Data Modeling and Functional Modeling – Which Comes First?

Peretz Shoval

Dept. of Information Systems Engineering, Ben-Gurion University, Beer-Sheva, Israel

Abstract

In the analysis phase of the information system development (ISD), the users' requirements are studied and defined. In most ISD methodologies, the requirements are defined by two models: a conceptual data model in form of e.g. an entity-relationship (ER) diagram or a class diagram; and a functional model in form of e.g. data flow diagrams (DFD) or use cases. Different ISD methodologies prescribe different orders of carrying out these modeling activities but there is no commonly agreed order for performing them.

In order to find out whether the order of analysis activities makes any difference, and which order leads to better results, we have conducted two independent comparative experiments in laboratory settings. The subjects in the two experiments were asked to create data models and functional models of a certain system while working in the two opposite orders. In each experiment we used a different ISD methodology and different subjects, but assigned the same modeling tasks. The main results of the experiments are that the data models are of better quality when created as the first modeling task, and that analysts prefer to first create the data model and then the functional model. The results of the two experiments have been reported in Shoval & Kabeli (2005) and in Shoval et al. (2009). This paper provides an overview of the two studies.

1. Introduction

System analysis is usually an iterative process of refinement, consisting of two main complementing activities: modeling the functional requirements and modeling the data requirements. Existing ISD methodologies differ in the prescribed order of performing these activities: some start with creating a conceptual data model and continue with developing a functional model, while others prescribe working in the opposite order. Methodologies starting with creating the data model, e.g. a class diagram, argue that the initial class diagram maps the problem domain and allows describing the functional requirements within a well-defined context. The entities in the class diagram serve as an essential glossary for describing the functional requirements, and they rarely change. On the other hand, methodologies starting with creating a functional model argue that the classes should be based on the functional requirements and should be elicited from them.

Our prior assumption is that creating a data model prior to defining the functional requirements would yield better results, because data objects are more "tangible" than functions: analysts can identify and classify more easily the objects they are dealing with and their attributes and relationships than the functions of the developed system, which are not "tangible" and may be vague. Of course, data modeling is not trivial either, but still the task of data modeling seems to be more structured and less complex compared to the task of defining and describing the system's functions. Besides, the analyst has to create just one data model (class diagram) for the system, compared to many DFDs or use cases. While in data modeling the analyst concentrates on the data-related aspects only, in functional modeling the analyst actually deals at the same time with more aspects, because DFDs or use cases are not merely about functions; they are also about data, user-system interaction and the process logic. Hence, it seems to us that starting the analysis process with the more simple and structured task would be more effective (in terms of quality of the created models) and efficient (in terms of time taken to create the models). Not only that the first product (the data model) will be good; it will ease the creation of the following one (the functional model).

One theory supporting our assumptions about the better order of analysis activities is the ontological framework proposed by Wand and Weber, known as BWW (see, for example Weber, 2003). According to this framework, the world consists of things that possess properties; this fits the "data-oriented" approach, particularly in the context of a class diagram. More complex notions such as processes (functions) are built up from the foundations of things and properties; this fits the "functional orientation" of DFDs or use cases. If we accept that a key purpose of modeling is to represent aspects of the real world, then it follows from the above framework that it is appropriate to start the modeling process by dealing with the basic "things and properties", i.e. data modeling, and based on that continue with the modeling of the more complex constructs, i.e. functional modeling.

Another support for our assumptions comes from cognitive theory, specifically from COGEVAL (Rockwell & Bajaj, 2005), a propositional framework for the evaluation of conceptual models. COGEVAL tries to explain the quality and readability of models using various cognitive theories. Proposition 1 of COGEVAL relies on Capacity theories, which assume that there is a general limit on a person's capacity to perform mental tasks, and that people have considerable control over allocation of their capacities for the task at hand. Based on these theories, for a given set of requirements, the proposition is that it will be easier to create correct models that support chunking of requirements. This proposition supports our belief that it will be easier to create a data model as it deals only with data structure constructs. Proposition 3 of COGEVAL is based on Semantic Organization theories, which indicate how items are organized in long-term memory (LTM). Spreading activation theory depicts concept nodes by relationship links, with stronger links having shorter lengths. The modeling process can be considered as taking concepts and relationships in the users LTM and capturing them in a model. Based on this proposition, data modeling methods whose constructs allow for creation of concept-node and relationship-arc models, as a class diagram, will offer greater modeling effectiveness than other modeling methods, such as DFDs and use cases, whose constructs are not organized as concept nodes and relationships.

Based on the above discussion, our initial expectation is that it is better to start with data modeling. However, as shown in the survey in Section 2, in fact different methodologies advocate different orders. Therefore, for the sake of these studies, we hypothesize (in Section 3) that there is no difference in the quality of the analysis models when created in either order. Similarly, we hypothesize that there is no difference in the analysts' preference of the order of activities. These hypotheses were tested in two comparative experiments that are reviewed in Sections 4 and 5. Section 6 concludes and discusses limitations of the experiments.

2. Order of Analysis in ISD Methodologies

In the early ISD methodologies, which emerged in the mid 1970's (e.g. DeMarco, 1978), the emphasis of analysis was on defining the functional requirements by conducting a functional decomposition using mainly DFDs. Later on, with the introduction of conceptual data models, many methodologies included also data modeling in the analysis phase (e.g. Yourdon, 1989). Nevertheless, functional analysis was still the primary activity and the data analysis was only secondary to it.

In the early 1990's, with the emergence of the object-oriented (OO) approach to ISD, new methodologies were introduced to support the analysis and design of information systems according to this concept. Some of these methodologies (e.g. Coad & Yourdon, 1991; Meyer, 1998; Rumbaugh et al., 1991; Shlaer & Mellor, 1992) suggested a different analysis concept: object-driven, instead of functional-driven, i.e. the analysis emphasis is on finding the domain objects, and based on them identifying the services that these objects ought to provide. However, there were still methodologies that kept with the functional-driven approach, where a functional analysis is performed first and then the object model is derived from it, whether directly or through a conceptual data model; for example: Jacobson et al. (1992), and Rubin & Goldberg (1992).

Aiming at solving the problems raised by the vast amount of methods and techniques, UML (Unified Modeling Language) was adopted by the Object Management Group as its standard for modeling OO systems. UML's main tools for describing the user/functional requirements and the object model of a system are use cases and class diagrams, respectively. Being merely a modeling language, UML defines no development process. However, many methodologies that use the UML have been developed. Despite many variations between different UML-based methodologies, in most of them the analysis phase comprises two main activities: data model using class diagrams, and functional modeling using use case diagrams and descriptions.

UML-based methodologies which adopt use cases as the requirements description tool are usually "use case-driven". Unified Process - UP (Jacobson, Booch & Rumbaugh, 1999) is one of the most documented and influential use case-driven methodologies. The first analysis activity according to UP is creating use case scenarios, whilst an initial class diagram is only created in the next phase called Use Case Analysis. Since UP is a most commonly used methodology, it is common in industry that users spend a great deal of effort on conducting use case analysis aimed at identifying the business requirements, while class diagrams are seen as more closely associated with system design and implementation, therefore often delayed until a reasonable use case analysis is performed.

Another example for an OO methodology that starts with functional modeling is Insfran et al. (2002). It starts by specifying high-level functional requirements, which are then systematically decomposed into a more detailed specification. They define a requirements model (RM) that captures the functional and usage aspects, and a requirements analysis process that translates those requirements into a conceptual schema specification. The RM includes a functional refinement tree, i.e. a hierarchical decomposition of the business functions, which are used for building use case specifications. The use cases are used as basis for the conceptual modeling phase. The major activities at this stage are to define an object model, a dynamic model, and a functional model. This process is based on the analysis of each use case aimed at identifying classes and allocating their responsibilities.

But there are also UML-based methodologies that start the analysis process with data modeling. For example, UMM (Hofreiter et. al, 2006), an enhancement of UP specifically intended to model business processes and support the development of electronic data interchange (EDI) systems. Unlike the previously surveyed methodologies, UMM starts with a phase called Business Modeling in which the first step is performing domain analysis. Identifying the “top-level entities” in the problem domain is done using Business Operations Map, a tool that presents a hierarchy of Business Areas, Process Areas and Business Processes. Subsequently, use cases are created to describe the functional requirements.

Similarly, ICONIX (Rosenberg & Kendall, 2001) suggest starting the analysis by 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.

FOOM methodology (Shoval, 2007; Shoval & Kabeli, 2001) too suggest starting the analysis by creating a conceptual data model termed initial class diagram, and then creating a functional model consisting of hierarchical OO-DFDs. An OO-DFD is similar to a "traditional" DFD but instead of data stores it includes data classes, which are taken from the initial class diagram. At the end of the analysis phase, before moving on to designing the system, the two modes are verified and synchronized.

3. The Research Goals and Hypotheses

The goal of our experiments was to determine through controlled comparative experiments whether there is a preferred order for performing the two modeling activities. In each experiment we formed two homogeneous groups of subjects who played the role of analysts. Each subject was asked to create a functional and a data model of a certain system based on a requirements document. We used the same system in the two experiments - the *IFIP Conference* case study (Mathiassen et al, 2000). Subject in one group started with data modeling and continued with functional modeling; subject of the other group worked in the opposite order. The difference between the two experiments was mainly in the ISD methodology: in the first experiment we used FOOM methodology, while in the second – a UML-based methodology.

4. Experiment 1

4.1 The ISD methodology

In this experiment we used the FOOM methodology (Shoval, 2007; shoval and Kabeli, 2001), which combines the functional and the OO approaches. In the analysis phase of FOOM, two models are created: a) a conceptual data model, in the form of an initial class diagram, and b) a functional model, in the form of hierarchical OO-DFDs. The initial class diagram consists of data (entity) classes, with attributes and various types of associations. (The notation of the FOOM class diagram is somewhat different from the UML class diagram; for example, it includes reference attributes in addition to association links between classes). The OO-DFDs are similar to traditional DFDs but, as said, they include data classes instead of data stores.

The above modeling activities of the FOOM analysis phase can be performed in opposing orders, one beginning by creating the OO-DFDs and then specifying the initial class diagram, and the other by first creating the initial class diagram and then specifying the OO-DFDs. When the performing order begins with functional analysis the analyst first produces a hierarchy of OO-DFDs (based on the users' requirements), and then creates an initial class diagram including classes already appearing in those OO-DFDs. This amounts to mainly defining appropriate associations among the classes and their attributes. When the performing order begins with data modeling, the analyst first creates an initial class diagram (based on the users' requirements) and then creates the OO-DFDs using the already defined classes. In any case, no matter which order of activities is followed, FOOM completes the analysis phase by verifying that the two products are in sync; i.e., that each class specified in the class diagram appears in at least one of the OO-DFDs, and vice versa, and that each attribute of a class is updated and retrieved by at least one function.

In the design phase, the above products are used to create a complete class diagram, including Interface, Input, Output and Control classes, and mainly to design the various class methods. (More details about FOOM are beyond the scope of this paper.)

4.2. Description of the experiment

The rationale behind this experiment, besides the above discussion about various methodologies and their approach to this issue, we were concerned about which order of analysis activities to recommend in the FOOM methodology because the two orders are possible. In this experiment we have defined three dependent variables, which are surrogates for performance: a) Quality of model: we examined how good/correct are the functional and the data models created by the subjects while working in the opposite orders; b) Time: we examined how long it takes to create each of the two models while working in the opposite orders; c) Preference: we examined which order of analysis activities the analysts prefer.

In order to measure and compare the correctness of the created models and control other variables, we instructed the subjects to analyze the requirements document and create an initial class diagram and three OO-DFDs: a root DFD and two sub-level DFDs. The subjects were undergraduate students of Information Systems Engineering who took the same courses and studied the functional and OO approaches and the FOOM methodology in the same classes with the same instructor. The subjects were motivated by the fact that their grades on quality (correctness) of specifications would be considered as part of their course grades. The subjects were divided randomly into two groups: the 27 members in one group (marked DF) started with creating the data model, while the 29 members in the other group (marked FD) started with creating the functional model.

When the experiment began each subject received the requirements document of the *IFIP Conference* system and solution sheets for the first task. After a subject completed the first task, sheet(s) for the second task were handed out. The solution sheets of the first task were not handed in but were used in preparing the solution for the second task. The start and completion times of each task were recorded. At the end of the experiment each subject was asked to complete a short subjective questionnaire, using a 7 point scale, on the preferred order of analysis.

The quality or correctness of each model (class diagram or OO-DFDs) was measured according to the number and type of error found. We used a grading scheme which determined the number of points deducted per error type according to the severity of the error, a method that was also used in previous studies. Each model (diagram) was graded separately. The grade of the functional model was computed as the average of the three OO-DFD grades.

4.3 Results

4.3.1 Quality of models

Tables 1 present the grades of the quality of models. The last column denotes if the results are significant.

Table 1. Quality of Models

Model	Analysis Order	Mean Grade (%)	F	P-value (α=0.05)	Significance in favor of
Data model	DF FD	71.48 58.50	7.23	0.0098	**Starting with data model**
Functional model	DF FD	83.86 83.71	0.004	0.9474	-

As can be seen, the quality of the data model (1st row) is significantly better when created before the functional model (mean grades 71.48 and 58.5, respectively).

On the other hand, we found no significant difference in qualities of the functional models (2nd row). (Note that in both orders of analysis, the grades of the functional model are relatively high compared to the grades of the data model. A possible explanation to this may be that the grading scheme caused the deduction of more error points on the data model. In any case, this does not influence the conclusions so long as the same grading scheme is used on all subjects.)

Since we found no significant difference with respect to the functional models, but a significant difference with respect to the data models, we conclude that all together the analysis order beginning with data modeling provides better results.

4.3.2 Time taken to complete the tasks

Time in itself may be an unimportant factor (mainly when dealing with "toy" systems that take only about two hours to model), but when we relate time with performance (quality) interesting results may arise. As can be seen in Table 2, the total time taken to create the two models was almost the same for the two groups, but within each group, while the time taken to perform the two tasks by the DF subjects (who started with data modeling) was almost the same (58.09 and 56.29), the time taken to perform the first task by the FD subjects (who started with functional modeling) was much longer that the time taken to perform the second task (74.24. compared to 38.6). In other words, those who started working on the functional model had to spend so much time on it that not much time was left for the data modeling task. This can explain why these subjects performed so poor on the data modeling task (as seen in Table 1).

Table 2. Times to Complete the Analysis tasks

Analysis Order	Mean Time (minutes)		Significance in favor of
	Data model	Functiona l model	
DF	58.09	56.29	**Starting with data model**
FD	38.6	74.24	

4.3.3 Preferences of analysts

Table 3 presents the results of the subjective preferences of analysts. Row DF shows the scores of preference of the analysis order commencing with data modeling, and row FD shows the scores of preference of the analysis order commencing with functional modeling. Column DF shows the scores given by subjects that first performing data modeling and column FD the scores of those first performing functional analysis. Column "All together" shows the mean scores of all subjects together. It can be seen that all subjects were in favor of the analysis order beginning with data modeling. Interestingly, subjects who began with functional analysis also preferred the order of beginning with data modeling, even though they received low grades on it.

Table 3. Preference of Analysis Order

Analysis Order	Mean Scores (1-7 point scale)			Significance in favor of
	DF (N=27)	FD (N=29)	All together	
DF	4.48	5.724	5.103	**Starting with data model**
FD	4.14	2.586	3.367	

5. Experiment 2

5.1 The ISD methodology and description of the experiment

This experiment was carried out about two years later, with a different group of students and a different methodology. The main reason for repeating the experiment was that in the first experiment we used the FOOM methodology, which is not as common as UML-based methodologies. So in order to examine the validity of the previous results we have decided repeat the experiment, using the same *IFIP Conference* system, but for data modeling we used the "standard" UML class diagram, and for functional modeling we used use case diagrams and descriptions.

The subjects of this experiment too were senior students of Information Systems Engineering. We performed this experiment as a mid-term exam in the OO Analysis and Design course. Before the exam, the subjects learned the OO analysis approach and UML, including use cases and class diagrams. To direct the subjects toward a unified style of describing use cases, a tutorial was conducted that included teaching Alistair Cockburn's (2001) use case writing guidelines.

In this experiment there were 121 subjects: 57 in the DF group and 64 in the FD group. Here too we motivate the subjects to perform the tasks to the best of their ability by considering their quality grades as a part of the final course grade. In this experiment the dependent variables were the quality of the created models and the subjective preferences of the analysts (but we did not measure the time taken to complete the tasks). Similar to the 1st experiment, the quality of the analysis models was measured using grading schemes that determined the number of points to deduct for each error type in every model. But here the schemes were adapted for UML class diagrams and use cases. Since class diagram is a well-defined tool with a strict structure and syntax, mapping the possible errors in it was straightforward. Use cases, on the other hand, are less structured and described using free text. Mapping the possible errors in the use case descriptions required first defining the components that the analysts are expected to describe and that we would like to measure in terms of quality. To create the grading scheme, we were assisted by the use of case error mapping in Anda & Sjoberg (2002), which is also based on Cockburn's approach (Cockburn, 2001). After identifying the components, as in the class diagram, we mapped the possible errors in each component, and determined the error points.

5.2 Results

5.2.1 Quality of models

Table 4 presents the grades of quality of the models. The first row presents the results for the data model (class diagram). As can be seen, the grades are significantly higher when starting the analysis with data modeling (73.63) compared to when starting with functional modeling (70.25). The second row presents the quality results of the functional models (use cases). As can be seen, here there are no significant differences.

It should be noted that the quality results of this experiment are consistent with the results obtained in the 1st experiment.

Table 4. Quality of Models

Model	Analysis Order	Mean Grade (%)	F	P-value (α=0.05)	Significance in favor of
Data model	DF FD	73.63 70.25	4.386	0.038	**Starting with data model**
Functional model	DF FD	63.72 65.03	0.192	0.662	-

5.2.2 Preferences of analysts

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 a class diagram, and 7 means total preference to start with use cases. Table 5 presents the results for each group of subjects and for all together. The results show that the subjects definitely prefer to create a class diagram first and then use cases (mean preference of all is 2.78; much closer to 1 than to 7). Interestingly, 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 expectation and the results obtained in the first experiment.

Table 5. Analysts' Preferences

Analysis order	Mean preference	STD	Significance in favor of
DF	2.91	1.54	**Starting with data model**
FD	2.61	1.82	
All together	2.78	1.66	

6. Conclusions and Limitations

The results of the two experiments reveal that starting the analysis with data modeling yields a better data model. On the other hand, we did not find a significant effect of the analysis order on the quality of the functional models. Thus, starting the analysis process with data modeling should result in a better overall quality of the analysis models. We also found out that analysts prefer creating the data model first, and that when working in this order, the time to create the two models is allocated more effectively.

As other controlled experiments in laboratory settings, these two too have limitations threatening their validity. An obvious limitation is that we used a relatively small and simple problem (the *IFIP Conference* system) while in reality problems are usually 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 is that we used students with almost no industrial experience as surrogates for analysts. This limitation is common to almost all experimental work in Software Engineering. We cannot predict if and how the cumulative experience of analysts might affect the preferred order of analysis activities.

A potential limitation of the experiments is the grading schemes. Some subjectivity may exist in the weights of error points. We determined the weights based on our assessment of the importance of each error type. In doing so we followed earlier studies who also adopted subjective grading schemes to assess quality of models. The potential problem with such grading schemes is that the subjective weights (error points) assigned to 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 a separate research.

Being "laboratory" experiments, 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. This is not really an analysis task; in reality there are real users who have real needs, and the analysis process involves a lot of interaction between analysts and users. 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, still different methodologies prescribe certain orders of activities without even questioning if the prescribed order is good. Even if we agree that our studies do not necessarily simulate real analysis processes, they at least propose a good strategy to create analysis models in cases where user requirements are already given in the form of a requirements document.

References

Anda, B. & Sjoberg, D. (2002). Towards an inspection technique for use case models. *Proceeding of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE '02),* Ischia, Italy, 127-134.

Coad, O. & Yourdon, E. (1991). *Object-Oriented Design.* Englewood Cliffs, NJ: Prentice Hall.

Cockburn, A. (2001). *Writing Effective Use Cases*: Addison-Wesley.

DeMarco, T. (1978). *Structured Analysis and System Specifications*: Yourdon Press.

Hofreiter, B., Huemer, C., Liegl, P., Schuster, R. and Zapletal, M. (2006). UN/CEFACT'S modeling methodology (UMM): a UML profile for B2B e-commerce. Retrieved from http://dme.researchstudio.at/publications/2006/

Insfran, E., Pastor, O. & Wieringa, R. (2002). Requirements engineering-based conceptual modeling. *Requirements Engineering,* 7, 61-72.

Jacobson, I., Booch, G. & Rumbaugh, L. (1999). *The Unified Software Development Process*: Addison-Wesley.

Jacobson, I., Christerson, M., Jonsson, P. and Overgaard, G. (1992). *Object-Oriented Software Engineering: A Use Case Driven Approach.* Addison Wesley.

Mathiassen, L., Munk-Madsen, A., Nielsen, P. & Stage, J. (2000). *Object Oriented Analysis and Design.* Marko Publishing, Alborg, Denmark.

Meyer, B. (1998). *Object Oriented Software Construction*: Prentice Hall.

Rockwell, S. & Bajaj, A. (2005). COGEVAL: Applying cognitive theories to evaluate conceptual models. In: Siau, K. (Ed.): *Advanced Topics in Databases Research*, Idea Group, Hershey, PA, 255-282.

Rosenberg, D. & Kendall, S. (2001). *Applied Use Case-Driven Object Modeling*: Addison-Wesley.

Rubin, K. & Goldberg, A. (1992). Object Behavior Analysis. *Communications of the ACM,* 35(9), 48-62.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. & Lorensen, W. (1991). *Object Oriented Modeling and Design.* Englewood Cliffs, NJ: Prentice Hall.

Shlaer, S. & Mellor, S. (1992). *Object Lifecycles: Modeling the World in States*: Prentice Hall.

Shoval, P. & Kabeli, J. (2001). FOOM: functional- and object-oriented analysis and design of information systems: An integrated methodology. *Journal of Database Management,* 12(1), 15-25.

Shoval, P. & Kabeli, J. (2005). Data modeling or functional analysis: which comes next? - An experimental comparison using FOOM methodology. *Communications of the AIS,* 16, 827-843.

Shoval, P. (2007). *Functional and Object-Oriented Analysis & Design - An Integrated Methodology*. IGI - Idea Group. (337 pages).

Shoval, P. & Shiran, S. (1997). Entity-relationship and object-oriented data modeling - an experimental comparison of design quality. Data & Knowledge Engineering, 21, 297-315.

Shoval, P., Last, M & Yampolsky, A. (2009). Data modeling and functional modeling - examining the preferred order of using UML class diagrams and use cases. In: Halpin, Proper & Krogstie (Eds.): Innovations in Information Systems Modeling - Methods and Best Practices. IGI Global, Ch. 7, pp. 122-142.

Weber, R. (2003). Conceptual modeling and ontology: possibilities and pitfalls. *Journal of Database management,* 14 (3), 1-20.

Yourdon, E. (1989). *Modern Structured Analysis*: Prentice Hall.