On the Effectiveness, Efficiency and Perceived Utility of Architecture Evaluation Methods: A Replication Study

Javier González-Huerta, Emilio Insfrán, Silvia Abrahão

ISSI Research Group, Universitat Politècnica de València
Camino de Vera, s/n, 46022, Valencia, Spain
{jagonzalez, einsfran, sabrahao}@dsic.upv.es

Abstract. In this paper we describe the results of a replication study for comparing the effectiveness, efficiency and perceived utility of the quality-driven product architecture derivation and improvement method (QuaDAI), an architecture derivation and evaluation method that we presented in recent works, as opposed to the Architecture Tradeoff Analysis Method (ATAM), a well-known architectural evaluation method used in industry. The results of the original experiment (conducted with undergraduate students) showed that QuaDAI was found to be more efficient and was perceived as easier to use than ATAM. However, although QuaDAI performed better than ATAM, we could not confirm the other variables, as the differences between both methods were not statistically significant. Therefore the goal of the replication was to verify these findings with a group of more experienced students. In the replication study QuaDAI also performed better than ATAM, but as opposed to the original study, all the variables proved to be statistically significant.

Keywords: Controlled Experiment, Experiment Replication, Software Architecture, Architecture Evaluation Methods, Quality Attributes, ATAM

1 Introduction

Software architecture is a key asset for organizations that build complex software systems. Software architecture is also a means to achieve the non-functional requirements[1] (NFRs) that have to be fulfilled. The Software Product Line (SPL) development paradigm is an approach that takes advantage of the massive reuse of software assets as a means to improve productivity and product quality. SPL is defined as a set of software-intensive systems that share a common, managed set of features developed from a common set of core assets in a prescribed manner [9]. In SPL development, the product line architecture should contain variation mechanisms that help to achieve a set of explicitly permitted variations [9]. These variations may include structural, behavioral and of course quality concerns. The product line architecture should therefore be designed to cover the whole set of variations within the product

[1]Non-Functional Requirements can be defined as the qualities that a product must have, such as an appearance, or a property of speed or accuracy [24].

line. The product architecture can thus be derived from the product line architecture by exercising its built-in architectural variation mechanisms, which support functional and non-functional requirements for a specific product.

Once it has been derived, the product architecture should be evaluated in order to guarantee that it meets the specific requirements of the product under development [9]. However, when the required levels of quality attributes for a specific product fall outside the original specification of the SPL (and cannot be attained by using product line variation mechanisms), certain architectural transformations should be applied to the product architecture to ensure that these NFRs are met [7].

We have addressed the solution to this problem in recent works [17][18][15][16], in which we have presented the Quality-Driven Product Architecture Derivation and Improvement (QuaDAI) method to guide the software architect in the derivation and improvement of product architectures in a model-driven software product line development process.

In this paper we report the results of a replication of a first experiment presented in [17], whose intention was to compare the effectiveness, efficiency, and perceived utility as regards participants using QuaDAI as opposed to the Architecture Tradeoff Analysis Method (ATAM) [20]. Perceived utility was measured by means of the perceived ease of use, intention to use and perceived usefulness. The context of the original experiment was a group of fifth-year Computer Science undergraduates at the Universitat Politècnica de València. The results of the original experiment showed that that QuaDAI was found to be more efficient and was perceived as easier to use than ATAM. However, although QuaDAI was also more effective and perceived as being more useful and more likely to be used by the subjects than ATAM, the results for these variables were found not to be statistically significant.

The remainder of the paper is structured as follows. Section 2 discusses related works in the field. Section 3 presents the architecture evaluation methods being compared. Section 4 presents the original experiment, its design, the variables, the instrumentation, the execution and the results. Section 5 presents the replication study consisting of the motivation, the level of interaction with the original study, the changes to the original experiment and a comparison of the results. Finally, our conclusions and future work are presented in Section 6.

2 Related Work

Although there is a growing need to systematically gather empirical evidence about the advantages of tools and methods in the software architecture field [1], there are few works that report results from empirical studies comparing software architectural evaluation methods [2], [25], [21] or some aspects related to them [3], [4], [14]. Among them, Ali Babar et al. 2004 [2] and Roy and Graham [25] have presented two different classifications of architectural evaluation methods based on different criteria. Martens et al. [21] reported a series of experiments comparing the accuracy and effort of software architecture performance evaluation methods. The aim of the study was to

establish whether it is better to use monolithic or component-based methods for conducting the performance evaluation of software architectures.

Other studies are focused on specific aspects of software architecture evaluation processes [3], [4], [14]. Ali Babar et al. 2008 [3] reported the results of an experiment comparing distributed and face-to-face meetings within the software architecture evaluation process. The goal of the study was to analyze the effectiveness of both types of meetings based on the quality of the scenario profiles developed in each case. Ali Babar et al. 2007 [4] reported the results of an experiment assessing the use of LiveNet, a groupware tool for supporting the software evaluation process. The objective of the study was to analyze the perceived ease of use and usefulness of the tool after performing various collaborative tasks. Falessi et al. [14] reported the results of a replicated experiment to analyze the perceived utility of the information associated with Architectural Design Decisions Rationale Documentation (DDRD). The subjects were requested to perform different activities (described using DDRD Use Cases) and to then rank the categories of DDRD with a 3-point ordinal scale.

In summary, there is a lack of empirical evidence in the software architecture field to support the methods and tools proposed. In general the empirical evidence refers to specific aspects of the evaluation processes, and it is difficult to find experiments that compare general-purpose evaluation methods. This type of experiments would help researchers and practitioners when selecting the architectural evaluation method that best fits the characteristics of their project.

3 Compared Software Architecture Evaluation Methods

In the controlled experiments being reported two architectural evaluation methods had been compared: our proposal QuaDAI and ATAM. ATAM has been selected for comparison with QuaDAI since i) it is a widely used software architecture evaluation method [22], ii) it is able to deal with multi-attribute analysis [2], and iii) it can be used to evaluate both product line and product architectures at various stages of SPL development (conceptual, before code, during development, or after deployment) [9].

3.1 The QuaDAI Method

QuaDAI is a method for the derivation and improvement of the product architecture that defines an artifact (the multimodel) and a process consisting of a set of activities conducted by model transformations. QuaDAI has been designed taking into account the weak points of existing architecture evaluation methods, in order to improve their usability and effectiveness. QuaDAI allows gathering the experts' architectural knowledge, which is reused in architectural evaluations by less-skilled evaluators. QuaDAI relies on a multimodel [15] that permits the explicit representation of relationships among entities in different viewpoints. A multimodel is a set of interrelated models that represents the different viewpoints of a particular system. A viewpoint is an abstraction that yields the specification of the whole system restricted to a particular set of concerns, and it is created with a specific purpose in mind. In any given viewpoint it is possible to produce a model of the system that contains only the objects that are visible from that viewpoint [5]. Such model is known as viewpoint mod-

el, or view of the system from that viewpoint. The multimodel permits the definition of relationships among model elements in those viewpoints, capturing the missing information that the separation of concerns could lead to. The multimodel plays two different roles in SPL development: i) in the *domain engineering phase,* during which the core asset base is created, the multimodel explicitly represents the relationships among the different views; ii) in the *application engineering phase,* during which the final product is derived, the relationships drive the different model transformation processes that constitute the production plan used to produce the final product.

The multimodel used to describe SPLs is composed of (at least) four interrelated viewpoints: *functional*, *variability*, *quality*, and *transformation*:

The variability viewpoint expresses the commonalities and variability within the product line. Its main element is the feature, which is a user-visible aspect or characteristic of a system [9] (see Fig.1 top left).

The functional viewpoint contains the structure of a system represented by the SPL architecture and the core assets (software components) that satisfy the requirements of the different features (see Fig.1 top right).

The quality viewpoint includes a quality model for software product lines defined in [16]. This quality model extends the ISO/IEC 2500 (SQuaRE) standard [19], thus providing the quality assurance and evaluation activities in SPL development with support (see Fig.1 bottom left). The multimodel also permits the specification of the product line NFRs as constraints defined over the quality view, affecting characteristics, sub-characteristics and quality attributes [16]. The definition of NFRs as constraints in the quality view provides a mechanism for the automatic validation of their fulfillment once the software artifacts have been obtained.

The transformation view contains the explicit representation of the design decisions made in the different model transformation processes that integrate the production plan for a model-driven SPL (see Fig.1 bottom right). Alternatives appear in a model transformation process when a set of constructs in the source model admits different representations in the target model. The application of each alternative transformation could generate alternative target models that may have the same functionality but might differ in their quality attributes. In this work, we focus on architectural patterns [8], [13]. Architectural patterns specify the solutions to recurrent problems that occur in specific contexts [8]. They also specify how the system will deal with one aspect of its functionality, impacting directly on the quality attributes. Architectural patterns can be represented as architectural transformations as a means to ensure the quality of the product architectures.

The multimodel permits the definition of relationships among the elements in each viewpoint with different semantics, such as composition, impact or constraint relationships [15]. These relationships among the functional, variability, and quality views can be used to drive the product configuration, the core asset selection and the product architecture derivation processes. The relationships defined between the transformation view and the quality view, meanwhile, facilitate the use of the quality attributes as a decision factor when choosing from alternative pattern-based architectural transformations.

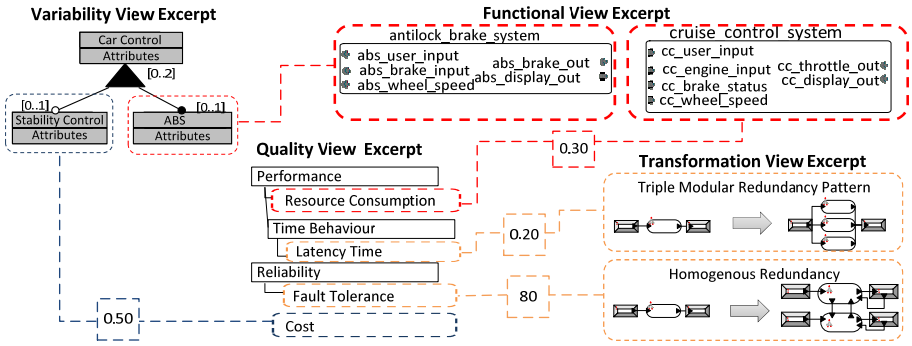

Fig.1 SPL Multimodel overview

The QuaDAI process includes different activities in which the multimodel is used to drive the model transformation processes for the derivation, evaluation and improvement of product architectures in SPL development. The activity diagram of the process supporting the approach is shown in Fig. 2.(a). It consists of the product architecture derivation from the product line architecture in the *Product Architecture Derivation* activity, its evaluation through the *Product Architecture Evaluation* activity and, in those cases in which the NFRs cannot be attained, its transformation through the application of pattern-based architectural transformations in the *Product Architecture Transformation Activity*. After this latter activity, the resulting architecture must again be revaluated using the *Product Architecture Evaluation Activity.*

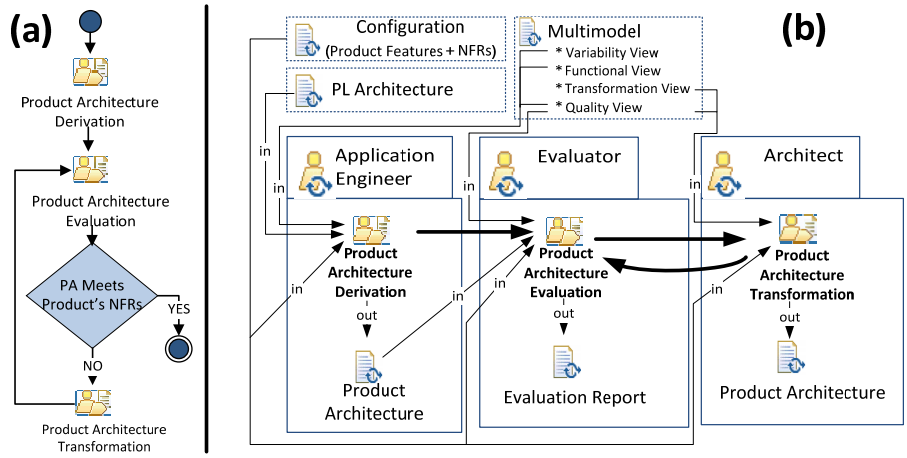

Fig. 2 Overview of the QuaDAI process

Product Architecture Derivation. The product architecture is derived from the product line architecture in the *Product Architecture Derivation* activity, taking as input the product line architecture, the variability and functional views of the multimodel, and the product configuration, containing both the product specific features and the product-specific NFRs selected by the application engineer (see Fig. 2.b). In this activity, the decision as to which functional components should be deployed in the product architecture is made by considering: i) the composition relationships be-

tween features and functional components; ii) the impact relationships between functional components and NFRs; and iii) the impact relationships between features and NFRs. The output of this activity is a first version of the product architecture which must be evaluated in order to analyze the attainment of non-functional requirements.

Product Architecture Evaluation. In the second model transformation process, the *Product Architecture Evaluation* applies the software measures described in the quality view of the multimodel to a product architecture in order to evaluate whether or not it satisfies the desired NFRs. This transformation takes as input the product architecture derived, the product specific NFRs and the quality view of the multimodel containing the metrics to be applied in order to measure the NFRs, generating as output an evaluation report (see Fig. 2.b).

Product Architecture Transformation. Finally, in those cases in which the non-functional requirements cannot be achieved by exercising the architectural variability mechanisms in the third activity, the *Product Architecture Transformation* applies pattern-based architectural transformations to the product architecture. The inputs of the *Product Architecture Transformation* are the product architecture, the relative importance of the different NFRs and the transformation view of the multimodel, containing the representation of the transformations to be applied. It generates a product architecture as output in an attempt to cover the NFRs prioritized by the architect (see Fig. 2.b). The architect introduces the relative importance of each NFR that the product must fulfill as normalized weights ranging from 0 to 1 as external parameters when executing the transformation. The transformation process uses the relative importance of each NFR and the impact relationships among transformations and quality attributes to select the architectural transformation to be applied.

3.2 The Architecture Trade-Off Analysis Method

The second method being analyzed is ATAM. The purpose of ATAM is to assess the consequences of architectural design decisions in the light of quality *attributes* [20]. ATAM helps in foreseeing how an attribute of interest can be affected by an architectural design decision. The quality attributes of interest are clarified by analyzing the stakeholder's scenarios in terms of stimuli and responses. Finally, ATAM helps to define which architectural approaches may affect quality attributes of interest. ATAM makes use of Utility Trees to translate the business drivers of a system into concrete quality attribute scenarios. Utility trees are a hierarchical structure in which the utility of a system is specified in terms of quality attributes which are further broken down into requirements and scenarios.

The main goals of the ATAM are to elicit and refine the architecture's quality goals; to elicit and refine the architectural design decisions and to evaluate the architectural design decisions in order to determine whether they address the quality attribute requirements satisfactorily. ATAM consists of nine steps that can be separated into four groups: i) *Presentation,* which involves the presentation of the method, the business drivers and the architecture being evaluated; ii) *Investigation and analysis,* which involves the identification of architectural approaches, the generation of the quality attribute utility tree and the analysis of the architectural approaches based on the high-priority scenarios identified in the utility tree; iii) *Testing,* which involves a

brainstorming and prioritization of the scenarios elicited in the utility tree, the analysis of the architectural approaches taking into account the high priority scenarios of the utility tree and the definition of the approaches to be applied, the risks and non-risks, sensitivity points and tradeoff points; and iv) *Reporting,* which involves presenting the results of ATAM.

The outputs of ATAM are: i) a prioritized statement of quality attribute requirements; ii) a mapping of approaches to quality attributes; iii) a catalog of the architectural approaches used; iv) risks and non-risks; v) quality-attribute-specific analysis questions, and vi) sensitivity points and tradeoff points [20].

4 The Original Study

The original experiment was designed by considering the guidelines proposed by Wohlin et al. [26]. According to the Goal-Question Metric (GQM) paradigm [6], the goal of the experiment was to **analyze** QuaDAI and ATAM **for the purpose** of evaluating them **with regard to** their effectiveness, efficiency, and perceived utility (measured by means of three subjective variables, ease of use, usefulness and intention of use) in order to obtain software architectures that meet a given set of quality requirements **from the viewpoint** of novice software architecture evaluators.

In this experiment we focus on the QuaDAI activities that occur after obtaining the product architecture: the *Product Architecture Evaluation* and the *Product Transformation* activities. These activities deal with the evaluation and improvement of product architectures, which are aligned with the main purpose of ATAM.

4.1 Context and Subject Selection

The software architectures to be evaluated are the software architecture of an Antilock Braking System *(ABS System)* from an automobile control system and the software architecture of the *Saavi* application (http://goo.gl/1Q49O), a mobile application for emergency notifications. We also selected a set of four architectural patterns that can be applied to improve the quality attribute levels in each of the product architectures. The experimental tasks include the evaluation of these quality attributes by means of two software metrics in each experimental object before and after applying the architecture evaluation methods. We used these two examples because the complexity of the system architectures was similar, the quality attributes to be promoted were the same and the complexity of the patterns was also similar. Thirty one subjects were selected from a group of fifth-year Computer Science students at the Universitat Politècnica de València who were enrolled on an Advanced Software Engineering course from September 2012 to January 2013, where they have eight hours of theoretical contents about Software Architectures and Architecture Evaluation.

4.2 Selected Variables

The **independent variable** of interest was the use of each method (ATAM or QuaDAI). There are two **objective dependent variables:** *effectiveness* of the method, which is calculated as a function of the *Euclidean Distances* between the NFR values

attained by the subject and the optimal NFR values that can be attained; and *efficiency*, which is calculated as the ratio between the effectiveness and the total time spent on the evaluation method. The *Perceived Utility* has been measured by means of three **subjective dependent variables:** *perceived ease of use (PEOU)***,** which refers to the degree to which evaluators believe that learning and using a particular method will be effort-free, *perceived usefulness (PU)*, which refers to the degree to which evaluators believe that using a specific method will increase their job performance within an organizational context and *Intention to Use (ITU)*, which is the extent to which a person intends to use a particular method. This variable represents a perceptual judgment of the method's efficacy – that is, whether it is cost-effective and is commonly used to predict the likelihood of acceptance of a method in practice. These three subjective variables were measured by using a *Likert* scale questionnaire with a set of specific closed questions related to each variable. The aggregated value of each subjective variable was calculated as the mean of the answers to the variable-related questions.

The Effectiveness is calculated by applying Formula (1) to normalized *Euclidean distances*. The normalization is calculated by applying Formula (2) to the *Euclidean distances, which is* calculated by applying Formula (3) and returns a value ranging from 0 to 1. The normalization is required in order to avoid the effects of the scales of the metrics that measure each NFR. The *Optimal* function in Formulas (1) and (2) returns the optimal values of the NFRs that can be achieved for a given experimental object. The *Max* function returns the maximal distance *D* observed for a given experimental object.

$$Effectiveness(p) = 1 - Norm\big(D(p, optimal(Object))\big) \quad (1)$$

$$Norm\big(D(p, Optimal(Object))\big) = \frac{D(p, Optimal(Object))}{Max(Object)} \quad (2)$$

$$D(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \quad (3)$$

The **hypotheses** of the experiment were:

- **$H1_0$:** There is no significant difference between the effectiveness of QuaDAI and ATAM / **$H1_a$:** QuaDAI is significantly more effective than ATAM
- **$H2_0$:** There is no significant difference between the efficiency of QuaDAI and ATAM / **$H2_a$:** QuaDAI is significantly more efficient than ATAM.
- **$H3_0$:** There is no significant difference between the perceived ease of use of QuaDAI and ATAM/ **$H3_a$:** QuaDAI is perceived as easier to use than ATAM.
- **$H4_0$:** There is no significant difference between the perceived usefulness of QuaDAI and ATAM / **$H4_a$:** QuaDAI is perceived as more useful than ATAM.
- **$H5_0$:** There is no significant difference between the perceived intention to use of QuaDAI and ATAM / **$H5_a$:** QuaDAI is perceived as more likely to be used than ATAM.

4.3 Experiment Operation and Execution

The experiment was planned as a balanced within-subject design with a confounding effect, signifying that the same subjects use both methods in a different order and with different experimental objects. We established four groups (each group applying one method on one object) and the subjects were randomly assigned to each group. The experiment was planned to be conducted in three sessions. On the first day, the subjects were given 120 minutes complete training on the evaluation methods and also on the tasks to be performed in the execution of the experiment. On the second and third days the subjects were given an overview of the training before they applying one evaluation method on an experimental object. We established a slot of 60 minutes without a time limit for each of the methods to be applied.

The experiment took place in a single room, and no interaction between subjects was allowed. The questions that arose during the session were clarified by the same conductors during the experiment.

Several documents were designed as instrumentation for the experiment: slides for the training session, an explanation of the methods, gathering data forms, the pattern description, the metric documentation, and two questionnaires. Excel spread sheets were also designed in order to automate the calculation of the metrics and the QuaDAIs trade-off among architectural transformations. The material of the experiments, including the metrics, the patterns and the NFRs to be fulfilled is available at http://www.dsic.upv.es/~jagonzalez/JISBD2013/instrumentation.

With regard to the data validation we verified that one of the subjects did not complete the 2nd session and therefore it was necessary to eliminate his first exercise. Since we had 30 subject distributed in four groups, it was additionally necessary to discard two subjects, selected randomly, to maintain the balanced design consisting of a total of 28 subjects, seven in each group.

4.4 Results

The results, obtained through descriptive statistics, lead us to interpret that QuaDAI was more effective and efficient, and also that it was perceived as being easier to use, more useful and more likely to be used by the subjects than ATAM. The cells highlighted in bold type in Table 1 show the best values for each of the statistics. In order to check the statistical significance of these tests we performed the Mann-Whitney non-parametric test so as to verify the statistical significance of the *Effectiveness*, *PEOU*, *PU* and *ITU* variables, since they are not normally distributed (Shappiro-Wilk normality test <0.05), and the 1-tailed *t*-test for independent samples to verify the statistical significance of the *Efficiency* variable (Shappiro-Wilk normality test >0.05).

Table 1. Descriptive results of the Original Experiment

	Effectiveness		**Efficiency**		**Duration(min)**	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
QuaDAI	**0.68**	0.39	**0.029**	0.018	**25.36**	7.26
ATAM	0.63	0.36	0.020	0.013	31.11	9.15

	PEOU		**PU**		**ITU**	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
QuaDAI	**3.98**	0.88	**3.80**	0.83	**3.65**	0.84
ATAM	3.50	0.82	3.72	0.73	3.55	0.70

The Mann-Whitney test results were 0.906 for *Effectiveness*, 0.030 for *PEOU*, 0.941 for *PU* and 0.767 for *ITU*. The *p*-value obtained from the 1-tailed *t*-test for *Efficiency* was 0.015. These results led us to conclude that the difference in terms of *Efficiency* and *PEOU* was statistically significant. However, with regard to the *Effectiveness*, *PU* and *ITU*, although the subjects achieved their best results with the QuaDAI method, we found that the differences were not statistically significant.

5 The Replication Study

To verify the remaining issues of the first study we conducted a replication of this experiment using a group of more experienced students. We used the same materials as in the original studies, with the addition of control questions to analyze the comprehension of the patterns and the metrics being applied. These questions help the subjects to focus on understanding the patterns and metrics and allow us to control their comprehension of the problem. We also changed one level of an NFR in the experimental object O2 since we realized that it in this experimental object it was easier to find the best solution (100% of the subjects when they dealt with O2 in the original study had selected the best pattern) as compared to the experimental object O1 (only 71% of the subjects had selected the best pattern, regardless the method).

5.1 Context and Subject Selection

The subjects were 19 students enrolled on a Masters' degree program in software engineering at the Universitat Politècnica de València. They were asked to perform the controlled experiment as part of the laboratory exercises conducted within the "Quality of Software Systems" course held from February to June 2013. The experiment took place during the first two weeks of March. We selected this course because was a specialized course in software quality and they have also more than eight hours of theoretical contents of *Software Architectures* and *Architecture Evaluation*.

5.2 Replication Design

The design of the replication was exactly the same as the original experiment, and we used the same variables to measure the effectiveness and efficiency and the same questionnaire to measure the subjective dependent variables. In order to maintain the balanced design, it was additionally necessary to randomly discard three subjects from the data to be analyzed, consisting of a total of 16 subjects, 4 in each group.

5.3 Data Analysis

The quantitative analysis was performed by using the SPSS v16 statistical tool using an α=0.05. A summary of the results of the evaluation is shown in Table 1. Mean and standard deviations have also been used as descriptive statistics for the qualitative subjective variables *PEOU, PU* and *ITU*. The five-point Likert scale ranging from 1 to 5 adopted for the measurement of the subjective variables has also been considered as an interval scale [9]. The cells highlighted in bold type in Table 2 show the best values for each of the statistics.

Since the sample size is smaller than 50 we applied the Shapiro-Wilk normality test to verify whether the data was normally distributed in order to select the tests needed to verify the hypotheses. For the *Effectiveness* and *Efficiency* variables, only *Efficiency* was normally distributed both for ATAM and QuaDAI (<0.05). In the case of the Subjective dependent variables, we analyzed the results of each method separately: for QuaDAI, PU was normally distributed, but neither *PEOU* nor *ITU* were (<0.05); and for the ATAM variables *PEOU*, *PU* and *ITU* were normally distributed.

Table 2. Descriptive results of the Replication Experiment

	Effectiveness		**Efficiency**		**Duration(min)**	
	Mean	**Std. Dev.**	**Mean**	**Std. Dev.**	**Mean**	**Std. Dev.**
QuaDAI	**0.797**	0.209	**0.023**	0.0069	**34.62**	7.974
ATAM	0.461	0.448	0.014	0.015	39.19	11.356
	PEOU		**PU**		**ITU**	
	Mean	**Std. Dev.**	**Mean**	**Std. Dev.**	**Mean**	**Std. Dev.**
QuaDAI	**4.312**	0.714	**4.167**	0.860	**4.063**	0.710
ATAM	4.020	0.714	3.927	0.672	3.906	0.831

The boxplots in Fig.3 containing the distribution of the dependent variables per subject per method show that QuaDAI performed better in each single variable. In order to determine the statistical significance of the results we applied: the Mann-Whitnney non parametric test to verify H1 (since QuaDAI's *Effectiveness* is not normally distributed), and the 1-tailed *t*-test for independent samples to verify H2; for the qualitative variables we applied the 2 tailed *t*-test in order to compare each variable with the test value 3 for the normally distributed variables (QuaDAI's *PU* and ATAM's *PEOU*, *PU* and *ITU*) and the one sample Wilcoxon test with the test value 3 for the non-normally distributed variables (QuaDAI's *PEOU* and *ITU*) to verify H3, H4 and H5 for each method separately.

The *p*-values obtained for the Mann-Whitnney test for *Effectiveness* was 0.036. The p-values obtained for the 1-talied *t*-test for *Efficiency* was 0.026. The respective *p*-values for the independent variables *PEOU*, *PU* and *ITU* were 0.01, 0.00 and 0.01, for the QuaDAI method and 0.00, 0.01 and 0.01 for ATAM. These results therefore support the rejection of all the null-hypotheses and the acceptance of their respective alternative hypotheses.

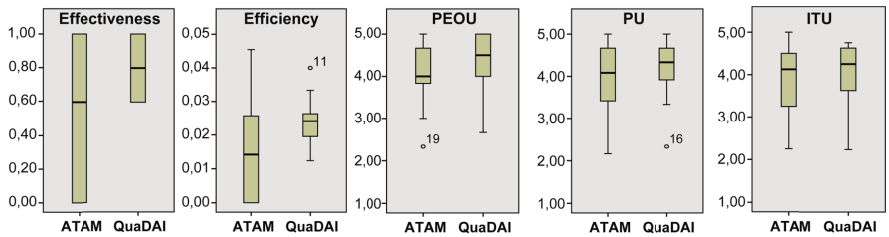

Fig.3 Boxplots for the various dependent variables

5.4 Discussion

The results of the two experiments, obtained through descriptive statistics, show the same tendency (QuaDAI performed better than ATAM) but with small differences. The subjects spent more time on the replication as compared to the original study.

This can be explained by the inclusion of the control questions associated with the patterns and metrics. It is also possible to observe differences in the effectiveness, particularly in the case of ATAM. As stated previously, we changed one level of an NFR in the experimental O2 in order to balance the difficulty with regard to the experimental object O1 (in the original study O2 was so much easier than O1), and in the case of ATAM we observed that the more difficult the decision was, the worse the subjects performed. Finally, in the case of the qualitative subjective variables it will be noted that the values on the replication study are higher. This can be explained by the subjects' level of experience, since in the replication study the subjects had more experience with which to assess the evaluation methods. On the other hand, in the replication study all the differences in the variables of the study were found to be statistically significant, as opposed to the original study. This can be due to the sample size, the differences on the subject's experience or simply due to random effects.

5.5 Threats to Validity

The main threats to the **internal validity** are: learning effect, subjects' experience, information exchange among participants, and understandability of the documents. Two different experimental objects were used to deal with the learning effect: ensuring that each subject applied each method with different objects and considering all the possible combinations of both the method order and the experimental objects. There were no differences in the subjects' experience since none of them had experience in architecture evaluations. Information exchange was alleviated by the use of different experimental objects and monitoring the subjects while they performed the tasks. Since the experiment was designed to take place in two sessions, the subjects might have been able to exchange information during the time between the sessions, but this was alleviated by asking the participants to return the material at the end of each session. The understandability of the material was alleviated by clearing up all the misunderstandings that appeared in each experimental session.

The main threat to **external validity** is the representativeness of the results, which might have been affected by the evaluation design, and the participant context selected. The evaluation design might have had an impact on the results owing to the kind of architectural models and quality attributes to be evaluated. We selected two different architectures, from two different domains, two different opposed NFRs and four different patterns for each experimental object. The experiment was conducted with students with no previous experience in architectural evaluations, and who received only limited training on the evaluation methods. However, since they were Master students they can be considered as novice users of architectural evaluation methods, and the results could thus be considered as representative of novice evaluators.

The main threats to the **construct validity** are the measures applied in the analysis and the reliability of the questionnaire. Euclidean distance has commonly been used to measure the goodness of a solution with regard to a set of non-functional requirements with different purposes [11]. The subjective variables are based on the Technology Acceptance Method (TAM) [12], a well-known and validated model for the evaluation of information technologies. The reliability of the questionnaire was tested by applying the Cronbach test. Questions related to *PEOU*, *PU* and *ITU* obtained

Cronbach's alphas of 0.889, 0.898 and 0.814, which are higher than the acceptable minimum required (0.70) [23]. The threats to the **conclusion validity** are the validity of the statistical tests applied. This threat was alleviated by applying a set of commonly accepted tests employed in the empirical software engineering community [23].

6 Conclusions and Further Work

This paper has presented a replication study for comparing the effectiveness, efficiency, and perceived utility, measured by means of the perceived ease of use, intention to use and perceived usefulness as regards participants using QuaDAI as opposed to ATAM. In contrast to the original study, in which the effectiveness, the intention to use and the perceived usefulness were not found to be statistically significant, in the replication study all the variables proved to be statistically significant.

These results suggest that QuaDAI can perform better for the evaluation and improvement of Product Architectures in Model-Driven SPL development scenarios.

We consider that this replication has been successful, since it has allowed us to validate the results of the first study and to analyze the method with a group of subjects with different level of experience. However, more replications are needed to analyze whether the differences found in the replication are extensible to other groups with different levels of experience.

As future work we plan to replicate this experiment with practitioners and new groups of students, and to perform a meta-analysis in order to aggregate the results with the data gathered from future replications.

Acknowledgements: This research is supported by the MULTIPLE project (MICINN TIN2009-13838) and the ValI+D fellowship program (ACIF/2011/235).

References

1. Ali-Babar, M., Lago, P., Van Deursen, A.: Empirical research in software architecture: opportunities, challenges, and approaches. Empirical Software Engineering. October 2011, Volume 16, Issue 5, pp 539-543 (2011)
2. Ali Babar, M., Zhu, L., Jeffery, R.: A Framework for Classifying and Comparing Software Architecture Evaluation Methods. In: 15th Australian Software Engineering Conference, April 13-16, 2004, Melbourne, Australia (2004)
3. Ali Babar, M., Kitchenham, B., Jeffery R.: Comparing distributed and face-to-face meetings for software architecture evaluation: a controlled experiment. Empirical Software Engineering, Int J 13 (1) pp 39-62 (2008)
4. Ali Babar, M., Winkler, D., Biffl, S.: Evaluating the Usefulness and Ease of Use of Groupware Tool for the Software Architecture Evaluation Process. In: First Int. Symposium on Empirical Software Engineering and Measurement, September 20-21, 2007, Madrid, Spain (2007)
5. Barkmeyer, E.J., Feeney, A.B., Denno, P., Flater, D.W., Libes, D.E., Steves, M.P, Wallace, E.K.: Concepts for Automating Systems Integration NISTIR 6928, National Institute of Standards and Technology, U.S. Dept. of Commerce, USA (2003)
6. Basili, V.R., Rombach, H.D.: The TAME project: towards improvement-oriented software environments. IEEE Transactions on Software Engineering 14 (6), pp 758–773 (1988)

7. Bosch, J.: Design and Use of Software Architectures. Adopting and Evolving Product-Line Approach. Addison-Wesley, Harlow (2000)
8. Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stal M.: Pattern-Oriented software architecture, Volume 1: A System of Patterns. Wiley (1996)
9. Carifio, J., Perla, R.J.: Ten Common Misunderstandings, Misconceptions, Persistent Myths and Urban Legends about Likert Scales and Likert Response Formats and their Antidotes. Journal of Social Sciences, Volume 3, Issue 3, 106-116 (2007)
10. Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns, Addison-Wesley, Boston, USA (2007)
11. Datorro, J.: Convex Optimization & Euclidean Distance Geometry. Meboo Publishing (2005)
12. Davis, F.D.: Perceived usefulness, perceived ease of use and user acceptance of information technology. MIS Quarterly 13 (3), pp 319–340 (1989)
13. Douglass, B. P.: Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems. Addison-Wesley, Boston, USA (2002)
14. Falessi D., Capilla, R., Cantone, G.: A Value-Based Approach for Documenting DesignDecisions Rationale: A Replicated Experiment. In: 3rd Int. Workshop on Sharing and Reusing Architectural Knowledge, May 10–18, 2008, Leipzig, Germany (2008)
15. Gonzalez-Huerta, J., Insfrán, E., Abrahão, S.: A Multimodel for Integrating Quality Assessment in Model-Driven Engineering. In: 8th Int. Conference on the Quality of Information and Communications Technology, September 3-6, Lisbon, Portugal (2012)
16. Gonzalez-Huerta, J., Insfrán, E., Abrahão, S., McGregor, J.D.: Non-Functional Requirements in Model-Driven Software Product Line Engineering. In: 4th Int. Workshop on Non-functional System Properties in Domain Specific Modeling Languages, Insbruck, Austria (2012)
17. Gonzalez-Huerta, J., Insfrán, E., Abrahão, S.: Defining and Validating a Multimodel Approach for Product Architecture Derivation and Improvement. In: 16 International Conference on Model Driven Engineering Languages and Systems, Miami, USA (2013)
18. Insfrán, E., Abrahão, S., González-Huerta, J., McGregor, J. D., Ramos, I.: A Multimodeling Approach for Quality-Driven Architecture Derivation. In: 21st Int. Conf. on Information Systems Development (ISD2012), Prato, Italy (2012)
19. ISO/IEC 25000:2005. Software Engineering. Software product Quality Requirements and Evaluation SQuaRE (2005)
20. Kazman, R.; Klein, M.; Clements, P.: ATAM: Method for Architecture Evaluation (CMU/SEI-2000-TR-004, ADA382629). Software Engineering Institute, Carnegie Mellon University, (2000)
21. Martens, A., Koziolek, H., Prechelt, L., Reussner, R.: From Monolithic to Component-Based Performance Evaluation of Software Architectures: A Series of Experiments Analyzing Accuracy and Effort. In: Empirical Software Engineering, October 2011, Volume 16, Issue 5, pp 587-622 (2011)
22. Martensson, F.: Software Architecture Quality Evaluation. Approaches in an Industrial Context. Ph. D. thesis, Blekinge Institute of Technology, Karlskrona, Sweden (2006)
23. Maxwell, K.: Applied Statistics for Software Managers. Software Quality Institute Series, Prentice Hall (2002)
24. Robertson, S., and Robertson, J.: Mastering the requirements process. ACM Press (1999)
25. Roy, B., Graham, T.: Methods for Evaluating Software Architecture: A Survey. Technical Report 545, Queen's University at Kingston, Ontario, Canada, (2008)
26. Wohlin, C., Runeson, P., Host, M., Ohlsson, M.C., Regnell, B., Weslen, A.: Experimentation in Software Engineering - An Introduction, Kluwer (2000)