Class Diagrams and Use Cases - Experimental Examination of the Preferred Order of Modeling

Peretz Shoval*, Avi Yampolsky and Mark Last

Dept. of Information Systems Engineering
Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
*Shoval@bgu.ac.il http://www.ise.bgu.ac.il/faculty/shoval

Abstract. In most UML-based methodologies, the analysis tasks include mainly modeling the functional requirements using *use cases*, and modeling the problem domain using a *class diagram*. Different methodologies prescribe different orders of carrying out these tasks, and there is no commonly agreed order for performing them. In order to find out whether the order of these analysis activities makes any difference, and which order leads to better results, we carried out a comparative experiment. Subjects were asked to create the two analysis models for a certain system in two opposite orders, and the qualities of the produced models were then compared. The results of the experiment reveal that the class diagram is of better quality when created as the first modeling task, while no significant effect of the analysis order was found on the quality of the use cases. We also found out that analysts prefer starting the analysis with data modeling.

1 Introduction and Related Studies

In early development methodologies of the 70's (e.g. [7], [24]) the emphasis of analysis was on describing the functional requirements by conducting a functional decomposition of the systems using data flow diagrams (DFDs) or similar techniques. Later on, with the introduction of conceptual data models, many methodologies included also data modeling in the analysis phase (e.g. [23]). Nevertheless, functional analysis was still the primary task and data analysis was only secondary to it.

In the early 90's new, object-oriented (OO) development methodologies emerged (e.g. [5], [17] and [18]). In OO methodologies, the analysis phase emphasize on finding the domain objects, while the design phase emphasize on identifying of the services that the objects ought to provide and assigning responsibilities to them. These objects are not necessarily the ones from which the system is eventually built, but the principal entities (data objects) from the problem domain. Since this domain object model and the conceptual data diagram used in earlier methodologies are much alike, the analysis tasks eventually remained akin, whereas the order of performing them was inverted. However, there were still methodologies that kept with the functional-driven approach, by which a functional analysis is performed first and then the object model is derived from it, whether directly or through a conceptual data model. See for example [10].

Aiming at solving the problems raised by many of OO methods and tools, Object Management Group adopted UML (Unified Modeling Language) as its standard for modeling OO systems [4]. The UML techniques for describing the user/functional requirements and the object model are *use cases* and *class diagram*, respectively. UML's class diagram is an enhanced variation of common data models that have been used over the years (notably Entity-Relationship model). Use case is a piece of the system's functionality, describing the possible interactions between the system and a user entity external to it called "actor", for the purpose of achieving a goal of that actor. Use cases describe the system as a "black box", meaning that the internal structure of the system and its internal operations are not described. That is why this model is most appropriate to use during the analysis phase. Note that a use case is not only represented in a diagram; its details are described in a semi-structured format.

Many development methodologies which use the UML models/tools have been developed over the last decade. Despite many variations between different UML-based methodologies, in most of them the analysis phase comprises two main activities: data modeling, i.e., creating a class diagram to describe the application domain; and functional modeling, i.e., creating use case diagrams and descriptions to describe the functional requirements and the users' interactions with the system. UML-based methodologies which adopt use cases as the requirements description tool are usually "use case driven", meaning that the entire development process is derived by describing, realizing and developing use case scenarios.

The Rational Unified Process (UP) [9], [15] is one of the most common use case driven methodologies. It provides a wide and general framework for systems development, and as such offers guidelines with many optional variations. The first analysis task according to UP is creating a use case model, whilst an initial class diagram is only created in the next phase called Use Case Analysis. Since UP is a most commonly used methodology, in industry it is common that users spend a great deal of effort in conducting use case analysis aimed at identifying the business requirements and systems requirements at a high level, while class diagrams are seen as more closely associated with system design and implementation - therefore often delayed till a reasonable use case analysis is done. Larman [12] applies the UP methodology in an iterative process, and suggests starting with an initial use case model, stating the names of the use cases in the systems and describing only the most important and risky ones; then continuing with analysis-design-development iterations. In each of the analysis phase iterations, the use cases are detailed and a semantic class diagram, called Domain Model, is created. The concepts for the domain model are identified from the nouns in the use cases descriptions.

Contrary to the above examples, some UML-based methodologies suggest starting the analysis process with data modeling. For example, ICONIX [16] suggests starting with creating a class diagram describing the real world entities and concepts in the problem domain, using the name Domain Model for this preliminary class diagram. According to the authors, the general class diagram, which describes the domain and not a specific solution, is an important basis and a glossary for creating use cases that describe the functional requirements. In fact, ICONIX is the only UML-based process we found that actually discusses the issue of analysis order and argues that it is better to create a domain model before detailing the functional requirements.

While some methodologies advocate starting the analysis with functional modeling, and other – with class modeling, Maciaszek [13] claims that there is no particular order for creating the use cases and class diagram, as these two activities are done simultaneously and are feeding one another. However, since the analysis should start somehow, he eventually leaves the decision in the hands of the analyst.

Dobing & Parsons [8] investigated the role of use cases in UML and identified several problems with both the application and the theoretical underpinnings of use cases. In their opinion, the roles and values of the use case diagram are unclear and debatable. Moreover, they state that the process for moving forward from use case diagrams to class identification is neither universally accepted, even among use case adherents, nor does it appear to be clearly defined or articulated.

The above argument is partially supported by Siau & Lee [21] who examined the values of use case diagrams in interpreting requirements when used in conjunction with a class diagram. They found out that the interpretation of a sequence combination of use cases and class diagrams have no effect on the problem domain understanding; and assert that (given there is no significant difference between the sequences of interpretations) the order in which the diagrams are used or constructed during the requirements analysis may not be important. They suggest that both diagrams may need to be constructed concurrently and modified interactively. It should be noted, however, that [21] only examined user comprehension of the diagrams, not the quality of their construction, and that they considered only use case diagrams, but not the use case descriptions.

Shoval & Kabeli [20] is the only research we are aware of which dealt explicitly with the order of conducting these modeling tasks. They describe an experiment made to compare the two orders of modeling tasks using FOOM methodology [19]. The participants were given a requirements document of a certain system and were asked to perform the two modeling tasks according to that methodology, i.e., to create a class diagram modeling the data requirements, and OO-DFDs[1] modeling the functional requirements. The experiment revealed that starting the analysis with data modeling results in better class diagrams; yet no significant differences were obtained regarding the quality of the resulting functional models. In addition, the participants were asked about what they think is the better order of analysis activities; they preferred starting with creating a class diagram rather than OO-DFDs.

2 Research Goals and Hypotheses

The main goal of this research is to examine the order of performing the two main modeling tasks in the analysis phase of UML-based methodologies: functional modeling with use cases and data modeling with class diagrams. It is agreed that system analysis is an iterative process of refinement, not a linear sequence of activities. Still, the analysis must begin with a specific activity, so it is legitimate and important to examine whether the order matters and, if yes, which order is better. As we have seen, some methodologies prescribe to start with creating a class diagram and continue with

[1] OO-DFD, Object-Oriented DFD, is a variation of DFD which include object classes rather than data-stores.

use cases using the concepts identified in the class diagram; other prescribe to start with creating use cases and continue with a class diagram based on the concepts appearing in the use cases.

Methodologies starting with creating a class diagram argue that the initial class diagram maps the problem domain and allows describing the functional requirements within a well defined context. They claim that the entities in the class diagram serve as an essential glossary for describing the functional requirements and, since it is an abstraction of the part of the real-world relevant for the system, it only rarely changes and can serve as a solid basis for other future systems as well. On the other hand, methodologies starting with creating use cases argue that the classes should only be described after the functional requirements, and be elicited from them.

We expect that creating a class diagram prior to defining the functional requirements with use cases should yield better results, i.e. better class diagrams and better use cases. This is because objects are more “tangible”/"stable" than use cases; users can identify and describe more easily the objects they are dealing with and their attributes than functions or use cases of the sought system. On the other hand, functions are not “tangible” and may be vague. Different users may define differently what they expect the system to do for them. At any rate, users do not express their needs in terms of use cases. Of course, the task of data modeling is not trivial either; it is not always clear what is an object, how to classify objects into classes, what are the attributes and the relationships, etc. - still, the task of data modeling seems to be more structured and less complex compared to the task of modeling use cases. Besides, in data modeling the analyst has to create just one class diagram, while functional modeling involves many use cases. Note also that while in data modeling the analyst concentrates only on the data related aspects, in use case modeling the analyst actually deals at the same time with more aspects, because uses cases are not merely about functions; they are also about data, user-system interaction and the process logic of the use cases. Because of the above, it seems to us that starting the analysis process with a simpler and more structured task would be more efficient (in terms of analysis time) and effective (in terms of quality of the analysis products). Not only that the first model (the class diagram) is expected to be of good quality, it would also ease the creation of the following model (the uses cases) because at this stage the task would seem to be less complex. Hence, we also expect that analysts would prefer to work in that order, i.e. first create a class diagram and then use cases.

The above expectations and assumptions are supported by previous research. We have already referred to the experiment conducted by Shoval & Kabeli [20] who dealt with the same issue but in the context of another development methodology. According to that experiment, analysts who start the analysis process with data modeling produce better class diagrams than those who start the process with functional modeling, and they prefer working in this order. The current study can be viewed as a continuation of that one, but using UML tools.

Based on the above discussion and previous results, we expect that starting with data modeling would yield better models. However, as we have seen in the survey of Section 1, there are in fact different methodologies which advocate different orders of activities. Therefore, for the sake of this study, we hypothesize that there is no difference in the quality of the analysis models when created in either of the opposing or-

ders. Similarly, we hypothesize that there is no difference in the analysts' preference of the order of activities.

3 The Experiment

To examine whether there is a preferred order for performing the analysis tasks, we carried out a comparative experiment. In order to simulate the analysis process, we provided the participants with a requirements document describing the various requirements of a certain system, for which each participant was asked to create use cases and a class diagram.

3.1 The Research Model

Most experiments aimed at evaluating analysis products refer to data models only, and use a research model for evaluating user performance that identifies the effect of three factors and the interaction between them: the data model being used, the task characteristics and the human characteristics. A review of such studies is provided in [22]. Following the above research model, Figure 1 describes the research model of our experiment.

3.2 The Dependent Variables

The main dependent variable we used to evaluate the analysts' performance is ***quality of models***. Model quality was measured using a grading scheme that represents the correctness of the analysis artifacts. The grading scheme for each model will be described below. In addition, we asked the subjects about their subjective **preferences** regarding the better order of analysis, using a 7-point ordinal scale.

3.3 The Independent Variable

Our main interest is the order of creating the two analysis models: a class diagram to describe the problem domain, and use cases to describe the functional requirements. The ***analysis order*** is therefore the independent variable. Based on that, we created two treatment groups, as shown in Table 1.

Table 1. The Treatment Groups

Group	Analysis Order
Group A	1) Class Diagram; 2) Use Cases
Group B	1) Use Cases; 2) Class Diagram

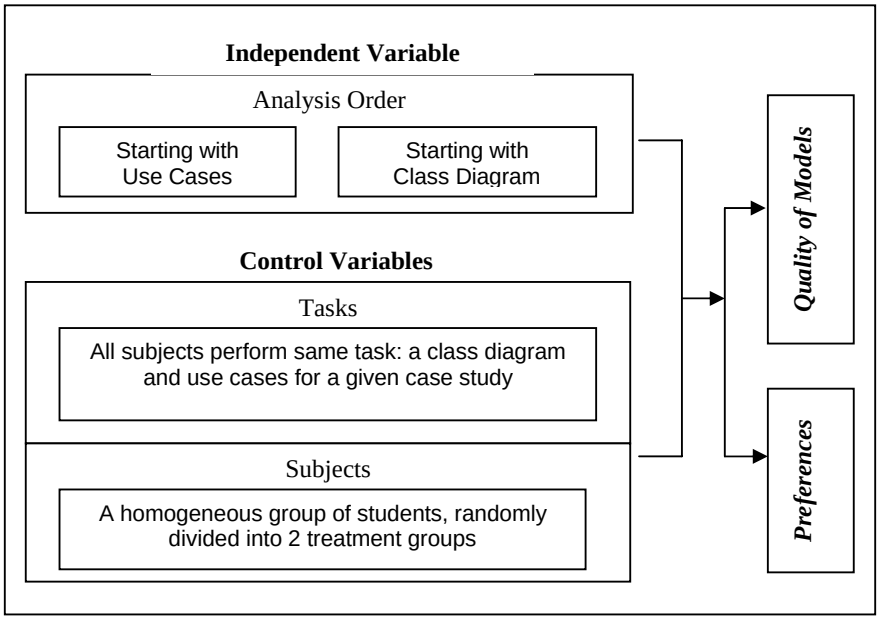

Fig. 1. The Research Model

3.4 The Control Variable

Two control variables are identified in the model: the **tasks** and the **subjects:**

- **Tasks**: As the experiment task we chose to use the *IFIP Conference* case study [14] that was also used in [20]. This problem is of manageable size and can be solved fairly easily. Furthermore, using the same case study as [20] would strengthen the validity of the results of the two experiments.

  In reality, analysts interact with users, elicit their requirements and based on that create the analysis models. However, in an experiment we create an artificial environment: instead of a real task and real users interacting with the analysts, we prepared a case study in the form of a requirements document. Such a document must, of course, include both data-related and functional-related requirements, but this may raise a problem because the order of presentation of the two types of requirements in the document may affect the quality of the models created by the analysts. To avoid possible bias due to this effect, we prepared two versions of the (same) requirements document: one version presenting the data-related requirements first and then the functional-related requirements; and other version presenting the requirements in the opposite order. The two versions of the requirements document were randomly distributed to the subjects within the two groups.
- **Subjects**: The subjects were senior undergraduate students of Information Systems at Ben-Gurion University. We performed the experiment as a mid-term exam in the OO Analysis and Design course. During the course, the participants learned

the OO analysis approach and UML, including use cases and class diagrams. Having a homogeneous group of subjects, i.e., students in the same class who took the same courses and where trained by the same instructor, allows us controlling potential biases such as differences in individual characteristics, analysis skills and task-related experience. Anyhow, the subjects have been assigned randomly to the two treatment groups.

To control the training variable and to direct the subjects toward a unified style of describing use cases, an extra hour and a half tutorial was conducted during which Cockburn's [6] use case writing guidelines were taught and a sample exercise was solved. In addition, we handed the subjects a solution for one of the use cases as an example for the way in which they are expected to describe the use cases. To motivate the subjects to perform the tasks as good as possible, their grades in the experiment were considered as a part of the final course grade.

3.5 The Grading Schemes

The qualities of the analysis models were measured using grading schemes, which list the possible error types and the number of points to deduct for each error type. Grading schemes have been used in previous studies for measuring quality of models (e.g., [3] and [11]). Table 2 presents the grading scheme for the class diagram.

Table 2. Grading Scheme of the Class Diagram

Element	Error	Points deducted
Class	Missing class	6
	Superfluous class	2
	Incorrect class type	1
Attribute	Missing attribute or attribute in the wrong class	2
	Superfluous attribute	1
Relationship	Missing relationship	4
	Erroneous relationship	3
	Superfluous relationship	3
	Incorrect relationship type	2
Relationship multiplicity	Missing or incorrect multiplicity	1
Inheritance	Missing inheritance	6
	Ordinary relationship instead of inheritance	2
	Superfluous inheritance	2

Since class diagram is well defined tool with strict structure and syntax, mapping the possible errors in it is straightforward. Use cases, on the other hand, are less structured and described using free text. Mapping the possible errors in use cases requires defining the components that an analyst is expected to describe. Assisted by the use case error mapping in [1], which is also based on Cockburn's approach [6], we identified the following three components:

- Actor: the external entity owning the goal of executing the use case.
- User goal: the goal that the use case has to achieve. Each use case has to achieve one user goal.
- Sub-goal: a user goal is a collection of sub-goals that are steps in accomplishing the user goal.

After identifying the components, as in the class diagram, we mapped the possible error types in each component, and determined the error-points. Table 3 presents the grading scheme for the use cases.

Table 3. Grading Scheme of the Use Cases

Element	Error	Points deducted
Actor	Incorrect actor	4
	Inconsistent actor	2
User goal	Missing goal	10
	Goal appears in title only	6
	Goal described as a part of other goal	4
	Superfluous goal	2
Sub-goal	Missing sub-goal	3
	Superfluous or erroneous sub-goal	2

In addition to these "semantic errors" [2], we identified the following "syntactic errors" and assigned them altogether six deduction-points: irrational solution, untidiness and lack of logical order, unclear script and inconsistent description.[2]

4 Results

The experiment was conducted in a controlled environment and in an exam format. The exam was taken by 121 students; it was planned to take two hours, but an extension of half hour was granted to allow the participants to complete their tasks.

We wanted to investigate the effect caused by the independent variable on the dependent variable - Quality (grade). Since grade is a continuous variable, and the independent variable has discrete levels, the suitable statistical test is two-way t-test. The analysts' preferences were tested using Wilcoxon test, a non-parametric test that allows testing results from an ordinal scale without imposing any other restrictions on the data samples.

4.1 Quality of Models

The two analysis models are of different nature and require different grading schemes to evaluate, which makes them incomparable. We hence compared the quality of **each model separately**. The null hypothesis for the statistical tests is that there is no dif-

[2] There might be some subjectivity in the above grading schemes. This limitation will be elaborated in the Summary section. Note, however, that we applied a specific grading scheme for each model separately, and we did not combine or compare the results across the two models.

ference between the values of the dependent variable (quality of models) for the different values of the independent variable tested.

Table 4 presents the results of the quality of the **class diagrams**. The 1st column presents the two values of the independent variable Analysis Order whose effect on the dependent variable is examined; the 2nd column (N) is the number of subjects in the group; the 3rd is the mean grade of the class diagram; the 4th is the **t** statistic of the independent variable; the 5th is the p-value; and the last column indicates if the difference between the results is significant.

As can be seen, the grades are significantly higher when starting the analysis with class diagram (73.63) compared to when starting with use cases (70.25). Note that these results are consistent with the results obtained in the previous experiment [20].

Table 4. Quality of Class Diagrams

Analysis Order	N	Mean grade (%)	t	p-value	Significance in favor of
1) Class Diagram; 2) Use Cases	57	**73.63**	4.386	.038	Starting with class diagram
1) Use Cases; 2) Class Diagram	64	**70.25**			

Table 5 presents the results of the quality of the **use cases**. As can be seen, there are no significant differences between the two analysis orders. Note again that these results are consistent with the results obtained in [20] for the functional models.

Table 5. Quality of Use Cases

Factor value	N	Mean grade (%)	F	p-value	Significance in favor of
1) Class Diagram; 2) Use Cases	**57**	**63.72**	**.192**	**.662**	-
1) Use Cases; 2) Class Diagram	64	65.03			

Looking at the grades of the class diagrams (Table 4) and the use cases (Table 5), we see that the use case grades were, in average, lower than those of the class diagrams. This may be explained by several factors: A) Different grading schemes: it is possible that because of the grading schemes and the number of points deducted per error type, more points were deducted due to errors in uses cases, comparing to errors in the class diagrams. B) Task complexity: as already discussed, use case modeling seems to be a more complex task than class modeling; disregarding the order they are worked out. C) Model formality: in line with the former discussion, we have seen that a class diagram is well structured and has clear and simple syntax, while a use case is less structured. Lack of well-defined syntax increases the frequency of errors caused by misunderstanding the required task. At any rate, as said, we made no attempt to combine or compare the results of the two models; we only compared the differences between the results within each model. Therefore, the above differences in the grades across the different models do not bias our results.

4.2 Analysts' Preferences

After the experiment each subject was asked to express to what degree he/she believes that the order of analysis used is good/appropriate using a 1-7 point scale, where 1 means total preference to start with class diagram, 4 means indifference, and 7 means total preference to start with use cases[3]. Table 6 presents the results, for each group of subjects and for all together.

Table 6. Analysts' Preferences

The order in which the subjects worked	N[4]	Mean preference	Standard deviation
1) Class Diagram; 2) Use Cases	22	2.91	1.54
1) Use Cases; 2) Class Diagram	18	2.61	1.82
All together	**40**	**2.78**	**1.66**

The results show that the subjects definitely believe that it is better to first create a class diagram and then use cases (mean preference of all is 2.78; much closer to 1 than to 7). It is interesting to see that the subjects who started the analysis with use cases showed even stronger preference to start with creating a class diagram (2.61 compared to 2.91). The preference towards an order of analysis starting with a class diagram matches both our hypothesis regarding analysts' preferences and the results obtained in the earlier experiment [20].

5 Summary and Further Research

The principal purpose of this research was to compare two interchangeable orders of performing the main analysis tasks in a use case-driven approach: creating a class diagram to model the problem domain, and creating use cases to describe the functional requirements of the system. The results of the experiment reveal that starting the analysis by creating a class diagram leads to a better class diagram. Nevertheless, we did not find a significant effect of the order of analysis on the quality of the use cases.

Interestingly, the results we obtained in this experiment are consistent with those obtained in an earlier experiment with a different group of subjects [20] where a variation of the same requirements document was used, but utilizing a somewhat different class diagram notation, and OO-DFDs instead of use cases to model the functional requirements. It appears that the conclusions with respect to the preferred order of analysis activities hold irrespectively of the analysis methodology.

Like other comparative experiments which compare methods and models in a laboratory setting, this one too has limitations that may question its external validity. For a discussion on common limitations of such experiments see [22]. An obvious limitation is that we used a relatively small and simple problem while in reality problems

[3] Recall that each participant performed the tasks according to one order only, so he/she could only express his subjective preference based on the task he/she performed.

[4] Only 40 participants replied to this question.

are much bigger and more complex; we cannot be sure how size and complexity of a system would affect the results with respect to the order of analysis activities.

Another limitation which hampers the external validity of our results is that they are based on one case study, the *IFIP Conference* system, which may represent only data-driven (or MIS) information systems. We cannot be sure if the results are also valid for other types of systems (e.g. real-time systems). Noting that the results of this study are consistent with the results of the earlier study [20] which used the same *IFIP Conference* system, we may be more confident that the results are valid for data-driven systems, but not necessarily for other.

Of course, there is the limitation of using students with almost no industrial experience as surrogates for analysts. This limitation is common to almost all experimental work published in the area [22]. We cannot predict if and how the cumulative experience of analysts might affect the preferred order of analysis activities.

Another potential confound of the experiment is the grading schemes. Some subjectivity may exist in the weights given to errors (in points) as described in Tables 2 and 3. We determined the weights based on our assessment of the importance of each error type. As said, in doing so we followed earlier studies that also adopted subjective grading schemes to assess quality of models. The potential problem with such grading schemes is that the subjective weights (points) assigned to the identified error types may affect the overall results. The problem is that there are no objective weights and grading schemes for different methods or models. This issue deserves separate research.

Being it a "laboratory" experiment, we used a requirements document to represent the real-world and the users' needs; we actually forced a one-way modeling process, where the analyst/subject reads a given requirements document and creates from it the analysis models to the best of his/her understanding. This is not the way a really analysis task is performed. In reality, the analysis process involves a lot of interaction between analysts and users for extracting the requirements. Although we may assume that such interaction would affect the quality of the resulting models, the question of which is the better order of activities is still valid. As already discussed, in spite of being aware of the interactive nature of the analysis process, different methodologies do prescribe certain orders of activities without even questioning if the prescribed order is good. Even if we agree that this study does not simulate a real analysis process, it at least proposes a good strategy for creating analysis models in cases where user requirements are already given in the form of requirements documents. Moreover, it suggests a good strategy to teach and train UML techniques.

For further research, we suggest to repeat the experiment using several case studies of different size/complexity and from different domains, not only data-driven systems, to see how the preferred order of analysis activities is affected by problem size/complexity and domain. It is especially interesting to see the results when simulating an analysis process that is similar to the real-world analysis, where the analysts have to elicit the requirements rather than work with a pre-defined requirements document. Another point is to conduct the experiments with experienced analysts rather than with students.

References

1. Anda, B. & Sjoberg, D. (2002). Towards an inspection technique for use case models. Proc. 14th Int'l Conference on Software Engineering and Knowledge Engineering (SEKE '02), Ischia, Italy, 127-134.
2. Batra, D. (1993). A framework for studying human error behavior in conceptual database modeling. Information & Management, 24, 121-131.
3. Batra, D., Hoffer, J. & Bostrom, R. (1990). Comparing representations with the Relational and Extended Entity Relationship model. Communications of the ACM, 33, 126-139.
4. Booch, G., Rumbaugh, J. & Jacobson, I. (1999). The Unified Modeling Language User Guide. Addison Wesley.
5. Coad, O. & Yourdon, E. (1991). Object-Oriented Design. Prentice Hall.
6. Cockburn, A. (2001). Writing Effective Use Cases. Addison Wesley.
7. DeMarco, T. (1978). Structured Analysis and System Specifications. Yourdon Press.
8. Dobing, B. & Parsons, J. (2000). Understanding the role of use cases in UML: a review and research agenda. Journal of Database Management, 11 (4), 28-36.
9. Jacobson, I., Booch, G. & Rumbaugh, L. (1999). The Unified Software Development Process. Addison Wesley.
10. Jacobson, I., Christerson, M., Jonsson, P. & Overgaard, G. (1992). Object-Oriented Software Engineering: A Use Case Driven Approach. Addison Wesley.
11. Kim, Y. & March, S. (1995). Comparing data modeling formalisms. Communications of the ACM, 38 (6), 103-115.
12. Larman, C. (2002). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design, and the Unified Process (2nd Edition). Prentice Hall.
13. Maciaszek, L. (2001). Requirements Analysis and System Design: Developing Information Systems with UML. Addison Wesley.
14. Mathiassen, L., Munk-Madsen, A., Nielsen, P. & Stage, J. (2000). Object Oriented Analysis and Design. Marko Publishing, Alborg, Denmark.
15. Rational Unified Process (RUP). http://encyclopedia.thefreedictionary.com/
16. Rosenberg, D. & Kendall, S. (2001). Applied Use Case-Driven Object Modeling. Addison Wesley.
17. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. & Lorensen, W. (1991). Object Oriented Modeling and Design. Prentice Hall.
18. Shlaer, S. & Mellor, S. (1992). Object Lifecycles: Modeling the World in States. Prentice Hall.
19. Shoval, P. & Kabeli, J. (2001). FOOM: functional- and object-oriented analysis & design of information systems: An integrated methodology. Journal of Database Management, 12 (1), 15-25.
20. Shoval, P. & Kabeli, J. (2005). Data modeling or functional analysis: which comes next? – an experimental comparison using FOOM methodology. Comm. of the AIS, 16, 827-843. An earlier version appeared in: Proc. of 8th CAiSE Int'l Workshop on Evaluation of Modeling Methods in Systems Analysis & Design (EMMSAD). Velden, Austria, June 03, 48-57.
21. Siau, K. & Lee, L. (2004). Are use case and class diagrams complementary in requirements analysis? An experimental study on use case and class diagrams in UML. Requirements Engineering, 9, 229-237.
22. Topi, H. & Ramesh, V. (2002). Human factors research on data modeling: a review of prior research, an extended framework and future research directions. Journal of Database Management, 13 (2), 188-217.
23. Yourdon, E. (1989). Modern Structured Analysis. Prentice Hall.
24. Yourdon, E. & Constantine, L. (1976). Structured Design. Prentice Hall.