. and Cao, Q. (2001) Unified Modeling Language: A Complexity Analysis, *Journal of Database Management*, 12(1), 26-34.
14. Spaans, A. and van der Kloot, W, (2004) PermuCLUSTER 1.0 User Guide. www.leidenuniv.nl/fsw/mtlab/software/PermuCluster%201.0%20User%20Guide.pdf (Accessed June 2005).
15. Su, X. and Ilebrekke, L. (2005) Using a Semiotic Framework for a Comparative Study of Ontology Languages and Tools, *Information Modeling Methods and Methodologies*. Krogstie, J., Halpin, T.A. and Siau, K. (eds). IDEA Group. 278-299.
16. UML (2004) Introduction to OMG's Unified Modeling Language™ (UML®). http://www.omg.org/gettingstarted/what_is_uml.htm
17. Wand, Y and Weber, R. (2002) Research commentary: Information systems and conceptual modeling - A research agenda. *Information Systems Research*, 13(4), 363-376.