On the Usefulness and Ease of Use of a Model-Driven Method Engineering Approach

Mario Cervera*, Manoli Albert, Victoria Torres, Vicente Pelechano

Centro de Investigación en Métodos de Producción de Software
Universitat Politècnica de València
Camino de Vera s/n, 46022 Valencia, Spain

Abstract

The Method Engineering (ME) discipline emerged as a response to the need for methods that are better adapted to context. Despite the potential benefits of ME and the emergence of Computer-Aided Method Engineering technology, there are hardly any reports on the practical application of ME available in the literature. Some authors argue that this is because practitioners often fail to see the usefulness of ME due to its high complexity. With the aim of facilitating the application of ME, we developed MOSKitt4ME, a lightweight approach that makes intensive use of reusable assets and Model-Driven Engineering. In previous work, we illustrated how MOSKitt4ME supports three phases of the ME lifecycle: design, implementation, and execution. In this paper, we evaluate the complexity of MOSKitt4ME. Specifically, we present a study that is based on the Technology Acceptance Model (TAM) and the Think Aloud method. The TAM allowed us to measure usefulness and ease of use in a subjective manner; the Think Aloud method allowed us to analyze these measures objectively. Overall, the results were favorable. MOSKitt4ME was highly rated in perceived usefulness and ease of use; we also obtained positive results with respect to the users' actual performance and the difficulty experienced.

Keywords:
Method Engineering, Model-Driven Engineering, Complexity, Think Aloud, Technology Acceptance Model

*Corresponding author. Tel.: +34 96 387 70 00x73564.
E-mail addresses: mcervera@pros.upv.es (M. Cervera), malbert@pros.upv.es (M. Albert), vtorres@pros.upv.es (V. Torres), pele@pros.upv.es (V. Pelechano).

1. Introduction

Software projects are diverse in nature. They differ, for example, in size, application domain, or team expertise. Due to these differences, it is generally agreed that software companies must define their methods in-house [1, 2, 3]; thus, these methods can be adapted to the needs of specific projects. To define methods efficiently and effectively, companies require systematic solutions that are built upon sound methodical foundations. Providing these solutions is the main goal of the Method Engineering (ME) discipline [4]. By adopting ME, companies gain flexibility in building project-specific methods [5, 6], and since these methods are defined in-house, developers are motivated to use them due to the feeling of method ownership [7].

Regardless of the potential benefits of ME and the emergence of Computer-Aided Method Engineering (CAME) technology [8], ME has never been widely used in industry [9, 10]. Kuhrmann *et al.* concluded in a recent mapping study [11] that there are hardly any reports on the practical application of ME available in the literature. Henderson-Sellers *et al.* argue in [2, 12] that practitioners often fail to see the usefulness of ME mainly due to its complexity and cost in terms of time, money, and people. The complexity of ME was also noted by Ter Hofstede *et al.* [13], who identified several complexity issues related to the selection, storage, retrieval, and assembly of method fragments.

With the aim of facilitating the use of ME, we developed MOSKitt4ME, a ME approach that is fully implemented by a CAME environment [14]. MOSKitt4ME differs from traditional ME in that it is lightweight: MOSKitt4ME is built upon reusability principles and it is also model-driven, which enables a high level of automation. In our previous work [15, 16], we illustrated how MOSKitt4ME makes intensive use of reusable assets and Model-Driven Engineering (MDE) to support three phases of the ME lifecycle: the initial design of the method, its implementation, and the final method execution. In this paper, we present an evaluation study that focuses on the complexity of MOSKitt4ME.

The study that is presented in this paper evaluates MOSKitt4ME by means of the Technology Acceptance Model (TAM) [17] and the Think Aloud method [18]. The TAM allowed us to assess the subjective perception of users with respect to two quality attributes: usefulness and ease of use. We evaluated perceived ease of use because this attribute represents a subjective measure of complexity [19, 20]. We evaluated perceived usefulness because this attribute is causally affected by perceived ease of use [21], and, for this reason, the usefulness of ME is often negatively perceived by practitioners (which represents a major obstacle for the success of ME and CAME technology). To reinforce the subjective results that were obtained by means of the TAM, we also evaluated usefulness and ease of use in an objective manner. To

this end, we analyzed the actual improvement in performance that MOSKitt4ME users achieved during the study and also the difficulties that they experienced[1]. Performance was assessed by measuring efficiency and effectiveness. Difficulty was assessed by analyzing the users' reasoning processes, which reveal the errors made by the users, the doubts that they experienced, and the problem-solving strategies that they followed, among other data. To analyze this data at the highest possible level of detail, we applied the Think Aloud method.

In summary, the contribution of this paper is the thorough evaluation of a model-driven ME approach (MOSKitt4ME) from both a subjective and an objective perspective. The main goal of this evaluation is to illustrate that MOSKitt4ME can be positively rated in terms of perceived usefulness and ease of use and that MOSKitt4ME can also improve the users' performance while posing little difficulty of use. Our positive results contrast with traditional ME, whose usefulness is often negatively perceived by practitioners and whose complexity remains an unsolved issue. As a collateral benefit of the study, we also illustrate how MOSKitt4ME reduces the complexity of ME by means of MDE techniques, which alleviate the users' workload in three phases of the ME lifecycle: design, implementation, and execution.

The remainder of the paper is structured as follows. Section 2 discusses related work and Section 3 summarizes our model-driven ME approach. Then, Section 4 provides an overview of the evaluation study. Each of the four phases that comprise the study are detailed in Sections 5, 6, 7, and 8, respectively. Finally, Section 9 presents some conclusions and outlines future work.

2. Related Work

In 1996, Tolvanen *et al.* [22] noted that ME researchers had focused mostly on the theoretical foundations of the discipline and highlighted the need for investigating usability issues such as usefulness or complexity. A similar conclusion was reached in 1997 by Ter Hofstede *et al.* [13], who stated that more empirical research was needed to substantiate the claims associated with the potential benefits of ME. Despite these demands for more empirical research, two decades later it is still hard to find empirical studies that investigate methods and tools for ME [11].

One of the few empirical studies that have been conducted in the context of ME is the work by Sousa *et al.* [23]. This work evaluates the graphical notation of a language for method design: the ISO/IEC 24744 standard [24]. The main contributions of this work are suggestions for improving the notation. Other studies are those by Kelly *et al.* [25] and Kerzazi *et al.* [26]. The former evaluates an approach

[1]According to Davis [17], perceived usefulness and perceived ease of use are the people's subjective appraisal of performance and effort/difficulty, respectively.

for testing metaCASE environments; this approach is based on an error classification that allows the performance of metamodelers to be measured. The latter evaluates the usability of two method design tools: EPF Composer and DSL4SPM.

In a more theoretical context, we can find two ME approaches that take complexity into consideration. In [9], Bajec *et al.* present the Process Configuration Approach (PCA), which was conceived to be simple enough to be adopted by software companies. The general idea of the PCA is that project-specific methods are designed by selecting components from a base method. On the other hand, in [3] Karlsson *et al.* propose the Method for Method Configuration (MMC). The MMC is based on the notion of method component [27], which combines ME with activity theory to make ME less cumbersome.

In addition to the above research efforts, which deal with usability issues, we can also find empirical studies that concern other aspects of ME. For instance, Qumer *et al.* [28] tested the applicability of a framework for assessing method agility, while in [29] Karlsson describes the lessons learned in the evaluation of a wiki-based approach for method tailoring. On the other hand, Seidita *et al.* [30] performed a study where they tested their approach for the design of agent-oriented methods.

The analysis of all the aforementioned studies allowed us to identify two important limitations. First, most of the empirical research that has been performed in the ME field only investigates the method design phase of the ME lifecycle; thus, the method implementation and execution phases are almost completely neglected. Second, even though some authors take complexity into consideration [3, 9, 13], none of them provide a detailed empirical analysis of the usefulness and ease of use of a ME approach when it is put into practice by means of a supporting CAME environment. In order to fill these gaps, our study makes a detailed analysis of the usefulness and ease of use of a model-driven ME approach (MOSKitt4ME) when it is put into practice during three phases of the ME lifecycle: design, implementation, and execution.

3. Model-Driven Method Engineering: the MOSKitt4ME Approach

Following the definition of ME that was given by Brinkkemper in [4], we define model-driven ME as a paradigm for ME where models play a key role in the design, construction, and adaptation of methods, techniques, and tools for the development of information systems.

The model-driven ME approach that is implemented in MOSKitt4ME makes intensive use of MDE techniques (e.g., metamodeling, model transformations, and models at runtime) to support the design, implementation, and execution of methods. These three phases of the ME lifecycle are depicted in Figure 1 and are summarized below. For further details, we refer the reader to our previous work [15, 16].

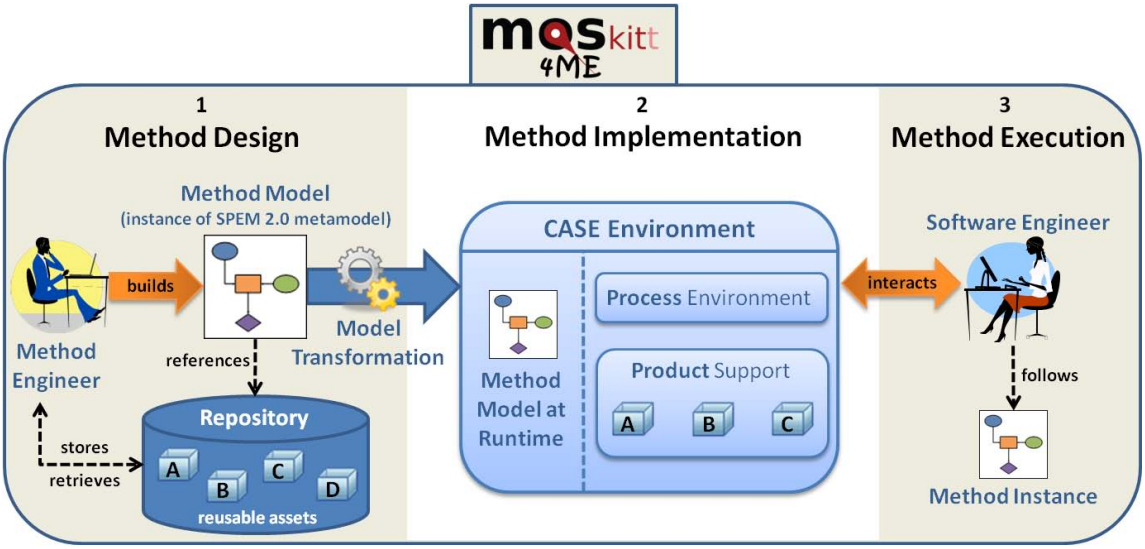

Figure 1: Overview of MOSKitt4ME

3.1. Method Design

The method design involves the creation of a *method model* by means of the instantiation of the SPEM 2.0 metamodel [31]. The method model specifies (among other elements) the tasks to be carried out, the people that participate in these tasks, and the products to be developed to reach the final system. Method engineers must link these elements with *reusable assets* that are stored in a repository. These assets contain technical data; that is, software tools such as textual or graphical editors. Thus, a software tool that is associated to a method element will support this element during the method execution; for instance, a UML editor that is associated to a product called "Class model" will support the creation of specific instances of this product (i.e., specific UML class models). The repository of MOSKitt4ME is an important advantage of our tool due to the assets that it already contains. Since the only requirement is that these assets be implemented as Eclipse plug-ins, we could incorporate tools developed by the Eclipse community. In addition to the starting set of assets, method engineers can increase the population of the repository by using the metatools that are integrated in MOSKitt4ME (e.g., the Eclipse Graphical Modeling Framework [32], which enables the construction of graphical editors).

3.2. Method Implementation

In this phase, a *CASE environment* that supports the method is automatically obtained by means of a *model transformation*. This CASE environment includes a *process environment* as well as software support for the creation and manipulation of the method *products*. The process environment (which is always included in the

CASE tool regardless of the method that has been specified) provides a graphical user interface and a process engine that interpret the *method model at runtime* to assist software engineers during the method execution. The software support for the method products is obtained from the reusable assets that were linked to the method elements during the design phase.

3.3. Method Execution

The method execution involves the enactment of *method instances* (in specific development projects) using the CASE environment that is obtained in the previous phase. In this CASE environment, the method execution is assisted by the process environment, which indicates the tasks that are executable and the tools to be used in these tasks. When the tools do not require human participation, the process environment automatically starts the tool execution; otherwise, it provides guidance on the use of the tools. This functionality allows software engineers to follow the method prescriptions more easily and it also partially automates the development process. In addition to the process-related assistance, the process environment also allows software engineers to keep track of the method products; to this end, it provides a hierarchical view that classifies the products according to the categories that are defined in the method model.

4. Overview of the Evaluation Study

Even though MOSKitt4ME proves (by construction) that model-driven ME can be implemented, the benefits of model-driven ME must be demonstrated via rigorous evaluation methods. For this reason, we performed a study that evaluates MOSKitt4ME with respect to two quality attributes: usefulness and ease of use.

4.1. Measures of Usefulness and Ease of Use

Figure 2 summarizes the measures of usefulness and ease of use that are employed in our study. As the figure shows, our study employs two types of measures: subjective and objective. The use of two types of measures has two main advantages [33]. First, since each type of measure may lead to different conclusions, obtaining similar results reinforces the evaluation study. Second, the combination of two types of measures provides a more complete picture of the phenomenon that is studied.

The subjective measures that are used in our study evaluate the users' satisfaction with MOSKitt4ME. Similarly to most usability studies (which use questionnaires to quantify satisfaction [33]), we used two questionnaires; specifically, the questionnaires defined by the TAM [17]. The TAM is the most widely applied model for evaluating usefulness and ease of use in a subjective manner [34, 20]. This evaluation is done through two measures: *perceived usefulness* and *perceived ease of use*.

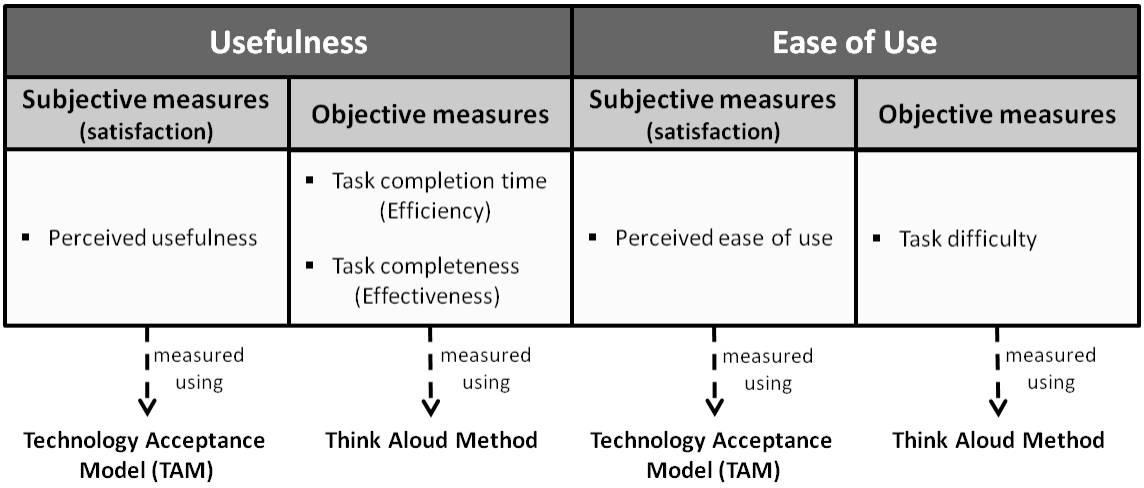

Figure 2: Measures used in the evaluation study

On the other hand, the objective measures of our study evaluate the performance of MOSKitt4ME users. Specifically, we measured *task completion time* (which is a measure of efficiency) and *task completeness* (which is a measure of effectiveness). These measures quantify the usefulness of MOSKitt4ME in the sense that our tool can be considered useful if it improves performance. Additionally, to evaluate the ease of use of MOSKitt4ME, we measured *task difficulty*. Unlike the other two measures, this measure was tested qualitatively; specifically, it was tested in terms of the challenges (or difficulties) faced by the users during the execution of the tasks of the study. These challenges disclose the complexity of MOSKitt4ME, and, consequently, they can be considered to be an objective appraisal of the ease of use of the tool.

The objective measures of our study were tested through direct observation [35] since this type of evaluation method provides the most in-depth understanding of the phenomenon under study [36]. Of all the methods based on direct observation, we selected the Think Aloud method [18] because it is the most systematic and valid [35, 37]. This method gathers data while a real user-system interaction is taking place, thus avoiding the problems of interviews and questionnaires. Note that questions to the user may be biased due to the tendency of people to describe their behavior in terms of formal methods that deviate from their real actions [18].

4.2. Experimental Process

For the evaluation of MOSKitt4ME, we followed the guidelines for experimentation in software engineering proposed by Wohlin *et al.* in [38]. Based on these guidelines, we performed four sequential phases: (1) definition and planning, (2) ex-

ecution, (3) data analysis, and (4) results. First, we established the scope of the study (by defining its goal) and its planning (i.e., how the study is conducted: subjects, research questions, etc.). Second, we executed the study with the subjects in order to collect the data to be analyzed. Third, we analyzed the collected data. Finally, the responses to the research questions were elaborated using the results obtained from data analysis. These four phases are detailed in the following sections.

5. Definition and Planning

This section details the first phase of the study. In this phase, we defined the goal of the study as well as the research questions, subjects, objects, factors, tasks, context, instrumentation, experimental setup, and validity evaluation.

5.1. Goal

The goal of the study is to evaluate two attributes of MOSKitt4ME: usefulness and ease of use. Following the template for goal definition that is suggested in [38], the goal of our study can be summarized as follows:

Analyze MOSKitt4ME

For the purpose of evaluation

With respect to usefulness and ease of use

From the point of view of the researcher

In the context of academia and industry

5.2. Research Questions

To achieve the goal of the study, we defined four questions that guided our research. The first two research questions (RQ1 and RQ2) focus on the subjective perception of users; specifically, RQ1 investigates perceived usefulness and RQ2 investigates perceived ease of use.

RQ1. What is the users' perceived usefulness of MOSKitt4ME?

RQ2. What is the users' perceived ease of use of MOSKitt4ME?

The next research questions (RQ3 and RQ4) focus on objective measures; specifically, RQ3 investigates the actual improvement in performance that is provided by MOSKitt4ME and RQ4 explores the actual difficulties faced by MOSKitt4ME users.

RQ3. To what extent does MOSKitt4ME enhance efficiency and effectiveness?

RQ4. To what extent can MOSKitt4ME be used free from difficulty?

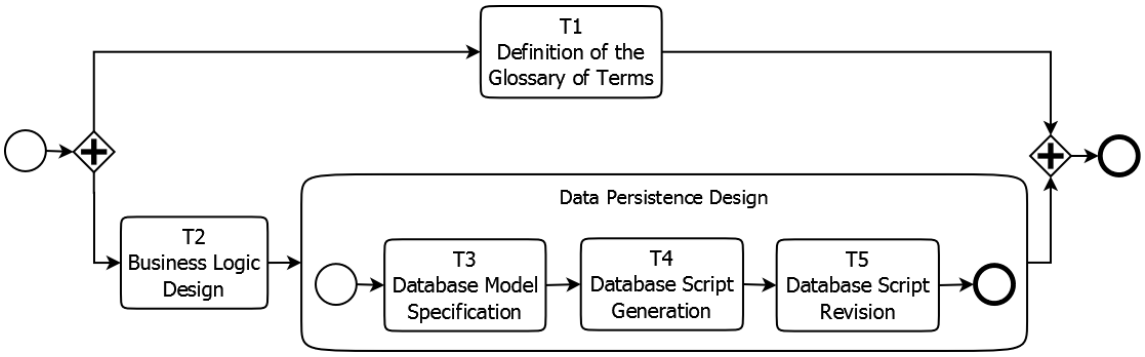

Figure 3: The object of the study

Table 1: Method details

Id	Inputs	Outputs	Roles	Tools
T1	None	Glossary model	Designer	Glossary editor
T2	None	UML 2.0 model	Designer	UML 2.0 editor
T3	UML 2.0 model	Database model	Designer	Database editor
T4	Database model	DDL Script	System	DB2DDL
T5	DDL Script	DDL Script	Developer	None

5.3. Subjects

Software developers are the population of interest for this study; in practical settings, they are the performers of the methods and they often work as casual method engineers. The study does not require expert developers, but subjects must have basic knowledge in software development methods: design of method models, implementation of tools that support methods, and execution of methods in development projects. Additionally, we require subjects to be familiar with Eclipse and MDE.

5.4. Object

The object that was selected for the study is a part of gvMétrica [39]: the method that is used at the Valencian Regional Ministry of Infrastructure, Territory, and Environment. The object selection was carried out with a twofold purpose in mind. First, we aimed to find a simple, understandable, and realistic scenario that included enough elements (e.g., tasks, roles, and products) for the complete use of MOSKitt4ME. Thus, subjects could use all of the MOSKitt4ME functionality without being affected by the excessive complexity of the selected object. Second, we aimed to minimize the threat of maturation [38] (see Section 5.10.2).

Figure 3 shows the object of the study; Table 1 contains details about the method tasks. The first task of the method is to build a glossary model, which defines the

terms involved in the system design. This model is built by a designer using a glossary editor. In parallel, the designer defines the business logic of the system by means of a UML 2.0 editor. Then, based on the UML 2.0 model, the designer defines a model of the database schema using a database editor. The database model enables the generation of the code that implements the schema in terms of a Data Definition Language (DDL). This generation is performed by the DB2DDL transformation. Finally, a developer revises the generated DDL script. The description of the method (as handed out to the subjects) can be found in [14].

5.5. Factors and Treatments

Our study applies a paired comparison of one factor (*ME approach*) with two treatments (*None* and *MOSKitt4ME*) [38]. In this design, both treatments are applied by all of the subjects of the study. When the subjects apply *None*, they perform the tasks of the study without using ME techniques; when they apply *MOSKitt4ME*, they perform the same tasks using MOSKitt4ME. Thus, subjects can contrast using MOSKitt4ME with not using any ME approach. Also, we can compare the subjects' performance using MOSKitt4ME with their performance without the tool.

5.6. Tasks

The study is divided into two parts – one for each treatment. Below, we describe the tasks to be performed by the subjects in each of these parts. The task descriptions (as handed out to the subjects) can be found in [14].

Treatment 1. ME approach = None.

- **Task 1.1. Method Design/Implementation.** We provide the subjects with a printed document containing the method presented in Section 5.4. Since the subjects do not have any method editor available, they do not perform the method design; instead, they build a supporting CASE environment. To do this, we give the subjects access to a repository that contains software tools (e.g., editors and model transformations). The challenge lies in manually integrating into the same Eclipse installation only the tools that are strictly necessary to support the method. This can be accomplished by copying the tools into the dropins folder of Eclipse and solving the dependency problems that appear. All of the tools required to solve the dependency problems can be found in the repository.
- **Task 1.2. Method Execution.** The subjects use the CASE environment built in Task 1.1 to run a development project. During the course of this project, the subjects must follow the method, executing the tasks in the correct order. To do this, the subjects can only use the printed document as assistance.

Treatment 2. ME approach = MOSKitt4ME.

- **Task 2.1. Method Design/Implementation.** We provide the subjects with a printed document containing the method presented in Section 5.4. The subjects must use this document to create a model of the method by means of MOSKitt4ME. To enable the definition of the method technical data, we give the subjects access to a repository that contains reusable software tools (e.g., editors and model transformations). When the method model is finished, MOSKitt4ME allows the subjects to automatically obtain the supporting CASE environment.
- **Task 2.2. Method Execution.** The subjects use the CASE environment generated in Task 2.1 to run a project. During the course of this project, the subjects must follow the method, performing the tasks in the correct order. This is facilitated by the process environment that is integrated in the CASE tool.

5.7. Context

The evaluation study was executed in an academic context; specifically, in a teaching laboratory of the *Departamento de Sistemas Informáticos y Computación* (DSIC) at the *Universitat Politècnica de València* (UPV).

5.8. Instrumentation

We used five instruments during the execution of the study:

Printed document. We provided subjects with a document containing the description of the method proposed in Section 5.4 and also the tasks of the study. After each task description, the document requests the mental effort invested in the task. Mental effort ranges from "very low" (0) to "very high" (6).

Characterization form. This form requests demographic data and quantifies the subjects' experience (see [14]). We took experience into consideration since it influences perceived usefulness and perceived ease of use [20].

User acceptance form. This form quantifies perceived usefulness and perceived ease of use (see [14]). We developed this form following the TAM [17], which suggests using two scales of six 7-point Likert items, ranging from "strongly disagree" (0) to "strongly agree" (6).

Interview questions. We elaborated a set of questions to gain further insight into the subjective perception of MOSKitt4ME users (see [14]). These questions were divided into two parts, which request, respectively, the subjects' opinion about their performance and specific functional aspects of MOSKitt4ME.

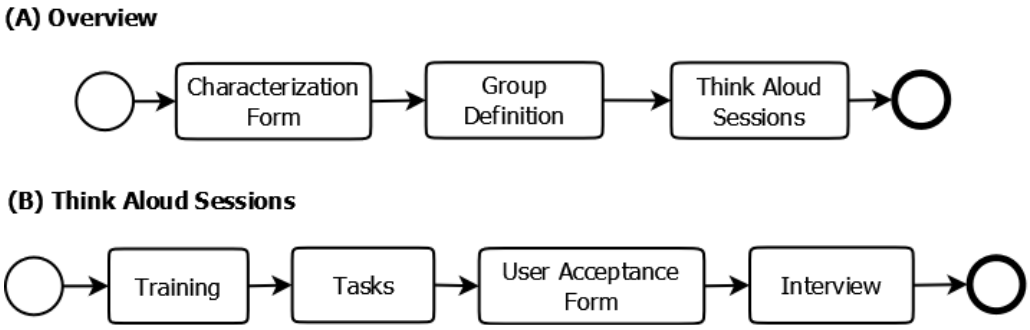

Figure 4: Experimental setup

Physical devices and tools. Following the Think Aloud method [18], we used a webcam to record the subjects' physical behavior and the uttered thoughts. We also used *HyperCam 3.5* to create screencasts that stored the subjects' work. The computer that we provided to the subjects was a *HP Spectre XT Pro Ultrabook 13-b000* with *Windows 7 Professional*, *Intel Core i5 1.7GHz*, and *4GB* of *RAM memory*. In this computer, we installed Eclipse, MOSKitt4ME, and the repositories required for Tasks 1.1 and 2.1.

5.9. Experimental Setup

The process that was followed in the study is shown in Figure 4 (A). First, we gathered demographic data by means of the characterization form. Based on this data, we assigned subjects to two groups of equal size and similar average experience. Then, the study was executed as Think Aloud sessions that were individual; that is, only the experimenter and one subject participated in each session. The groups were used to determine for each subject the treatment to be applied first; thus, we minimized the threat of maturation [38] (see Section 5.10.2).

The process that was followed in each of the Think Aloud sessions is shown in Figure 4 (B). Each session began with a training phase. In this phase, the experimenter assisted the subject in performing the tasks using a small example method; the experimenter also gave instructions on how to think aloud. After the training phase, the subject performed the tasks using the method defined in Section 5.4. During the performance of the tasks, the subject was asked to verbalize their thoughts. When the subject finished a task, he/she had to specify the mental effort invested. Once all of the tasks were finished, the subject filled out the user acceptance form, and, then, the experimenter conducted the interview.

5.10. Validity Evaluation

We considered four types of validity threats: conclusion validity, internal validity, construct validity, and external validity [38].

5.10.1. Conclusion Validity

Our study was affected by three threats to conclusion validity. First, our study was threatened by the reliability of the collected measures. Since we video-recorded the Think Aloud sessions, the collection of objective measures was separated from human judgement, and, hence, they can be considered to be reliable. We increased the reliability of subjective measures by using scales previously validated in other studies [17]. The second threat appears because the Think Aloud sessions took place on different dates, and, thus, their implementation may have differed. To reduce this threat, we replicated the same settings for all of the subjects. Finally, we reduced the random heterogeneity of subjects by evaluating their experience beforehand.

5.10.2. Internal Validity

Our study was affected by three threats to internal validity. The first threat is that different groups may behave differently (e.g., learning at different rates). We minimized this threat by placing subjects in two groups of similar average experience. The second threat is maturation, which implies that subjects may react differently as time passes (e.g., due to tiredness). To minimize this threat, we designed our study so that one group applied Treatment 1 first and the other group applied Treatment 2 first; additionally, we selected a test object that allowed subjects to finish the tasks in less than two hours. Finally, social threats were avoided because the Think Aloud sessions were individual and the subjects were not allowed to talk about the study.

5.10.3. Construct Validity

Our study was affected by two threats to construct validity. First, we reduced hypothesis guessing by hiding the goal of the study and the mechanisms used to collect data; thus, subjects could focus on the task at hand in the most spontaneous way possible. Second, we minimized the effect of the experimenter expectancies by reducing the interaction between the experimenter and the subjects to a minimum.

5.10.4. External Validity

Our study was affected by two threats to external validity. The first threat involves the selection of subjects that are not representative of the population of interest. We minimized this threat by selecting software developers from two industrial software companies. The second threat involves having an inadequate experimental setting. To minimize this threat, we utilized tools that are commonly used in industrial environments (e.g., the Eclipse platform); additionally, the object of the study is part of an industrial method: gvMétrica. Nonetheless, further experimentation is needed to assess how far the results of our study can be generalized to industrial settings and to other types of development methods.

Table 2: Subjects of the study

Id	Gender	Age	Work Status	Degree
S1	Female	41-55	Professional	Engineer
S2	Female	26-40	Academic	Master
S3	Male	26-40	Academic	PhD
S4	Male	26-40	Professional	Master
S5	Female	26-40	Academic	Master
S6	Male	26-40	Professional	Engineer
S7	Male	26-40	Professional	Engineer
S8	Female	18-25	Academic	Engineer

Table 3: Distribution of the subjects

	Group G1				Group G2			
Subjects	S1	S4	S6	S7	S2	S3	S5	S8
Experience	4.33	2.67	2.17	3.25	3.67	3.67	3.42	1.75
Average	3.10				3.12			

6. Execution

This section details the second phase of the experimental process. This phase involves three steps: preparation, operation, and data validation [38].

6.1. Preparation

The preparation for the study involved the selection of the subjects according to stratified random sampling [38]. Our population comprised two groups: one academic and one industrial. The former was composed of master/phd students and postdocs from the DSIC department; all of them had no relationship with MOSKitt4ME but they worked in the area of software engineering. The latter comprised software engineers from two valencian companies. The result of the selection is shown in Table 2. One of the subjects was a master student (S8), two were phd students (S2 and S5), and one was a postdoc (S3); the rest were industrial software engineers. We selected eight subjects since small samples are adequate in Think Aloud studies due to the richness and large amount of data that is produced [35, 40].

With respect to the experience of the subjects, the characterization form revealed that they had low experience in method modeling, medium in development projects and CASE environments, and high in Eclipse and MDE. Based on the subjects' experience (which was measured on a scale from 0 to 6), we evenly distributed them in two groups: G1 and G2. Table 3 shows the resulting distribution.

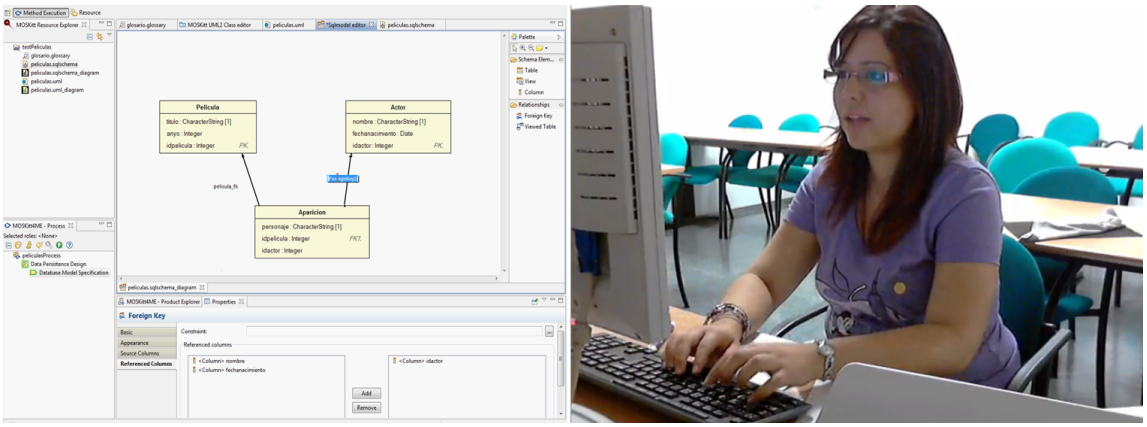

Figure 5: One of the subjects during a Think Aloud session

The preparation phase also involved the elaboration of the required instruments and the execution of a pre-test, where we simulated a Think Aloud session prior to the actual study. The pre-test allowed us to ensure the feasibility of the general setup and to improve the comprehensibility of the textual documents. The person that was selected for the pre-test did not participate in the actual study.

6.2. Operation

We successfully conducted the eight Think Aloud sessions over a two-week period in October 2013. The sessions lasted approximately 2.5 hours on average. To replicate the same settings in all of the sessions, we provided subjects with the same installations of Eclipse and MOSKitt4ME, and these tools were restored to their original state after each session. Additionally, to ensure that the experimental setup was strictly followed, the experimenter always stayed inside the laboratory. Nonetheless, he only talked to break silences after a fixed interval of 30 seconds. If the subjects needed help, they were allowed to consult the MOSKitt4ME user manual.

As an illustration of a Think Aloud session, Figure 5 shows a snapshot of a subject using the CASE environment generated by MOSKitt4ME. As Figure 5 shows, the camera was directed at the subject to give a clear view of the subject's face and hand movements. This facilitated the subsequent interpretation of the verbal data.

6.3. Data Validation

As the Think Aloud method suggests [18], only one subject participated in each session, and, thus, we could ensure that the setup was strictly followed. We are also confident that all of the subjects understood how to fill in the user acceptance form and how to assess mental effort since we explained these tasks in great detail.

7. Data Analysis

In this section, we describe the data analysis phase. This section is divided into two subsections. One deals with subjective data, which allowed us to answer RQ1 and RQ2; the other deals with objective data, which allowed us to answer RQ3 and RQ4. Note that this section describes how we carried out the analysis (e.g., the processes that we followed and the statistical techniques that we applied). The results of the analysis are reported in Section 8.

7.1. Analysis of the Subjective Data

The subjective data corresponds to: (1) the quantitative feedback obtained by means of the user acceptance form, (2) the qualitative feedback obtained during the interviews, and (3) the mental effort that was reported by the subjects.

7.1.1. Quantitative Feedback

We analyzed the responses of the user acceptance form to obtain a quantitative view of the subjects' perceived usefulness and ease of use of MOSKitt4ME. To obtain this view, we considered the numerical values of the responses: from 0 for "Strongly disagree" to 6 for "Strongly agree". Thus, we could calculate the minimum, maximum, and average values for each Likert item of the form (and also the total averages combining all of the items). Additionally, we calculated the frequencies of the responses. The frequency of a response is the sum of occurrences of the response divided by the total number of questions.

7.1.2. Qualitative Feedback

To reinforce the results obtained for perceived usefulness, we analyzed the qualitative feedback collected during the interviews. The first part of the interviews allowed us to determine whether the subjects considered MOSKitt4ME to be useful; that is, whether they believed that MOSKitt4ME improved their performance. The second part allowed us to assess perceived usefulness with respect to specific aspects of the MOSkitt4ME functionality.

7.1.3. Mental Effort

To reinforce the results that were obtained for perceived ease of use, we analyzed the mental effort invested by the subjects. To this end, we performed two Wilcoxon signed-rank tests [38] using *IBM SPSS Statistics 2.0*. The Wilcoxon test is an appropriate technique for our study since we have paired samples; furthermore, the Wilcoxon test (in contrast to the paired t-test) does not require the data to be normally distributed, a requirement that was not met in our study. The normality tests that we performed can be found in [14].

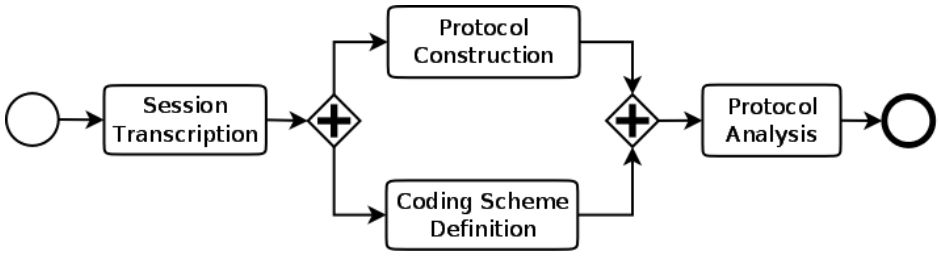

Figure 6: Data analysis process (adapted from [18])

Specifically, the Wilcoxon tests allowed us to verify if there was a significant difference in mental effort between Treatment 1 and Treatment 2. This difference should be in line with the results obtained for perceived ease of use; note that, e.g., a subject who invests little mental effort using MOSKitt4ME should consider the tool as easy to use.

The first Wilcoxon test focused on the method design/implementation, while the second test focused on the method execution. We considered the Wilcoxon tests to be two-tailed and they were performed at a confidence level of 95% ($\alpha = 0.05$). The null hypothesis (H_0) was the same for both tests: the median of differences in mental effort is equal to zero. H_0 can be rejected if $p < \alpha$, where p is the p-value obtained from the Wilcoxon tests.

7.2. Analysis of the Objective Data

The objective data is obtained by analyzing the subjects' behavior, which is stored in video records and screencasts. The process that we followed to obtain the objective data is outlined in Figure 6. First, we transcribed the Think Aloud sessions; that is, we typed out video records and screencasts as verbatim as possible. Then, we annotated the transcriptions using a coding scheme to obtain Think Aloud protocols. The coding scheme, which was developed in parallel to the protocols, contains codes that define different types of utterances and actions. Thus, the protocols are transcriptions whose utterances and actions are classified as, e.g., doubts or errors. When the transcriptions were fully annotated, we analyzed the resulting protocols. The four tasks of the process are detailed in the following subsections.

7.2.1. Session Transcription

Since it is hard to analyze audio records and screencasts, we transcribed them into text, and, then, we divided this text into segments. The segments of a transcription represent utterances (which are obtained from the video records) and actions (which are obtained from the screencasts). We produced a total of 8 transcriptions, which have 895 segments on average. Each of the segments of these transcriptions stores three items: time, type, and text. *Time* indicates the exact moment of occurrence of

the segment within the Think Aloud session. *Type* determines whether the segment is an utterance or an action. Finally, *text* represents the segment content. The content of an utterance is the textual representation of the subject's verbalization; the content of an action is a short description of the action.

7.2.2. Coding Scheme Definition

Because it is unreliable to analyze the transcriptions "as is", it is necessary to make a coding scheme that enables the classification of their segments. To develop the coding scheme, we analyzed the transcriptions, and, concurrently, we created new codes for each segment that did not fall neatly into the existing coding scheme. Then, we categorized the resulting codes. The result was 90 codes in 7 different categories (see [14]). The categories of the coding scheme are the following:

- *Actions (A).* General actions, such as deleting a file.
- *Tasks (T).* Actions that correspond to method tasks (e.g., revising the DDL script).
- *Errors (E).* Actions that do not adhere to any valid solution for the task at hand (e.g., setting a name incorrectly).
- *Comments (C).* General utterances such as opinions or doubts.
- *Strategies (S).* Utterances or actions whereby subjects express or adopt a plan to achieve a goal (e.g., postponing an analysis).
- *Expert Knowledge (EK).* Utterances or actions whereby subjects: suggest they require further knowledge, show they have previous knowledge, or gain new knowledge during the performance of a task (e.g,. the utterance "I do not need to check this because I am familiar with the tool" reflects previous knowledge).
- *Challenges (CH).* Utterances or actions that suggest the presence of a challenge or difficulty. Indicative of challenges can be utterances or actions from the E, C, S, and EK categories (e.g., if a subject applies the "postponing an analysis" strategy, the subject is probably facing difficulties that he/she decides to work out later).

7.2.3. Protocol Construction

The Think Aloud protocols are transcriptions that have been annotated with codes from the coding scheme. To increase the objectivity of our coding process, we selected two researchers that were external to the study and we trained them in the use of the coding scheme. These researchers revised the protocols that we coded; all of the discrepancies were discussed and fixed when agreements were reached.

Table 4 shows an excerpt of a protocol. Each row represents a different segment. In this example, the subject starts by consulting the method description (A10).

Table 4: Excerpt of a Think Aloud protocol

Time	Type	Text	Code
1:31:50	Action	Looks at method description	A10
1:31:50	Utterance	And now, business logic design	C6
1:31:53	Utterance	It is a UML class diagram	C3
1:31:54	Action	Looks for UML 2.0 editor	A29
1:31:57	Utterance	UML model	C7
1:31:58	Action	Selects the "UML Model" tool	A30, E1, CH2
1:31:59	Utterance	I assume that it is UML model	S1, EK3

Then, the subject verbalizes the information retrieved (C6) and resolves that he must create a UML class diagram (C3). For this reason, he looks for the UML 2.0 editor (A29). He finds a tool called "UML Model", reads it (C7), and selects the tool (A30). Since it is not the correct choice (E1), we consider selecting the correct tools to be a challenge of the method execution (CH2). Finally, the subject verbalizes that he assumes "UML Model" is the correct tool. He is adopting a "trial and error" strategy (S1), which indicates that he requires further technical knowledge (EK3).

7.2.4. Protocol Analysis

Once the protocols were obtained, we analyzed them to measure task completion time, task completeness, and task difficulty (see Section 4.1).

- *Task Completion Time.* We used the *Time* column to calculate the time spent by subjects on the tasks of the study. We applied Wilcoxon tests to analyze the differences between the tasks of Treatment 1 and Treatment 2. Thus, we could determine whether MOSKitt4ME allowed subjects to solve the tasks more quickly.
- *Task Completeness.* To calculate task completeness, we used the *Type*, *Text*, and *Code* columns of the protocols. As for the method design/implementation, we analyzed the subjects' actions and errors (i.e., the segments of type "Action" and code categories "A" and "E", respectively); this allowed us to determine whether the subjects built a CASE environment that provided complete support to the method. To determine the completeness of the method execution, we analyzed the subjects' errors and also the actions associated to method tasks (i.e., the segments of type "Action" and code category "T"); thus, we could determine the number of method tasks successfully performed by the subjects.
- *Task Difficulty.* To estimate the difficulty of the tasks of the study, we analyzed the segments falling into the "CH" category of the coding scheme. These segments allowed us to determine whether using MOSKitt4ME was a big effort for the subjects or rather it posed little difficulty.

Table 5: Results for perceived usefulness

Item	Min	Max	Avg
1 - Allows working more quickly	5	6	5.625
2 - Improves job performance	4	6	5.5
3 - Increases productivity	4	6	5.375
4 - Enhances effectiveness	4	6	5.375
5 - Makes work easier	5	6	5.375
6 - Is useful for the job	4	6	5.25
Total Average			5.42

8. Results

In this section, we present the results of the study by answering the four research questions that are formulated in Section 5.2.

8.1. RQ1. What is the users' perceived usefulness of MOSKitt4ME?

After the analysis of the responses of the user acceptance form, we obtained the results that are shown in Table 5. The minimum (*Min*) and maximum (*Max*) columns indicate that all of the subjects somewhat agreed (4), agreed (5), or strongly agreed (6) about each of the items of the usefulness scale. We obtained the best result for the first item (average: 5.625); that is, the subjects expressed a strong belief that tasks can be performed more quickly with MOSKitt4ME. The subjects also positively rated the improvement in performance (average: 5.5). In general, the subjects agreed that MOSKitt4ME was useful for solving the tasks of the study (total average: 5.42).

To provide further insight into perceived usefulness, Figure 7 shows a histogram that depicts the distribution of responses of the subjects. The horizontal axis contains the seven possible responses; the vertical axis represents their frequency. The histogram shows that the most common responses were "Strongly agree" and "Agree"; on average, more than 4 subjects selected "Strongly agree" in each of the Likert items, and nearly 3 subjects selected "Agree".

These results were reinforced by the qualitative feedback that was obtained during the interviews. All of the subjects expressed that MOSKitt4ME was useful since it allowed them to perform the tasks more easily. Most subjects emphasized the method execution; for instance, one subject stated: "*Executing the method with MOSKitt4ME, you click on the tasks and the tools are automatically opened. Without MOSKitt4ME, it is hard to find the right tools in the large set of tools that are offered by Eclipse*". The usefulness of the CASE generation capabilities was also emphasized by some subjects: "*I would not invest the time needed to implement a*

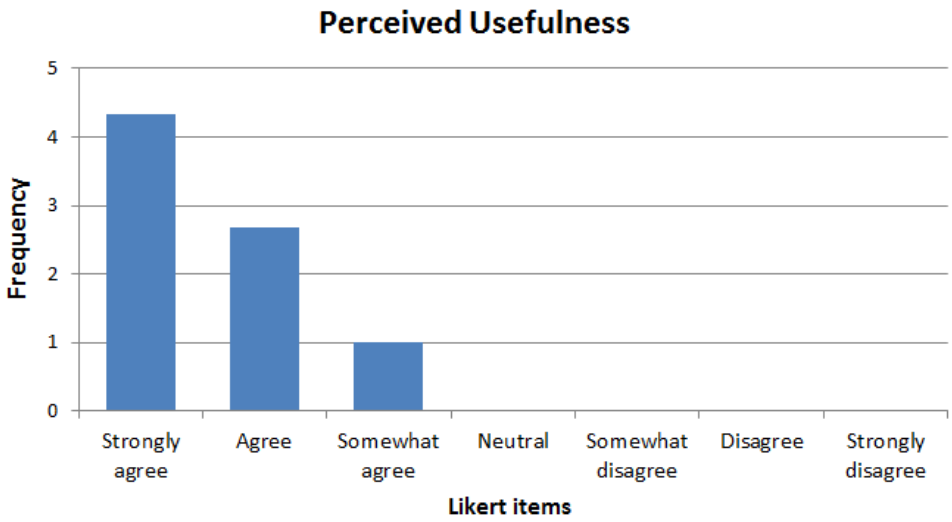

Figure 7: Frequencies of responses for perceived usefulness

Table 6: Results for perceived ease of use

Item	Min	Max	Avg
1 - Easy to learn	3	6	4.5
2 - Controllable	1	5	2.625
3 - Clear and understandable	4	6	4.75
4 - Flexible	4	6	4.625
5 - Easy to become skillful	3	6	4.75
6 - Easy to use	4	6	4.875
Total Average			4.35

CASE environment. Using MOSKitt4ME, I would consider the possibility". Four subjects also highlighted that MOSKitt4ME facilitates the method design by enabling the definition of methods as models at a high level of abstraction: "*It is so much more user-friendly and intuitive to edit the method using MOSKitt4ME, compared to the textual descriptions that we use in our company.*". Finally, we also found comments that, despite being more general, illustrate the subjects' willingness to use MOSKitt4ME: "*I hate to do work that can be avoided. Knowing the functionality of MOSKitt4ME is possible, I would not want to work differently from now on*".

8.2. RQ2. What is the users' perceived ease of use of MOSKitt4ME?

The results for perceived ease of use are shown in Table 6. Compared with perceived usefulness, the results were also positive, but we found more dispersion in

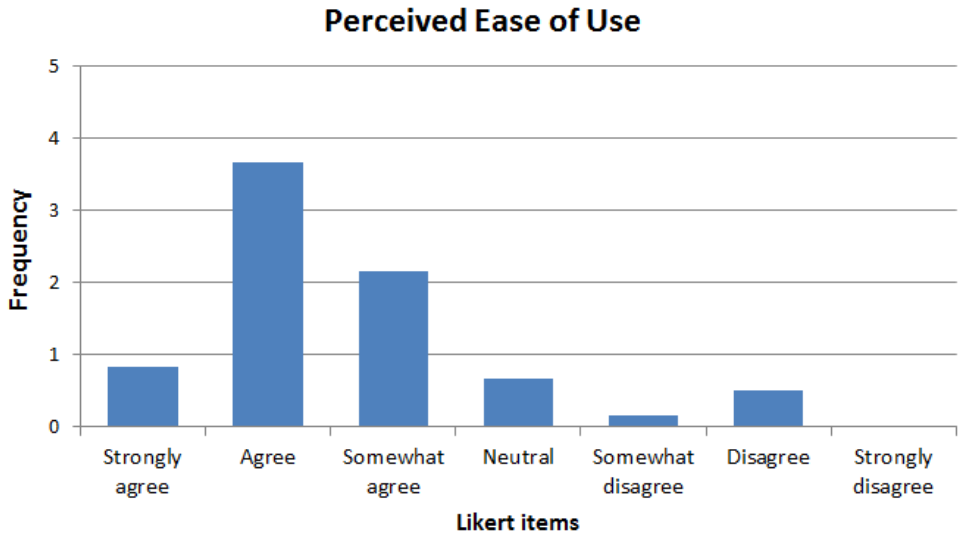

Figure 8: Frequencies of responses for perceived ease of use

them. The minimum (*Min*) and maximum (*Max*) columns indicate that all of the subjects considered (4, 5, or 6) MOSKitt4ME to be clear, understandable, flexible, and easy to use; but this did not occur for the other items. We obtained the worst result for the "Controllable" item (average: 2.625). This means that it was not easy for some subjects to get MOSKitt4ME to do what they wanted it to do. This result was due to the low level of experience using MOSKitt4ME: we observed that most subjects frequently consulted the user manual. In general, the subjects somewhat agreed that MOSKitt4ME can be used with little difficulty (total average: 4.35).

To provide further insight into perceived ease of use, Figure 8 shows a histogram that depicts the distribution of responses of the subjects. The most common were "Agree" and "Somewhat agree"; on average, nearly 4 subjects selected "Agree" in each of the Likert items, and more than 2 subjects selected "Somewhat agree".

These results were reinforced by the mental effort that was expressed by the subjects. The subjects' mental effort is depicted as a box plot in Figure 9. The horizontal axis contains the four tasks of the study, while the vertical axis represents mental effort; thus, each box represents, for one specific task, the mental efforts invested by the eight subjects of the study. The distribution of the data indicates that the subjects expended less effort executing the method with MOSKitt4ME than executing the method without the aid of our tool. The effort invested in the method design/implementation was low but similar in both approaches. This similarity was due to the low experience of the subjects in method modeling and the high experience in Eclipse. Note that when the subjects were not using MOSKitt4ME, they had to manually configure an Eclipse-based CASE environment, which was easy for some

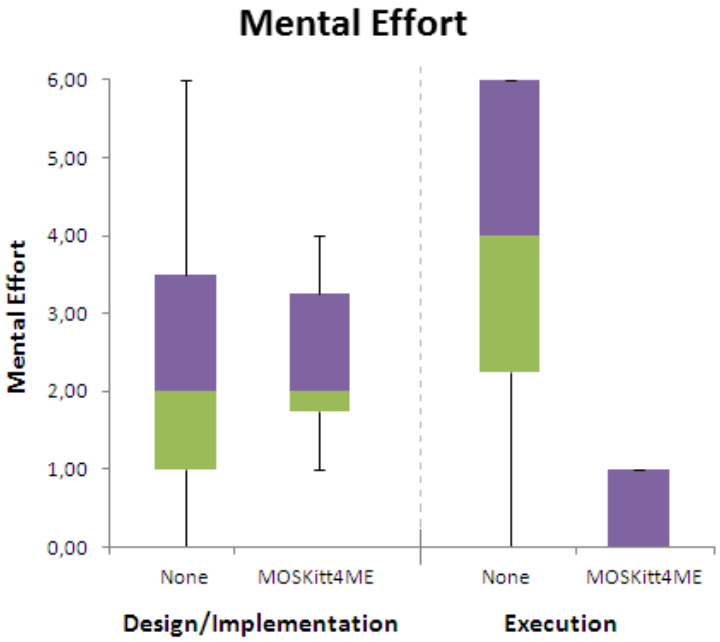

Figure 9: Mental effort (box plot)

subjects; in contrast, when they used MOSKitt4ME, the CASE tool was obtained automatically but it required the construction of a method model.

To verify whether the differences in mental effort were statistically significant, we performed two Wilcoxon signed-rank tests. In the first test, which focused on the method design/implementation, we obtained $p = 0.931$. Since $\alpha = 0.05$, then $p > \alpha$ and, therefore, we cannot reject H_0. Thus, there is no significant difference in mental effort in the method design/implementation. In the second Wilcoxon test, which focused on the method execution, we obtained $p = 0.027$. In this case, we can reject H_0 since $p < \alpha$. Thus, subjects expended significantly less mental effort when they executed the method with the aid of MOSKitt4ME.

8.3. RQ3. To what extent does MOSKitt4ME enhance efficiency and effectiveness?

To answer this question, we quantified the subjects' efficiency and effectiveness, and contrasted the results obtained in the two treatments of the study.

8.3.1. Efficiency

Figure 10 shows in a boxplot the subjects' efficiency. The vertical axis represents time; the horizontal axis contains the four tasks of the study. Thus, each box represents, for one specific task, the completion times of the eight subjects of the study. Analyzing the data distribution, we observed that subjects were more efficient in the

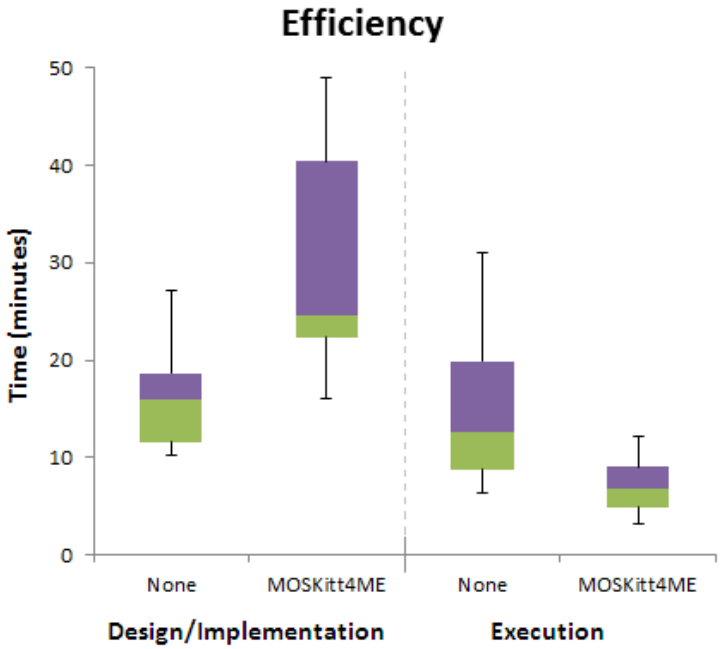

Figure 10: Efficiency (box plot)

method design/implementation without MOSKitt4ME than using our tool. The goal of this task was to obtain a CASE environment; thus, MOSKitt4ME failed to reduce the time needed to build this tool. The cause of this result was the high amount of time that some subjects (mainly those with low experience in method modeling) invested building the model of the method (which is mandatory in MOSKitt4ME since the CASE environment is obtained automatically from this model).

Despite this negative result, it is important to consider that having the method represented as a model brings important benefits that are not reaped in the manual approach. Some of the benefits that were reported by the subjects are the following. First, the model enables the generation of documentation of the method in different formats, such as HTML or plain text. Second, the method becomes easier to maintain and easier to navigate. Third, the method model facilitates the communication between the people involved in a project. Fourth, the method model enables a high level of automation thanks to the use of model transformations. Finally, the CASE environment can execute the method model at runtime to assist software engineers during the course of the projects.

The last benefit became apparent in our study. The subjects invested significantly more time during the method execution without MOSKitt4ME than executing the method with the assistance of our tool. This positive result is of particular relevance.

Note that, even though the development of the method model is costly in terms of time, this time is only invested once. In contrast, time savings during method execution occur any time a development project is performed. Thus, it seems fair to conclude that MOSKitt4ME improves efficiency if it is applied in multiple projects.

To verify whether the differences in efficiency were significant, we performed two Wilcoxon signed-rank tests. The first test analyzes the time invested during the method design/implementation; the second test focuses on method execution. In both tests, we obtained $p = 0.012$. Thus, the condition $p < \alpha$ was fulfilled, and, therefore, the differences in efficiency were statistically significant.

8.3.2. Effectiveness

After protocol analysis, we found that all of the subjects obtained the correct outcome in the method design/implementation (effectiveness: 100%). This outcome was obtained in both treatments; nonetheless, the manual approach (i.e., Treatment 1) caused severe problems when subjects tried to integrate tools into Eclipse. Since this Eclipse only contained a minimum set of plug-ins, installing new tools "by hand" raised problems of missing software dependencies. Solving these problems was considered by all of the subjects to be complex, tedious, and error-prone. None of these problems occurred with MOSKitt4ME since our tool automates the construction of the CASE environment.

In contrast to the method design/implementation, we found differences in effectiveness in the method execution. All of the subjects executed the entire method when they were assisted by MOSKitt4ME (effectiveness: 100%), while three subjects abandoned the execution in the manual approach. These subjects failed to perform T4 because they performed T3 incorrectly: they selected the wrong tool, and, therefore, they created incorrectly the input product of T4.

In addition to selecting incorrect tools and creating incorrect products, we also found other deviations from the method; for instance, one subject performed T2 and T3 in reverse order and omitted the execution of T5. None of these deviations occurred when the subjects were assisted by MOSKitt4ME.

8.4. RQ4. To what extent can MOSKitt4ME be used free from difficulty?

By analyzing the protocols, we found that the subjects only experienced difficulty regarding two aspects of MOSKitt4ME: SPEM 2.0 and the reusable assets.

8.4.1. Difficulty Using SPEM 2.0

Several subjects experienced difficulty understanding the SPEM 2.0 concepts. Specifically, two subjects had problems defining the output products of the tasks: it was not easy for these subjects to distinguish the different kinds of products that are proposed by SPEM 2.0. Additionally, six subjects experienced doubt during the

definition of the process: they were uncertain whether or not to make explicit the precedences between the tasks contained in "Data Persistence Design" and the tasks outside this activity. Finally, three subjects had problems distinguishing method content from method process. One of them spent three minutes trying to define an activity within a content package. Another subject took two minutes to realize that process tasks had to be defined by instantiation from content tasks. The third subject did not realize that the products defined as method content could be instantiated several times; he created the product "DDL Script" twice because it is the output of two different tasks. This was the only error that was made by the subjects as a result of all these minor difficulties.

8.4.2. Difficulty Defining Technical Data

Some subjects had difficulty associating method elements and reusable assets, and, in general, understanding the notion of reusable asset. Specifically, one subject stated that he did not understand why (unlike tasks, roles, and products) the method tools had to be defined using a repository. Another subject spent one minute trying to determine which reusable asset to associate to T5, even though this task must not have any tool associated to it. Another doubt was whether T3 was automatic; note that this task is not automatic because it is supported by a graphical editor. Some subjects also had problems understanding the semantics of some types of reusable assets. All of these doubts caused that three subjects made incorrect associations between method elements and reusable assets; therefore, this is an important aspect of MOSKitt4ME to be improved in the near future.

8.5. Discussion

The evaluation study of MOSKitt4ME provides valuable insight into the usefulness and ease of use of our model-driven ME approach. Below, we highlight the most relevant aspects of the results that are presented herein.

- The subjective perception that was expressed by the subjects of the study indicates their willingness to accept and use MOSKitt4ME. They perceived MOSKitt4ME to be a useful tool that can improve performance without posing severe difficulties for the users (see RQ1 and RQ2).
- The assistance that is provided by MOSKitt4ME allowed subjects to perform the method execution without deviations, and this led to a significant increase in efficiency and effectiveness (see RQ3). This result suggests that MOSKitt4ME facilitates the use of methods, and, thus, it reduces the distance that typically exists between methods and the real actions performed by software engineers [41]. Nonetheless, this benefit is reaped at the expense of the time that is required to build the method model, which may be high in cases of low modeling experience.

In order to reduce this time, we plan to enhance MOSKitt4ME in two ways. First, we will increase the starting population of the repository so that it also contains reusable method parts (e.g., tasks extracted from gvMétrica). This will reduce time by enabling rapid method assembly. Second, we will increase the level of automation of MOSKitt4ME by incorporating variability mechanisms that automate the adaptation of methods and CASE environments.

- The method design is the only phase of the ME lifecycle where MOSKitt4ME users experienced difficulty during the study (see RQ4). Most of these difficulties were of low severity and were related to the use of SPEM 2.0 and the reusable assets; therefore, these difficulties can be mitigated by enhancing MOSKitt4ME with appropriate assistance for method construction. To do this, we plan to include a wizard that will free users from having to be expert method engineers, allowing them to create method models following a set of intuitive steps. We expect this wizard to also reduce the time invested by the users in the method design phase.

In addition to all of the above findings, our study also had a collateral benefit. Even though it was not the focus of the evaluation, our results suggest that MDE plays a key role in the reduction of ME complexity that is achieved by MOSKitt4ME. Four subjects strongly believed that (meta)modeling techniques reduce the complexity of the method design phase by enabling the definition of methods as models at a high level of abstraction (see RQ1). This result is in line with one of the most recognized benefits of MDE: the reduction of the complexity of software development by providing higher levels of abstraction that hide platform-specific details [42]. On the other hand, the eight subjects of the study were strongly satisfied with the level of automation of MOSKitt4ME (see RQ1 and RQ3). This level is achieved thanks to model transformations, which reduce the complexity of the method implementation phase by automating the CASE environment construction. This is in line with another benefit of MDE: the reduction of complexity by means of the automation of labor-intensive and error-prone tasks [42]. Finally, all of the subjects experienced a significant improvement in efficiency and effectiveness during the method execution phase (see RQ3). This result is a strong indicator of the reduction of complexity that is achieved by using the method model at runtime. Thus, the modeling effort made at design time is not only useful for automating the CASE environment construction, but it can also assist software engineers during the process of software development.

9. Conclusions and Future Work

This paper presents a study that evaluates the usefulness and ease of use of a model-driven ME approach: MOSKitt4ME. Our motivation is to demonstrate that

MOSKitt4ME mitigates an important problem of traditional ME: its high complexity. To achieve this goal, the study evaluates MOSKitt4ME using the TAM and the Think Aloud method. While the TAM enables the evaluation of the subjective perception of users, the Think Aloud method allows us to evaluate the tool objectively.

The results of the study are encouraging. All of the subjects either somewhat agreed, agreed, or strongly agreed about each of the items of the usefulness scale (see RQ1); we also obtained positive results for perceived ease of use, even though we found more disparate opinions (see RQ2). These subjective results were reinforced by an increase in efficiency and effectiveness (see RQ3) as well as by the little difficulty that was experienced by the subjects of the study (see RQ4). We believe that these results were obtained thanks to the use of MDE techniques (such as metamodeling, model transformations, and models at runtime), which reduce the complexity of three phases of the ME lifecycle: design, implementation, and execution.

In contrast to these positive findings, we also found several challenges that are inherent to MOSKitt4ME usage (see RQ4). With the aim of providing better tool support for model-driven ME, we will address these challenges in the near future. As Section 8.5 describes, to reduce the time needed for method design, we will enhance the repository of MOSKitt4ME, incorporate support for variability, and include a wizard that enables guided model creation. Even though it is not related to the study, another enhancement of MOSKitt4ME will be to make it into a more collaborative environment; to this end, we will persist models in centralized online databases so that users can work concurrently during the ME lifecycle.

References

[1] A. Cockburn, Selecting a project's methodology, IEEE software 17 (4) (2000) 64–71.

[2] B. Henderson-Sellers, J. Ralyté, Situational method engineering: State-of-the-art review, J. UCS. 16 (2010) 424–478.

[3] F. Karlsson, P. J. Ågerfalk, Method configuration: adapting to situational characteristics while creating reusable assets, Information and Software Technology 46 (9) (2004) 619–633.

[4] S. Brinkkemper, Method engineering: Engineering of information systems development methods and tools, Information and Software Technology 38 (4) (1996) 275–280.

[5] S. Brinkkemper, M. Saeki, F. Harmsen, Meta-modelling based assembly techniques for situational method engineering, Information Systems 24 (3) (1999) 209–228.

[6] J. Ralyté, C. Rolland, An assembly process model for method engineering, in: K. Dittrich, A. Geppert, M. Norrie (Eds.), Advanced Information Systems Engineering, Vol. 2068 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2001, pp. 267–283.

[7] B. Henderson-Sellers, M. Serour, Creating a dual-agility method: The value of method engineering, Journal of Database Management 16 (4) (2005) 1–24.

[8] A. Niknafs, R. Ramsin, Computer-aided method engineering: an analysis of existing environments, in: Advanced Information Systems Engineering, Springer, 2008, pp. 525–540.
[9] M. Bajec, D. Vavpotič, M. Krisper, Practice-driven approach for creating project-specific software development methods, Information and Software Technology 49 (4) (2007) 345–365.
[10] C. Rolland, Method engineering: towards methods as services, Software Process: Improvement and Practice 14 (3) (2009) 143–164.
[11] M. Kuhrmann, D. Méndez Fernández, M. Tiessler, A mapping study on the feasibility of method engineering, Journal of Software: Evolution and Process (2014) 1–22.
[12] B. Henderson-Sellers, Method engineering for OO systems development, Communications of the ACM 46 (10) (2003) 73–78.
[13] A. H. Ter Hofstede, T. Verhoef, On the feasibility of situational method engineering, Information Systems 22 (6) (1997) 401–422.
[14] MOSKitt4ME, users.dsic.upv.es/∼mcervera/moskitt4me/.
[15] M. Cervera, M. Albert, V. Torres, V. Pelechano, The MOSKitt4ME approach: providing process support in a method engineering context, in: Conceptual Modeling, Vol. 7532, Springer, 2012, pp. 228–241.
[16] M. Cervera, M. Albert, V. Torres, V. Pelechano, A model-driven approach for the design and implementation of software development methods, International Journal of Information System Modeling and Design (IJISMD) 3 (4) (2012) 86–103.
[17] F. D. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS quarterly 13 (3) (1989) 319–340.
[18] M. W. van Someren, Y. F. Barnard, J. A. C. Sandberg, The Think Aloud Method: A Practical Guide to Modelling Cognitive Processes, Academic Press Limited, 1994.
[19] C. K. Riemenschneider, B. C. Hardgrave, F. D. Davis, Explaining software developer acceptance of methodologies: a comparison of five theoretical models, Software Engineering, IEEE Transactions on 28 (12) (2002) 1135–1145.
[20] V. Venkatesh, F. D. Davis, A model of the antecedents of perceived ease of use: Development and test, Decision sciences 27 (3) (1996) 451–481.
[21] F. D. Davis, User acceptance of information technology: system characteristics, user perceptions and behavioral impacts, International Journal of Man-Machine Studies 38 (3) (1993) 475–487.
[22] J.-P. Tolvanen, M. Rossi, H. Liu, Method engineering: current research directions and implications for future research, in: Method Engineering, Springer, 1996, pp. 296–317.
[23] K. Sousa, J. Vanderdonckt, B. Henderson-Sellers, C. Gonzalez-Perez, Evaluating a graphical notation for modelling software development methodologies, Journal of Visual Languages & Computing 23 (4) (2012) 195–212.
[24] ISO, Software Engineering: Metamodel for Development Methodologies. ISO/IEC 24744 (2007).
[25] S. Kelly, M. Rossi, Evaluating method engineer performance: an error classification and preliminary empirical study, Australasian Journal of Information Systems 6 (1)

(1998).
[26] N. Kerzazi, M. Lavallee, Inquiry on usability of two software process modeling systems using iso/iec 9241, in: 24th Canadian Conference on Electrical and Computer Engineering (CCECE), 2011, pp. 000773–000776.
[27] F. Karlsson, K. Wistrand, Combining method engineering with activity theory: theoretical grounding of the method component concept, European Journal of Information Systems 15 (1) (2006) 82–90.
[28] A. Qumer, B. Henderson-Sellers, An evaluation of the degree of agility in six agile methods and its applicability for method engineering, Information and Software Technology 50 (4) (2008) 280–295.
[29] F. Karlsson, A wiki-based approach to method tailoring, in: Proceedings of the 3rd International Conference on the Pragmatic Web: Innovating the Interactive Society, ACM, 2008, pp. 13–22.
[30] V. Seidita, M. Cossentino, S. Gaglio, Adapting PASSI to support a goal oriented approach: a situational method engineering experiment, in: Proc. of the Fifth European workshop on Multi-Agent Systems (EUMAS'07), 2007.
[31] OMG, Software & Systems Process Engineering Metamodel (v2.0) (2007).
[32] Graphical Modeling Project, http://www.eclipse.org/modeling/gmp/.
[33] K. Hornbæk, Current practice in measuring usability: Challenges to usability studies and research, International journal of human-computer studies 64 (2) (2006) 79–102.
[34] Y. Lee, K. A. Kozar, K. R. Larsen, The technology acceptance model: past, present, and future, Communications of the Association for Information Systems 12 (1) (2003) 50.
[35] R. Benbunan-Fich, Using protocol analysis to evaluate the usability of a commercial web site, Information & Management 39 (2) (2001) 151–163.
[36] P. Runeson, M. Host, Guidelines for conducting and reporting case study research in software engineering, Empirical software engineering 14 (2) (2009) 131–164.
[37] R. D. Henderson, M. C. Smith, J. Podd, H. Varela-Alvarez, A comparison of the four prominent user-based methods for evaluating the usability of computer software, Ergonomics 38 (10) (1995) 2030–2044.
[38] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering: An Introduction, Kluwer Academic Publishers, Norwell, MA, USA, 2000.
[39] gvMétrica, www.gvpontis.gva.es/cast/proyectos-integra/.
[40] S. Owen, P. Brereton, D. Budgen, Protocol analysis: a neglected practice, Communications of the ACM 49 (2) (2006) 117–122.
[41] B. Fitzgerald, The use of systems development methodologies in practice: a field study, Information Systems Journal 7 (3) (1997) 201–212.
[42] P. Mohagheghi, W. Gilani, A. Stefanescu, M. A. Fernandez, An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases, Empirical Software Engineering 18 (1) (2013) 89–116.