Comprehension and quality of analysis specifications—a comparison of FOOM and OPM methodologies

Judith Kabeli, Peretz Shoval*

Department of Information Systems Engineering, Ben-Gurion University, Beer-Sheva 84105, Israel

Received 30 November 2003; revised 6 July 2004; accepted 25 August 2004

Abstract

FOOM—Functional and Object Oriented Methodology—combines two essential software-engineering paradigms: the functional (or process-oriented) approach and the object-oriented (OO) approach. The two main products of the analysis phase of FOOM are an initial class diagram and OO-DFDs (dataflow diagrams including data classes rather than traditional data-stores). We evaluated these analysis products by comparing them with the analysis products of OPM—Object-Process Methodology—which also combines the functional and object-oriented approaches, using a unified diagrammatic notation. FOOM and OPM were compared in two controlled experiments from two main points of view: users and analysts. From the point of view of users we compared mainly comprehension of analysis specifications in each methodology. From the point of view of analysts we compared mainly quality, namely correctness of specifications created by analysts who utilized the two methodologies. The main results of the experiments are that FOOM specifications are more comprehensible and preferred by users, and that analysts create more correct specifications when using FOOM methodology.

© 2004 Published by Elsevier B.V.

Keywords: Conceptual modeling; Data modeling; Functional analysis; Experimentation; Method engineering; Method evaluation and comparison; Object-oriented analysis and design; Requirements engineering; System development methodologies; ADISSA; FOOM; OPM

1. Introduction

FOOM [16] is a methodology for analysis and design of information systems that combines the two essential software-engineering paradigms: the functional approach (or process-oriented) and the object-oriented (OO) approach. FOOM utilizes well-known techniques such as DFDs (Data Flow Diagrams) and provides simple visual modeling and notations. It covers the structural and the behavior aspects of a system through the analysis and design phases, and provides a natural and smooth transition from one phase to the other. Object-Processes Methodology (OPM) [5] is another methodology for analysis and design of information systems that combines the process and object-oriented approaches, providing a unified notation for the structural and behavior aspects of a system. Like FOOM, OPM also utilizes ‘DFD-like’ diagrams, applies hierarchical decomposition, and creates class diagrams. Unlike UML-based methodologies, which utilize a multitude of diagram types and notations covering various aspects of the modeled system, FOOM and OPM utilize only a small number of diagram types, which combine data and functional modeling. However, the methodologies differ in that OPM diagrams include more symbols and link types than the FOOM equivalent diagrams. They also differ in the way they use their analysis products in the next phases of system development. Because of their similarities and differences in the way they perform the phase of system analysis, we found it interesting to evaluate and compare their analysis products, i.e. analysis specifications.

Methodologies can be evaluated and compared on various dimensions, e.g. quality of the analysis or design products, comprehensibility, learn ability, ease of use, user/developer satisfaction or preference, and more. In this paper we report on two controlled experiments in which we evaluated and compared the analysis specifications created...
in FOOM and OPM from two points of view: users and analysts. In one experiment, which took the users’ point of view, we measured: (a) comprehension: how well users understand the analysis specifications presented in the two methodologies; (b) time: how long it takes the users to comprehend the specifications; and (c) preference: which specifications users prefer. In the other experiment, which took the analysts’ point of view, we measured quality of specifications, i.e. how correct are the specifications created by analysts who utilized the two methodologies. The main results of the two experiments are as follows: (a) users comprehend better the FOOM specifications; (b) it takes them less time to comprehend FOOM specifications; (c) users prefer the FOOM specifications; and (d) analysts create more correct specifications when using FOOM methodology.

The rest of the paper is structured as follows: Sections 2 and 3 describe briefly FOOM and OPM methodologies, concentrating on the analysis phase, on which we focus in this study. Section 4 provides a short overview of related studies on experimental evaluation and comparison of methods. Section 5 deals with theoretical issues concerning the comparison between the two methodologies, and the research hypotheses. Then, Section 6 describes the ‘comprehension of specifications’ experiment, and Section 7 describes the ‘quality of specifications’ experiment. Section 8 discusses threads to validity of the results, and Section 9 summarizes and discusses further research issues.

2. Essential of FOOM methodology

FOOM methodology uses well-known visual modeling techniques and a small number of diagram types encompassing the analysis and design phases of development, while enabling a smooth transition from one phase to the other. The analysis phase of FOOM consists of two main activities: data analysis and functional analysis, and its two main products are: (a) a 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 classes, but without methods. The OO-DFDs are similar to traditional DFDs but they include data classes rather than data-stores; the classes are taken from the initial class diagram. These products of analysis are used to design the system. The main products of the design phase are a complete class diagram, including all the classes and detailed descriptions of their methods. Other products of the design phase include the user-interface (a menus tree) and the input and output screens and reports.

FOOM is described in more detail in [8,16]. Here we provide a brief description only, and demonstrate the analysis products using examples from the IFIP Conference case study [12].

2.1. The analysis phase

The main products of the analysis phase are a data model: an initial class diagram, and a functional model: hierarchical OO-DFDs. The initial class diagram consists of data (or entity) classes, i.e. classes that are derived from the application requirements and contain ‘real world’ data. Other classes will be added at the design stage. Each class includes attributes of various types. Relationships between classes include ‘regular’ associations, generalization (inheritance) hierarchies, and aggregation (part-of) links. The initial class diagram does not include methods because methods results from a functional modeling process; these will be added at the design phase. An example of an initial class diagram is shown in Fig. 1.1

The hierarchical OO-DFD diagrams specify the functional requirements of the system. An OO-DFD may consist of general (decomposable) and elementary functions, external-entities (user-entities, time and real-time entities), data classes (instead of data stores in traditional DFDs), and data flows among them. Examples of OO-DFDs are shown in Figs. 2 and 3: Fig. 2 shows the root OO-DFD of the IFIP Conference, and Fig. 3—an OO-DFD, which details one of its general functions: Papers Selection.

The activities of the FOOM analysis phase can be performed in two different orders. One possibility is to specify first the OO-DFDs and then the initial class diagram. An opposite possibility is to specify first the initial class diagram and then the OO-DFDs. An experimental comparison of these two orderings of activities revealed that it is better to start by specifying the initial class diagram and then the OO-DFDs, thus utilizing the already-defined data classes [9]. At any rate, the products of the analysis phase are synchronized before continuing to the design phase. Synchronization involves two types of consistency checks: At the class level, it is checked that all classes appearing in the class diagram appear also in the OO-DFDs, vice versa. At the attributes level, it is checked that each attribute of a class is updated by at least one function (via a ‘write’ dataflow) and is retrieved by at least one function (via a ‘read’ dataflow) of an OO-DFD.

2.2. The design phase

Since this study is mainly concerned with evaluation of the analysis specifications, the design activities are only briefly described.

2.2.1. Top-level design of transactions

This stage is performed according to ADISSA methodology [18], which defines a transaction as a process supporting a user who performs a business function, triggered as result of an event (similar to a ‘use case’ in UML).

1 Due to space limitation, we do not explain the examples.
The transactions of a system are identifiable in the OO-DFDs: a transaction consists of one or more chained elementary functions, and of classes and external-entities connected to these functions. For example, the OO-DFD in Fig. 3 includes several transactions; one transaction consists of Function 2.1 and the components connected to these functions by data flows: external-entities E2 and E7, and classes Author and Reviewer. Another transaction consists Fig. 1. Initial class diagram of IFIP Conference.
of Functions 2.2 and 2.3 and the components connected to them. Note that every elementary function can belong to only one transaction, while an external-entity or a class may participate in many transactions.

The products of this stage include transactions diagrams, which are extracted from the OO-DFDs; and their top-level descriptions. Each transaction will later-on be decomposed into one or more methods that will be attached to proper

Fig. 2. OO-DFD-0 of IFIP Conference.
classes; the ‘main’ method of each transaction will be attached to an abstract class named ‘Transactions’. A top-level transaction description is provided in pseudo-code. It refers to all components of the transaction: every data-flow from or to an external-entity is translated to an ‘Input from...’ or ‘Output to...’ command; every data-flow from or to a class is translated to a ‘Read from...’ or ‘Write to...’ command; and every function is translated into an ‘Execute
function...’ command. The process logic of a transaction is expressed using standard structured programming patterns. The analyst and the user, who presents the application requirements, determine the process logic of each transaction; this cannot be deduced automatically from the transaction diagrams alone, because a given diagram can be interpreted in different ways, and it is up to the user to determine its intended meaning/way of use.

2.2.2. Interface design—the Menus class

This stage is also performed following ADISSA methodology. A menu-tree interface is derived in a semi-algorithmic way from the hierarchy of OO-DFDs. Generally, a general function connected to a user-entity creates a menu, and an elementary function connected to a user-entity creates a menu-item within its parent menu. Eventually, all menus become objects of the ‘Menus’ class, which is added to the class diagram.

2.2.3. Design of the inputs and outputs—the Forms and Reports classes

The design of the input/output screens and reports is based on the ‘Input from...’ and ‘Output to...’ commands appearing in the transaction descriptions For each ‘Input from...’ command, an input screen/form is designed; and for each ‘Output to...’ command, an output screen/report is designed. Depending on the process logic of each transaction, some or all of its input or output screens may be combined. Consequently, two new classes are added to the class diagram: ‘Forms’ class for the inputs, and ‘Reports’ class for the outputs. Obviously, the objects of each of these classes are the input screens and output screens/reports, respectively.

2.2.4. Design of system behavior—the methods

In this stage, the top-level descriptions of the transactions are converted into detailed descriptions of class methods. The transition from a top-level transaction description to detailed descriptions of methods emanating from it is done as follows: Every ‘Input from...’ and ‘Output to...’ command in the top-level description is translated to a message invoking a ‘display’ method of the respective Forms or Reports class-object; this method will enable performing the input or just presenting the output, respectively. Every ‘Read from...’ or ‘Write to...’ command is translated to a message invoking a basic-method of the appropriate data class. Basic methods include Create, Read, Update and Delete (CRUD), which every data class is assumed to have or inherit from a super class. Every ‘Execute-Function...’ command can be translated to a basic-method of a certain class, or to a specific-method that will be attached to a proper class. A specific method may perform a procedure/function which is specific to the application, beyond the basic CRUD methods. Each procedure which is defined as a basic or a specific method of some class is removed from the transaction’s description and replaced by a message to that class-method. A procedure/function which encompasses several classes must not be defined as a specific-method of a certain class; rather it may remain within the transaction. The remaining parts of the transaction’s description are defined as transaction-method. This method will become the ‘main’ part of the transaction’s program, and belongs to the ‘Transactions’ class. Hence, when a user wants the system to perform a task (transaction), he/she actually invokes the ‘main’ method of the respective transaction (via proper menus selections); while that method executes, it may invoke (namely, send messages to) other, basic- or specific-methods of respective classes—according to the process logic of the transaction.

Each transaction-method or specific-method can be described in two complementing forms: pseudo-code and message-chart. A pseudo-code description details the process logic of a method using standard structured programming patterns. A message-chart provides an equivalent description; it is similar to a structured flowchart, showing the procedures, classes, methods and messages included in a method, and the order/process logic of their execution.

In summary, the products of FOOM’s design phase include: (a) a complete class diagram, including Data, Menus, Forms, Reports and Transactions classes; (b) detailed menu objects of the Menus class; (c) detailed form and report objects of the Forms and Reports classes; (d) detailed descriptions of the transaction-methods and the specific class-methods, expressed in pseudo-code or message-charts. These products will be used to create the software in any OO development environment.

3. Essentials of OPM methodology

OPM [4,5] is a systems development methodology that combines the major system aspects—function, structure and behavior—within a single graphic and textual model, in which both objects and processes are represented without suppressing each other. This approach counters object-oriented systems development methods, notably UML, which require several models to completely specify a system. OPM is, therefore, not yet another OO analysis and design method, as it recognizes the fact that separating structure from behavior while engaging in system modeling, which results in the model multiplicity problem, is counterintuitive.2

In the OPM ontology, objects are viewed as persistent, state preserving things (entities) that interact with each other through processes—another type of

2 This argument of the creator of OPM is perhaps not very convincing for readers who think that the ‘separation of concerns’ of UML is exactly what is needed to model complex phenomena.
things. Thing is a generalization of an object and a process. Processes are patterns of behavior that transform objects by transforming them. Transformation is a generalization of effect, consumption and generation. Hence, transforming objects implies affecting them (i.e. changing their states), or generating new objects, or consuming existing objects.

To avoid model multiplicity, OPM incorporates the static-structural and behavioral-procedural aspects of a system into a single, unifying graphic-textual model. It uses Object-Process Diagrams (OPDs) for the graphic specification. OPDs are workflow-like hyper graphs that model the system or parts of it at various levels of detail. Objects and processes are connected by procedural links, which can be either enabling links or transformation links. These two different kinds of links are used to connect objects to processes, depending on the roles that these objects play in the process to which they are linked. Objects may serve as enablers—instruments or intelligent agents, which are involved in the process without changing their state. Objects may also be transformed (change their state, generated, consumed, or affected) as a result of a process acting on them.

An enabling link connects an enabler to the process that is enables. Enabler is an enabling object that needs to be present in order for the process to occur but it does not change as a result of the process occurrence. An enabling link can be an agent link or an instrument link. An agent link denotes that relative to the enabled process, the enabler is an intelligent agent—a human or an organizational unit that comprises humans, such as a department or an entire enterprise. An instrument link is an enabling link denoted by a white circle at the process end, which denotes that the enabler is an instrument—a non-human physical or informational object (machine, file, etc.) that must be present for the process to take place but is not affected by the process. The consumption link is a transformation link denoted as a unidirectional arrow from the consumed object to the consuming process.

OPM uses Object-Process Language (OPL) for the textual specification. Based on a constrained context-free grammar, a textual description in a natural-like language can be automatically extracted from the diagrammatic description in the OPD set. Figs. 4 and 7 show OPDs created for the same *IFIP Conference* case study. Figs. 4 and 5 show part of the object-class model (equivalent to FOOM’s class diagram shown in Fig. 1). Note that Fig. 5 is an exploding of the object ‘Paper’. Fig. 6 shows the main OPD (equivalent to FOOM’s OO-DFD-0 in Fig. 2), and Fig. 7 explodes the ‘Papers Selection’ process (equivalent to FOOM’s OO-DFD-2 in Fig. 3).

![Fig. 4. OPD-class model of IFIP Conference.](image-url)
Fig. 5. Explosion of ‘Paper’ object.

Fig. 6. Main OPD of IFIP Conference.
4. Related studies on model/method comparisons

In their review and synthesis of research on data modeling, Topi and Ramesh [19] surveyed 27 studies that appeared between 1978 and 2001, which employ mainly laboratory experiments to evaluate the usability of data models/methods. As they found out, the most frequent independent variable in those studies is the data model. The next category of independent variables consists of user characteristics (e.g. experience, education and intellectual ability). Other independent variables are task characteristics (e.g. comprehension and task complexity). The dependent variables are divided into two main categories: performance and attitude. Performance is divided into three subcategories: model correctness, time used to create the solution, and declarative knowledge (understanding of the notation). In most cases, the correctness of a model has been measured according to the degree to which it corresponds to a predefined solution. Attitude includes mainly preference to use a certain model, and perceived ease of use. There are studies that take an analyst/designer/modeler perspective; these are mainly concerned with measuring comprehensibility of models and preference of models by users. A few relevant studies are summarized below.

Batra and Antony [2] compared the Entity-Relationship (ER) with relational modeling from the point of view of analyst performance. Two groups of analysts received a textual description of user requirements of a certain information system, and were asked to map the description to ER diagrams and to normalized-relations, respectively. Analyst performance was measured according to the correctness and completeness of the produced schemas. The authors found that ER diagram reached significantly higher grades compared to the relational schemas.

Kim and March [10] compared EER (Extended ER) and NIAM (Nijssen Information Analysis Method) from the point of view of user comprehension of diagrams. The experiment subjects were graduate students who were randomly assigned to one of two groups, and trained in one of these modeling techniques. The authors prepared two equivalent specifications of a manufacturing company, one in each method. Comprehension was measured by counting the number of correct answers to questions concerning various modeling constructs. No significant difference in comprehension between the two models was found.
Shoval and Frumermann [15] compared EER and OO models from the perspective of user comprehension of diagrams. Two groups of users were given equivalent EER and OO (class) diagrams. Comprehension was based on a questionnaire consisting of statements concerning various constructs of the two data models. Subjects of the experiment were students of Behavioral Science and Management. The authors found a significant difference in comprehension of ternary relationships in favor of the EER model, and no significant difference in comprehension of other constructs (e.g., binary relationships).

In a follow up study, Shoval and Shiran [17] compared the same EER and the OO models from the point of view of analysts/designers, namely quality of schema specification. They also measured the time to complete the design tasks and the designers’ preferences of models. The subjects of this experiment were students of Information Systems. Subjects in two groups were given the same design tasks, but each group using a different model. Performance was measured according to the number of correct/incorrect constructs created using each model. The authors found that the EER model is better than the OO model in specifying unary and ternary relationships, but there were no significant differences in specifying other constructs. They also found that it takes less time to create EER schemas; and that designers prefer modeling with EER.

Peleg and Dori [13] compared two methodologies: OPM/T, a variant of OPM for real-time systems; and OMT/T, a similar variant of OMT [14]. The study included comparison of both data modeling and process specifications from the points of view of comprehension and quality. The subjects, students of Information Systems Engineering, were randomly divided into two groups. In the ‘comprehension’ part of the experiment, the subjects received diagrammatic specifications of a real-time system; subjects in one group received specifications in OPM, and subjects in the other group—in OMT. They were asked to demonstrate comprehension using a questionnaire consisting of statements that were classified according to different model constructs (similar to [15]). The level of comprehension was measured by counting for each subject the number of correct answers for each construct. The authors found that in some cases (constructs) OPM specifications are more comprehensible than OMT specifications; in some cases OMT specifications are more comprehensible; and in some cases there are no significant differences. Overall, in more cases OPM specifications are more comprehensible than OMT.

In the ‘quality’ part of the experiment the subjects received a textual problem statement of another real-time system. Participants in one group were asked to specify the system using OMT, while participants in the other group were asked to use OPM. Quality was measured by counting the number of errors found in their specifications. The authors found out that OPM analysts produced significantly less errors than OMT analysts. They explained that OPM users made fewer errors because OPM uses a single model, as opposed to the multiple models in OMT. In their opinion, the source of most of the errors in OMT is a lack of integration among the methods’ different models and the need to maintain consistency and gather information that is scattered across these models. The most significant conclusion of the authors is that the likelihood of making errors increases with the number of models in the method.

5. Comparison of FOOM and OPM—theory and hypotheses

In light of the above comparisons of models/methods, we highlight some similarities and differences of FOOM and OPM methodologies, which are subject of comparison in this study. At the analysis phase, the two methodologies seem to have much in common: both combine and integrate the functional and structural views, utilize ‘DFD-like’ diagrams, apply hierarchical decomposition, and create class diagrams. However, they differ in that OPM diagrams (OPDs) include more symbols and many types of links, as detailed in Section 3 and illustrated in Figs. 4 and 7. Since reality can be modeled equivalently by different models, we believe that a model that uses a simpler notation and consists of less symbol types is easier to learn, comprehend and apply correctly. On one hand, multiplicity of models and corresponding diagramming tools as in UML, may be too complicated, hamper coherent understanding and lead to the production of erroneous models and systems. On the other hand, a single hybrid notation as in OPM must be very rich in order to elicit all points of view, thus may lead to a too complex model of reality. FOOM integrates the functional and object approaches, but it does not use a single notation: for the data model it uses a class diagram, and for the functional model—OO-DFDs. Each of the two uses relatively simple notations, and yet they are synchronized and then utilized in the next phases of development.

Kitchenham et al. [11], who presented guidelines for improving the quality of performing and evaluating empirical research in software engineering, claim that useful research may flow solely from empirical observations, but unfortunately, empirical studies of software engineering phenomena are often contradictory; without any underlying theories, we cannot understand the reason why empirical studies are inconsistent. Moreover, without the link from theory to hypothesis, empirical results cannot contribute to a wider body of knowledge.

Bajaj and Rockwell [1] claim that much of the empirical work in the area of conceptual models evaluation has lacked a theoretical basis for the results. Based on cognitive theories, they propose COGEVAL, a propositional framework to evaluate conceptual models, and try to show how it can be used to explain earlier empirical results as well as
a guiding framework for future empirical work. They organize their framework along two main aspects of conceptual model performance: modeling (quality of specifications, in our terminology) and readability (comprehension, in our terminology). For the purpose of our evaluation and comparison of methodologies, we mention here only some of their propositions.

With respect to modeling (quality), Proposition 2 of their framework, which is based on theories on short-term memory, states that the greater the number of simultaneous items required creating schema segments or chunks, the lower is the modeling effectiveness and efficiency of the model. More specifically, when creating a model schema, if more different aspects of reality need to be considered in order to create chunks or parts of the schema, then some of these items will need to be stored in long-term memory or in some stable storage; this implies both greater possibility of errors, as well as greater effort to produce the schema. Based on this proposition, it may be hypothesized that FOOM would outperform OPM because OPDs consist of more items than OO-DFDs. In particular, OPDs consist of many types of links between objects and processes.

Proposition 4 of COGEVAL states, based on theories on levels-of-processing and long-term memory, that the greater the amount of semantic processing required to create a schema, the less the modeling effectiveness and efficiency of the model. More specifically, since structural processing requires less processing, a model whose constructs require only structural processing to create schemas will offer greater modeling effectiveness and efficiency. Based on this proposition, it may be hypothesized that FOOM would out perform OPM because in FOOM the analyst is concerned at the beginning only with structural modeling, creating only an initial class diagram; then, using the already defined classes; he/she creates the functional model (OO-DFDs). Contrarily, in OPM the analyst is concerned with both functional and structural modeling concurrently, because the two models are interwoven in the OPDs.

With respect to readability, comprehension theory of readability leads to Proposition 8 of COGEVAL, which states that the greater the degree of fragmentation of requirements by the constructs of a model, the lower the readability effectiveness and efficiency. In other words, models whose constructs scatter information across different diagrams will lead to poorer readability. Based on this proposition, it may be hypothesized that OPM would out perform FOOM because it uses only one type of diagram (OPD), while in FOOM there are two: class diagram and OO-DFD. But it must be noted that in spite of having only one diagram type in OPM compared to two in FOOM, actually a schema of a system consists of many diagrams (whether OPDs or OO-DFDs), so at any rate, information is scattered across many diagrams. Besides, an OO-DFD includes also classes which are relevant to the functions appearing in it, so it is not exactly true to claim that in FOOM information is scattered across different diagrams.3

Sections 6 and 7 describe two controlled experiments in which we compared FOOM and OPM on the two dimensions: comprehensibility and quality of schemas.

6. Comparison of FOOM and OPM on comprehension of analysis specifications

6.1. Experimental design

In this experiment, we compared FOOM and OPM analysis specifications from the point of view of users, i.e. comprehension of specifications. Fig. 8 presents the research model. The null hypotheses are: (a) there is no significant difference in user comprehension of the analysis specifications created with the two methodologies; (b) there is no difference in time it takes to complete the tasks of comprehension; (c) there is no difference in user preference of the specifications created with the two methodologies. The alternative hypotheses are that there are significant differences on the three aspects.

6.1.1. Subjects

The experimental subjects used in this study were 126 fourth-year undergraduate students from the Department of Industrial Engineering and Management at Ben-Gurion University (BGU) in Beer-Sheva who took the same courses, including the course ‘Information Systems Analysis’, in which they learned the two methodologies. The emphasis of that learning was to be able to ‘read’ and understand specifications presented in the respective diagrams. The same amount of time was spent on learning the two methodologies. The subjects were motivated to participate in the experiment by considering their grades on comprehension of specifications as part of their final grade in the course.

6.1.2. Tasks

Comprehension of the analysis specifications was measured using two case studies: the IFIP Conference [12] and the Greeting Cards [4]. For each case study we prepared analysis specifications with each methodology. The FOOM specifications included an initial class diagram and a set of three OO-DFDs; the OPM specifications included equivalent OPDs. Figs. 1 and 3 show examples of FOOM specifications for the IFIP Conference case study, and Figs. 4 and 7 show equivalent examples of OPM specifications.

Along with the specifications, we prepared for each case study a questionnaire consists of 40 ‘true’/‘false’ statements

---

3 But this claim seems to be true for UML-based methodologies which employ various diagrams and notations.

4 Preliminary results of this comparison have been reported in [7].
about facts appearing in the diagrams. We distinguished between two categories of statements: statements about ‘structure’ (the data model) and statements about ‘behavior’ (the functional model). About a third of the statements dealt with ‘structure’ and two-thirds—about ‘behavior’. Table 1 shows examples of statements from the IFIP Conference case study. In order to avoid a possible bias arising from the order of statements in the questionnaire, we prepared four different sets of questionnaires for each case study, each with a different ordering of the (same) statements. The questionnaires were randomly distributed to the subjects.

In addition, we prepared a 7-point scale questionnaire to be used after completion of the tasks. This questionnaire was used to measure the degree to which the subjects prefer using each methodology.

6.1.3. Group assignment

The subjects were randomly divided into two main groups. Each subject in each group performed two tasks: to comprehend FOOM specifications for one case study, and to comprehend OPM specifications for the other case study, one at a time. More specifically, each subject in Group ‘A’ received FOOM specifications of the Greeting Cards along with its questionnaire, and OPM specification of the IFIP Conference along with its questionnaire. Subjects in Group ‘B’ received OPM specifications and questionnaire of the Greeting Cards, and FOOM specifications and questionnaire of the IFIP Conference. To avoid possible bias arising from the order of tasks, the subjects in each group were further divided into two subgroups: in each subgroup they started to work with a different case study and model. The division of subjects in groups is shown in Table 2.

The start and end times of each task were recorded to enable measuring the time it took to complete the comprehension tasks. At the end of the experiment, each subject was asked to express his/her subjective preference of each methodology using the 7-point scale, where 1 means that the user absolutely dislikes reading specifications presented in the methodology and 7 means that he/she absolutely likes reading them.

6.1.4. Variables

Following the research hypotheses, we define three dependent variables. The main dependent variable is comprehension of analysis specifications. The auxiliary dependent variables are: time: how long it takes to comprehend the specifications; and preference: which specifications the users prefer.

The independent variables are methodology (FOOM and OPM) and case study (IFIP Conference and Greeting Cards). We treat the case studies as dependent variables because, as we will see, it turned out that they are different in terms of complexity. Recall that the analysis
specifications include both structural (data model) aspects and behavioral (functional model) aspects, and we wanted to find out not only if there is a difference in the overall comprehension of the methodologies’ specifications but also if there is a difference in comprehension of each of the two aspects of the methodologies. The control variables include tasks (all subjects had to perform the same tasks of comprehension), and subjects (all subjects took the same course of studies and were randomly assigned to the treatment groups.

Table 1 Examples of True (T) and False (F) statements—the IFIP Conference

<table>
<thead>
<tr>
<th>Statements in the ‘Behavior’ category:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(F) An organization committee member (OCM) plans all sessions of papers presentations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T) A program committee member (PCM) creates a ‘call for participant and rules’.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) A reviewer’s assignment to a paper is done by a PCM according to reviewers’ interests and the paper’s topic, unless he/she has reviewed more than four papers.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T) One cannot know if a reviewer sends his/her reviews in time.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) A reviewer who completed in time all his/her paper reviews gets an acknowledgement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T) Before a PCM decides to accept or reject a paper, he/she requires seeing the reviewers’ reports.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) A panelist who does not send an acknowledgement is rejected by the system and a message is sent to him/her automatically.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T) The system produces reminders to late reviewers of papers once in a week.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) After the decision of OCM to accept a paper, an appropriate message is sent to its authors, and the papers presentation session is updated accordingly.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T) A panelist who is invited to participate in a panel must confirm his/her participation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) When a PCM schedules an accepted paper to a papers presentation session, the presenters of the paper are assigned.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T) A person who is assigned to chair a paper-presentation session must have confirmed his/her participation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) The conference program is sent to all performing participants and authors.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T) The conference program is sent to all authors whose papers have been accepted.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) The PCM sends reminders to late reviewers.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) The system checks every day the acknowledgements from speakers and sends it to the PCM.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T) When a PCM assigns a paper to a reviewer, the status of both the paper and the reviewer are updated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) A PCM sends reminders to late reviewers.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Statements in the ‘Structure’ category:

<table>
<thead>
<tr>
<th>Statements in the ‘Structure’ category:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(F) A reviewer must review at least two and no more than four papers.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T) A chairperson can chair more than one session.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) An accepted paper must be presented by one author.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T) One can know the start time and end time of a session.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) An author of a paper cannot be a reviewer of another paper.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T) The date, hour and place of a presentation of paper are known.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) Session of papers presentations may not perform simultaneously.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T) A paper must be reviewed by at least two but no more than four reviewers.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) A conference program must consist of at least one session.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T) A conference program consists of sessions that could be performed simultaneously.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) An author of a paper cannot register as regular participant.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.2. Analysis

The level of comprehension of a specification by a user was computed by counting the number of correct answers (whether ‘true’ or ‘false’) given to the statements within each category. We summed the number of correct answers within each category (structure and behavior). Based on that, we computed the average grade of the subjects within each category and their overall grade, for each case study and methodology separately. This enabled us testing the significance of differences between the average grades, per category, case study and methodology. Based on the start and end times of each task completed by each user, we computed the average time to complete the tasks, for each case study and methodology separately. This enabled us testing the significance of differences between the times, per case study and methodology. Finally, based on the users’ replies to the 7-point questions on preferences, we computed the average preference of all users of each methodology, and based on that we tested the significance of difference between the preferences.

6.3. Results

Table 3 presents the results for the main factors: methodology, case study and task order, according to two dependent variables: overall grade and time, using 3-way ANOVA/MANOVA. The results show significant differences in the main effects and insignificant differences in their interactions.

- The results for the main effect methodology show that FOOM is more comprehensible than OPM: FOOM users gained an average grade of 71.45 compared to 68.41 gained by OPM users. Moreover, FOOM users completed the task in less time (60.05 min) compared to OPM users (73.36 min).
- The results for main effect case study show that the Greeting Cards case study was easier/simpler than the IFIP Conference case study: its overall grade of comprehension was higher, and it took less time to comprehend it. Because of the difference between the two case studies, we will later on also present the results of each case study separately (in Table 4).
The result for the main effect task order shows an interesting but reasonable phenomenon: subjects spent more time and scored higher on the first task, regardless of the methodology or case study.

As we can see, there was no interaction among the factors. (Therefore, we will not show the interactions in Tables 3 and 4.)

Table 4 presents the results for each case study separately. As can be seen, in each case study, FOOM scored higher on comprehension and it took less time to comprehend its specifications.

In the above tables we showed the overall mean grades, per case study. In Table 5 we distinguish between comprehension of the ‘structure’ and the ‘behavior’ categories. Surprisingly, in the Greeting Cards case study the advantage of FOOM on both categories is insignificant. In the more complex case study, the IFIP Conference, the FOOM has an insignificant advantage on ‘structure’, but it does have a significant advantage on ’behavior’. In other words, in any case (methodology) we found no significant difference in comprehension of the data models. This can be explained due to the fact that the data models of the two methodologies are not much different: the main difference between them is that in FOOM the attributes are listed within the class rectangles, whereas in OPM they are presented in separate rectangles. This difference turned out to be insignificant with respect to comprehension. On the other hand, FOOM’s functional model turned out to be more comprehensible in the case of the more complex case study (IFIP Conference). This can be explained because in complex/large-scale applications a model which consists of fewer symbols and uses simple notations is more comprehensible, as hypothesized.

The last row of Table 5 presents the results of the subjective preferences of methodologies. We can see a significant preference of FOOM (4.734) compared to OPM (4.00).

To summarize the results of comprehension of specifications experiment, we found that: (a) the analysis specifications of FOOM are more comprehensible than those of OPM, but this is only with respect to the functional model; (b) it takes less time to comprehend FOOM specifications; and (c) users prefer FOOM specifications. AN auxiliary finding is that the first task takes more time than the second, irrespective of the methodology.

7. Comparison of foom and opm on quality of analysis specifications

7.1. Experimental design

In this experiment we take an analyst/designer perspective, attempting to measure and compare the methodologies

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Overall results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor</td>
<td>Values of factor</td>
</tr>
<tr>
<td>(1) Methodology</td>
<td>OPM</td>
</tr>
<tr>
<td></td>
<td>FOOM</td>
</tr>
<tr>
<td>(2) Case study</td>
<td>Greeting Cards</td>
</tr>
<tr>
<td></td>
<td>IFIP Conference</td>
</tr>
<tr>
<td>(3) Task order</td>
<td>1st</td>
</tr>
<tr>
<td></td>
<td>2nd</td>
</tr>
<tr>
<td>Interaction 1,2</td>
<td>1.147</td>
</tr>
<tr>
<td>Interaction 1,3</td>
<td>0.123</td>
</tr>
<tr>
<td>Interaction 2,3</td>
<td>1.055</td>
</tr>
<tr>
<td>Interaction 1,2,3</td>
<td>0.842</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Results for each case study</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case</td>
<td>Category</td>
</tr>
<tr>
<td>Greeting Cards</td>
<td>Overall grade</td>
</tr>
<tr>
<td></td>
<td>(%)</td>
</tr>
<tr>
<td></td>
<td>Time</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>IFIP Conference</td>
<td>Overall grade</td>
</tr>
<tr>
<td></td>
<td>(%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5</th>
<th>Results of structure and behavior categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case</td>
<td>Category</td>
</tr>
<tr>
<td>Greeting Cards</td>
<td>Structure</td>
</tr>
<tr>
<td></td>
<td>Behavior</td>
</tr>
<tr>
<td>IFIP Conference</td>
<td>Structure</td>
</tr>
<tr>
<td></td>
<td>Behavior</td>
</tr>
<tr>
<td>Preference of methodology (1 is least; 7 is most)</td>
<td>OPM</td>
</tr>
</tbody>
</table>
on the dimension of quality of specifications Fig. 9 presents
the research model. We hypothesize that the analysis
specifications in FOOM would be of better quality than
those in OPM.

7.1.1. Subjects
This experiment involved students from two different
universities: BGU University in Beer-Sheva and the
Technion in Haifa. The subjects of the FOOM group
(from BGU) were 56 students of Information Systems
Engineering who took the same courses, including
‘Systems Analysis and Design’, where they studied and
exercised with FOOM. The subjects of the OPM group
(from the Technion) were 100 undergraduate students of
Information Systems Engineering who took the same
courses, including ‘Systems Analysis and Specification’,
where they studied and exercised with OPM. The
experiment took place in both institutes at the end of
the same semester. The subjects were motivated by
considering their grades on quality of specifications as
part of their final course grades.

7.1.2. Tasks
The task of each subject in each group was to prepare
analysis specifications for one case study, based on user
requirements presented to the subject in a textual form, i.e. a
requirements document. We used the requirements docu-
ment for the IFIP Conference case study. The task of
subjects in the FOOM group was to create an initial class
diagram (data model) and hierarchical OO-DFDs (func-
tional model). The task of subjects in the OPM group was to
create OPDs. In order to enable comparing the quality of the
results while neutralizing other possible variables, we
instructed the subjects who used FOOM to create an initial
class diagram and exactly three OO-DFDs: a root OO-DFD
and two sub-level OO-DFDs. The subject who used OPM
where instructed to create an equivalent number of OPDs.

Obviously, a requirements document includes both
structure (data) requirements and functional requirements.
There is reason to assume that the order in which the
different requirements are presented to the analyst may have
an impact on the quality of the analysis products. Therefore,
we prepared four different versions of the same require-
ments document, each presenting the same requirements in
a different order: In version (1) of the requirements
document we presented all data structure requirements
first, followed by all functional requirements—marked DF.
In version (2) the requirements were grouped by the main
data structures and within each the relevant functional
requirements—marked DFDF. In version (3) the require-
ments were grouped by the main functions and within each
the data structures they use—marked FDFD. In version (4)
we presented all functional requirements, followed by all
data structure requirements—marked FD; in other words—
versions 1 and 4 are entirely in opposite orders.

7.1.3. Group assignment
The subjects in each group (BGU and Technion) were
divided randomly into two four subgroups, such that
members in each subgroup obtained a different version of
the requirements document Table 6 presents the division of
subjects into these groups. When the experiment began each
subject received a requirements document in one of the four
versions, and empty solution sheets to be used for drawing
their diagrams.
7.1.4. Variables

The dependent variable is quality, which is measured according to the correctness of the specifications created by the analysts. The two independent variables are methodology (FOOM and OPM) and the order of requirements. The control variables include the tasks and the subjects.

7.2. Analysis

In order to be able to test out the quality of each analysis specification, the authors of this paper and two experts in OPM created a unified grading scheme to measure the correctness of specifications in both methodologies. The scheme details the possible error types and the number of points to be deducted due to each error, according to its severity. The grading scheme was agreed upon by the two parties before the experiment began. Table 7 presents the grading schemes.

The quality of analysis specification created by each subject was measured by marking all errors he/she made compared to the requirements document, and counting the total number of the respective errors points. Each diagram created by a subject was graded separately, so we obtained one grade for the data model and three grades for the functional models (OO-DFDs or OPDs). Then we computed one grade of the functional model as average of the grades of the separate diagrams.

In order to verify that there is no bias in grading the FOOM specifications, 12 sets of solutions were randomly selected and given to two independent examiners, one of them an expert in OPM and the other an expert in FOOM. Obviously, both examiners used the same grading scheme. Similarly, in order to verify that there is no bias in grading the OPM specifications, fifty sets of solutions were randomly selected and given to two independent examiners who used the same grading scheme. Comparison of the grades of the different graders revealed that in both cases there were no significant differences between the grades of the independent graders.

7.3. Results

Tables 8 and 9 present the grades of the dependent variable quality of analysis specifications according to the effects caused by the two independent variables: methodology and order of requirements. We used 2-way ANOVA to analyze the results, in order to find the influence of each independent variable alone and their interaction. In each table, the first column presents the factor (i.e. each independent variable) and their interaction; the second column presents the values of each factor; the third column presents the mean grades; the fourth presents the F statistic of the independent variable; the fifth presents the P value (at α = 0.05); and the last column notes if the results are significant. The first row presents the effect of the methodology (FOOM or OPM) on the grade; the second row presents the effect of the order of requirements; and the third row—the interaction of the two.

Table 8 presents the effect caused by the two independent variables on the mean grades of the data models. The first row presents the effect caused by the methodology. As can be seen, there is a significant difference in favor of FOOM (average grade 65) compared to OPM (average grade 57.29).

The second row presents the effect of the order of requirements. It can be seen that there is a significant difference between the mean grades of the data models, meaning that the order in which user requirements are presented to analysts affects the quality of the data model produced. More specifically, subjects who used version (4) requirements, i.e. where the functional requirements are

---

Table 6

<table>
<thead>
<tr>
<th>Groups</th>
<th>FOOM group (BGU)</th>
<th>OPM group (Technion)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version of requirements doc.</td>
<td>1 2 3 4</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>No. of subjects per subgroup</td>
<td>16 14 13 13</td>
<td>25 24 26 25</td>
</tr>
<tr>
<td>Total no. of subjects per group</td>
<td>56</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 7

<table>
<thead>
<tr>
<th>Errors in OO-DFDs or OPDs</th>
<th>Points</th>
<th>Errors in class diagram (FOOM) or data structure elements (OPDs)</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing function</td>
<td>−4</td>
<td>Missing object</td>
<td>−3</td>
</tr>
<tr>
<td>Redundant function</td>
<td>−2</td>
<td>Redundant object</td>
<td>−2</td>
</tr>
<tr>
<td>Process without an input or output</td>
<td>−2</td>
<td>Incorrect use of inheritance</td>
<td>−2</td>
</tr>
<tr>
<td>Missing or redundant dataflow/link; missing label</td>
<td>−1</td>
<td>Missing or incorrect attribute</td>
<td>−1</td>
</tr>
<tr>
<td>Wrong connection (e.g. between classes)</td>
<td>−1</td>
<td>Incorrect notation</td>
<td>−1</td>
</tr>
<tr>
<td>Incorrect dataflow/link (e.g. wrong connection b/w functions; dataflow/link in wrong direction)</td>
<td>−1</td>
<td>Incorrect relationship, missing or incorrect cardinality of relationship</td>
<td>−1</td>
</tr>
<tr>
<td>Function not in the right diagram</td>
<td>−1</td>
<td>Missing or redundant relationship</td>
<td>−1</td>
</tr>
<tr>
<td>Missing relationship attribute (FOOM)</td>
<td>−1/2</td>
<td>Missing relationship attribute</td>
<td>−1/2</td>
</tr>
</tbody>
</table>
presented before the data requirement, yielded significantly higher grades than those who used other versions of requirements. Note that this result is irrespective of the methodology used. The third row shows that there is no significant interaction between the methodology and the order of requirements.

Table 9 presents the effect caused by the two independent variables on the mean grade of the functional models. The first row presents the effect caused by the methodology. As can be seen, here again, there is a significant difference in favor of FOOM: mean grade of 83.9 in FOOM compared to 72.37 in OPM.6

The second row in Table 9 shows that there is no significant difference between the mean grades of the functional models with respect to the different orders of user requirements. However, here too we can see that when the requirements document begins with functional requirements (version 4) the grades are higher than in other cases. So, again, it seems that there is some (though insignificant) advantage in presenting functional requirements first. The third row in the table shows that there is no significant interaction between the methodology and the order of requirements.

To summarize the results of this experiment, we found that the quality (correctness) of analysis specifications created in FOOM is significantly better than the quality of specifications created in OPM. This is true for both the data model and the functional model. These results are in line with our earlier discussion on the differences between the models, and support the earlier results on user comprehension and preference of the methodologies. In addition, we found that there is an advantage (though not always significant) if the functional requirements are presented to the analysts before the data requirements.

8. Threats to validity

This section discusses various threats to validity and the way we attempted to alleviate them in both experiments. Parts of the discussion are based on [3,6,11].

8.1. Construct validity

Construct validity is the degree to which the independent and dependent variables accurately measure the concepts they purport to measure. Threats to construct validity have been identified in both experiments:

(a) In the ‘comprehension’ experiment, a threat to validity is the measure of understandability. To alleviate the threat the questionnaire used for each case study consisted of 40 true/false statements addressing many facts appearing in the specifications. To be more specific about understandability, we measured separately understandability of structures (the data models) and understandability of behavior (the functional models). Of course, counting and summing the numbers of correct answers is not a perfect measure because we give an equal weight to all statements. In spite of the inherent weakness, we argue that measure of understandability we used is reasonable. Similar measures have been used in earlier comparative experiments of comprehension, as surveyed in Section 4.

(b) In the ‘quality’ experiment we measured quality of specifications by counting and summing the number of errors points deducted. One threat to validity is the grading scheme we defined: we cannot prove that it

---

6 Note that the grades of the functional model (Table 9) are significantly higher than the grades of the data model (Table 8) in both methodologies. A possible explanation to this is that the functional requirements were more clear or simple, or that the subjects were better in functional modeling than in data modeling, or that the grading scheme of the data model was more severe. At any rate, this difference does not bias the comparative results since we do not combine the results of the two models.
perfectly measures quality. To minimize the threat, experts of the two methodologies worked together to define the error types and their severity. Another threat to validity is that the count of error points in analysis specifications is not the ultimate measure of quality, only a surrogate. The ultimate measure of quality would have been quality of the end product, i.e. the working system developed according to the specifications. But this is not feasible to achieve in an experimental setting. Again, we note that a grading scheme is a common tool to measure and compare quality of methodologies.

(c) Another threat to construct validity in the ‘quality’ experiment is that we gave each subject (analyst) a requirement document of a certain case study (the IFIP Conference system). This does not represent a realistic situation: in reality analysts usually elicit requirements from real users or their representatives, in an interactive and iterative process. In the experiment we actually skipped this important aspect of system analysis, so it can be argued that we mainly measured how well subjects who play the role of analysts convert predefined textual requirements into diagrammatic specifications. We cannot be sure that same results would be obtained had the subjects being worked in a more realistic environment. It should also be noted that in reality the order of user requirements cannot be controlled as we did in this experiment. But the results of the experiment give us some evidence about the preferred order in which requirements should be presented to analysts.

(d) We treated the subjects as a controlled variable, assuming that there are no differences between them, because they were all students of Information Systems Engineering. But we have no proof that indeed the students in the two groups are similar in all dimensions, because they came from two different universities. For example, they did not take exactly the same courses, with the same teachers, in the same classes. But it was not realistic to use entirely similar subjects because in none of the universities the students are taught analyzing systems with the two methodologies.

(e) There is a potential bias in the results of the experiments due to the fact that the authors of this paper, who developed FOOM, played an active role in preparing the experiments and teaching participants. We follow Kitchenham’s et al. recommendation [11] that if one cannot avoid evaluating his own work, he/she should report what has been done to minimize bias. In the ‘comprehension’ experiment we minimized bias by spending the same amount of time in teaching the subjects to read and understand the two methodologies. In the ‘quality’ experiment the threat is less severe because the OPM analysts were taught by the OPM experts, while we taught only the FOOM analysts.

In addition, the specifications of each methodology were graded by experts of each methodology.

8.2. Internal validity

Internal validity is the degree to which conclusions can be drawn about causal effect of independent variables on the dependent variables. The following possible threats have been identified:

(a) For both experiments, a confounding effect between the methodologies and the problem domains used would mean that the subjects’ performance was affected by both these variables. This may occur because one problem domain is more familiar to participants or is more suited to a methodology. To help counter this threat, the problem domains we used were taken from textbooks and papers. As such, these domains were deemed to be amenable to any methodology. In addition, based on the courses of study of the participants, there was no reason to believe that they would be more or less familiar with any of the application domains.

(b) An instrumentation effect may result from differences in the experimental materials employed. The threat to the ‘comprehension’ experiment was a possible difference between the two case studies. To help counter this threat, we measured comprehensibility of the two case studies separately. Indeed, we found out that there is a difference between the two case studies, and this led us to conclude that there is no difference in comprehension of structures in the two methodologies, irrespective of the problem complexity, but there is a difference in comprehension of behavior, such that one methodology (FOOM) has an advantage in the case of the more complex system.

(c) Another possible threat to internal validity in the ‘quality’ experiments is the structure of the tasks. We asked each participant to create exactly three diagrams for the functional model (OO-DFDs or OPDs). We did so because we wanted to be able to compare the results, neutralizing other possible effects. We cannot be sure how good could the results (specifications) be had we let the participants to create as many diagrams as they like. On the other hand, there is no reason to assume that the instructions given to the participant had any bias on the quality of specifications created with in methodology.

8.3. External validity

External validity is the degree to which the results of the experiment can be generalized to the population under study
and other research settings. The following possible threats have been identified:

(a) The case studies used in the experiments are relatively small and may not be representative in terms of their size and complexity. This threat is common to almost any controlled experiment involving comparison of methodologies.

(b) The subjects who participated in the ‘comprehension’ experiment represent users or managers who read and evaluate analysis specifications, but they were not real users or managers who indeed evaluate specifications of systems developed for them. To help counter this threat, we used for this experiment students of Industrial Engineering and Management, not professional systems analysts. We have no reason to assume that the use of such students, who were randomly assigned to the treatment groups, might have caused any bias of the comparative results. In the ‘quality’ experiment we again used students rather than real analysts, who face real users and requirements; but in this experiment the subjects were students of Information Systems Engineering in the last year of their studies, and they were trained to become systems analysts. Each subject learned his/her methodology (FOOM or OPM) from an expert in that methodology. It should be noted again that in almost all experimental research on method evaluation the subjects were students, because it is not feasible to conduct such experiment with experienced analysts and real-world systems.

In further research, we plan to conduct more experimental comparisons of FOOM with other OO methodologies, and to include not only the analysis specifications but also the design products. The problem with this is to use other methodologies whose analysis or design products can be compared, as we could in these experiments. Obviously, in further experiments we plan to utilize different case studies with varying degrees of complexity.

9. Conclusions and further research

Based on the results of the two comparative experimental we conclude that: (a) the analysis specifications of FOOM are more comprehensible than those of OPM, but this is only with respect to the functional model; (b) it takes less time to comprehend FOOM specifications; (c) users prefer the specifications of FOOM; (d) the quality (correctness) of the analysis specifications created using FOOM is significantly better than those created with OPM; this is true for both the data model and the functional model; (e) there is an advantage (though not always significant) if the functional requirements are presented to the analysts before the data requirements.

As said, in the two experiments we used relatively small case studies, and the subjects were students, not real users or analysts. These limitations are common to almost all experimental studies on model/method evaluation and comparison. Another limitation worth mentioning again is that we measured comprehensibility and correctness of analysis specifications, whereas in reality the ultimate measure of success is performance of real/working systems.

References


