González Huerta, J.; Insfrán Pelozo, CE.; Abrahao Gonzales, SM.; Scanniello, G. (2015). Validating a model-driven software architecture evaluation and improvement method: A family of experiments. Information and Software Technology. 57:405-429. doi:10.1016/j.infsof.2014.05.018.

Validating a Model-Driven Software Architecture Evaluation and Improvement Method: a Family of Experiments

Javier Gonzalez-Huerta[1], Emilio Insfran[1], Silvia Abrahão[1], Giuseppe Scanniello[2]

[1] Department of Information Systems and Computation, Universitat Politècnica de València, Valencia, Spain
[2] DiMIE, Università della Basilicata, Potenza, Italy

Abstract:

Context: Software architectures should be evaluated during the early stages of software development in order to verify whether the Non-Functional Requirements (NFRs) of the product can be fulfilled. This activity is even more crucial in Software Product Line (SPL) development, since it is also necessary to identify whether the NFRs of a particular product can be achieved by exercising the variation mechanisms provided by the product line architecture or whether additional transformations are required. These issues have motivated us to propose QuaDAI, a method for the derivation, evaluation and improvement of software architectures in model-driven SPL development.

Objective: We present in this paper the results of a family of four experiments carried out to empirically validate the evaluation and improvement strategy of QuaDAI.

Method: The family of experiments was carried out by 92 participants: Computer Science Master's and undergraduate students from Spain and Italy. The goal was to compare the effectiveness, efficiency, perceived ease of use, perceived usefulness and intention to use with regard to participants using the evaluation and improvement strategy of QuaDAI as opposed to the Architecture Tradeoff Analysis Method (ATAM).

Results: The main result was that the participants produced their best results when applying QuaDAI, signifying that the participants obtained architectures with better values for the NFRs faster, and that they found the method easier to use, more useful and more likely to be used. The results of the meta-analysis carried out to aggregate the results obtained in the individual experiments also confirmed these results.

Conclusions: The results support the hypothesis that QuaDAI would achieve better results than ATAM in the experiments and that QuaDAI can be considered as a promising approach with which to perform architectural evaluations that occur after the product architecture derivation in model-driven SPL development processes when carried out by novice software evaluators.

Keywords: Software Architectures, Software Architecture Evaluation Methods, Quality Attributes, ATAM, Family of Experiments, Meta-analysis.

1. Introduction

Software architectures are a means to preclude or permit the achievement of the Non-Functional Requirements (NFRs) of a software system. In Software Product Line (SPL) development, in which a set of software intensive systems sharing a common managed set of features are developed from a common set of core assets, the product line architecture should contain variation mechanisms that help to achieve a set of permitted variations, including functional, structural and quality concerns [23]. The product architecture is derived from the product line architecture by exercising its built-in architectural variation mechanisms, which support both the functional and NFRs for a specific product.

Once it has been derived, the product architecture should be evaluated to assess the achievement of the product's specific requirements. 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 [15].

Various studies concerned with the derivation (e.g., [14], [61]) and/or evaluation of software architectures from several points of view (e.g., [65], [37], [70], [66]) have been proposed in literature. After reviewing these studies, we have observed that:

(a) There is a lack of systematic methods that model the impact between architectural design decisions and quality attributes to support the integrated derivation, evaluation and quality enhancement of software architectures.

(b) In the software architecture field, there is a lack of empirical evidences regarding the advantages of tools and methods [9]. Software architecture researchers must follow a two-pronged strategy: develop new techniques, methods or tools with which to improve on current practices, and perform systematic, rigorous assessments of existing and new techniques by following the empirical paradigm [31].

We have addressed the first issue by proposing the Quality-Driven Architecture Derivation and Improvement (QuaDAI) method in previous studies [40], [41]. QuaDAI is a model-driven approach to ensure the desired quality attribute levels for a product by applying architectural transformations to a product architecture derived from a product line architecture.

With regard to the second issue, in previous works, we have presented a first controlled experiment [41] and a replication study [42] as an initial step in the empirical validation of the QuaDAI strategy for the evaluation and improvement of product architectures that consist on the evaluation and transformation activities. The objective of these experiments was to compare the effectiveness, efficiency, perceived ease of use, perceived usefulness and intention to use with regard to participants using the product evaluation and transformation activities of QuaDAI (from now on QuaDAI) as opposed to the Architecture Tradeoff Analysis Method (ATAM) [50], a well-known and widely-used software architecture evaluation method. The results of the first 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. 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.

Two further replications were therefore conducted in order to provide more evidence about the validity of these results. These experiments were conducted with Computer Science Master's and undergraduate students from the Universitat Politècnica de València (UPV) in Spain and with Information Science undergraduate students from the Università degli Studi della Basilicata in Italy. All the experiments conducted form a family[1] of controlled experiments. The objective of this paper is, therefore, to report the results of a family of four controlled experiments with the aim of empirically validating the effectiveness, efficiency, perceived ease of use, perceived usefulness and intention to use with regard to participants using QuaDAI as opposed to ATAM. We have also carried out a meta-analysis in order to aggregate the results obtained in the individual experiments to provide more general conclusions.

This paper is organized as follows. Firstly, related works on the empirical validation of software architecture evaluation methods are discussed in Section 2. The architecture evaluation methods that were evaluated in the family of experiments (QuaDAI and ATAM) are introduced in Section 3. The family of experiments is described in Section 4. The details of the individual design of each experiment are provided in Section 5. The results of each experiment are reported and analyzed in Section 6. The results of the family of experiments are summarized in Section 7, together with those of the meta-analysis. Threats that might affect the validity of our results are discussed in Section 8. Finally, our conclusions and final remarks are presented in Section 9.

2. Related Work

The increasing size and complexity of software systems, along with the demand for high-quality systems, has driven the increased interest in the software architecture sub-discipline of software engineering [2]. In this context, several methods and tools with which to support the different activities in the architecture design and evaluation processes have been proposed [9]. However, little attention has been paid in this field to the empirical validation of the methods and tools proposed. Instead, in general, the empirical studies in

[1] The concept of replication is extended to the "family of experiments" reported by Basili et al. [12]. A family is composed of multiple similar experiments that pursue the same goal to build the knowledge needed to extract significant conclusions [1].

this field are focused on establishing taxonomies for classifying the methods or assessing specific aspects of a given evaluation method. In this section, we discuss related works that report on comparisons of architecture evaluation methods and empirical studies that assess various aspects of the architectural evaluation process.

2.1. Classification Frameworks for Comparing Software Architecture Evaluation Methods

Several studies comparing or establishing frameworks with which to compare software architecture evaluation methods have recently been reported in literature. The first literature review by Ali Babar et al. [2] compared four scenario-based software architecture evaluation methods: Scenario-Based Architecture Analysis (SAAM), the Architecture Level Modifiability Analysis (ALMA), the Performance Assessment of Software Architecture (PASA) and the Architecture Trade off Analysis Method (ATAM). In a subsequent work, the same authors presented an extension of this comparison in which they established a framework that could be used to characterize software architecture evaluation methods based on a literature review [3]. In this later study, the authors applied their framework in order to compare eight evaluation methods (e.g., SAAM, ATAM or ARID). In both studies, the authors followed a fifteen criteria schema (e.g., the maturity stage, the particular definition behind the method, the process support, the method's activities). This framework was also evaluated in Ali Babar and Kitchenham [4] through the use of a survey whose objective was to analyze the suitability of the elements in the framework. The results of this survey supported the majority of the elements in the framework, and only in a few of them (i.e., tool support, method activities and application domain) there were disagreements among the participants.

Roy and Graham [67] presented a survey in which they reviewed thirty-seven software architecture evaluation methods. They also established a taxonomy for their classification based on the development phase in which they are applied (i.e., early vs. late), the main analysis technique applied or the artifacts analyzed (i.e., scenario-based, mathematical models or metric-based), and their ability to deal with styles and patterns. The main conclusions were: i) it is difficult to be proficient in the use of architectural evaluation methods; ii) there is a lack of tools supporting the methods; and iii) the majority of the methods (except SAAM and ATAM) have not been empirically validated.

Etxebarria and Sagardui [30] presented a framework based on a literature review in order to classify sixteen software architecture evaluation approaches and techniques specifically defined for SPL development environments. This framework classifies software architecture evaluation approaches based on the evaluation time (i.e., design time vs. evolution time), the architecture being evaluated (product line architecture vs. product architecture) and the purpose of the evaluation (e.g., evolution-related product line architecture evaluation, evaluation during derivation, synchronization-related evaluation).

Finally, Breivold et al., [16] presented a systematic literature review in which, among other topics, the authors covered the quality evaluation of software architectures focusing on the evolution aspects. They principally focused on assessing experience-based, scenario-based and metric-based evaluation methods that are able to deal with evolvability. One of their conclusions was that the techniques that support quality considerations help to identify key quality attributes early in the software design phase. They also encouraged the definition of methods and tools with which to design (and manage) software architectures for ultra-large-systems (e.g., SPL).

The comparisons mentioned above provide an analysis of the characteristics of the methods under analysis, which can help practitioners and researchers to attain a holistic view of the methods available. However, they do not provide factual data as to which method is most efficient, effective, easy to use or useful for a given type of project or development scenario. The majority of these works report surveys that are based solely on subjective information and do not follow a predefined methodology. Our research, on the other hand, provides factual objective and subjective information collected by following a well-defined methodology.

2.2. Empirical Studies Assessing Software Architecture Evaluation

Despite the fact that the interest in the software architecture field has increased over the last few years, few experiments have been conducted to analyze different aspects of software architecture evaluation processes (e.g., [5], [7], [8], [32], [33], [39], [53]), or empirical validations through case studies or experience reports (i.e., [10], [64], [74]). A summary of each of these studies is presented below in chronological order.

Golden et al. [39] reported on a controlled experiment analyzing the value of the different parts of a usability supporting architectural patterns in the modification of a software architecture design. In this study the authors evaluated how the architectural solutions produced as a result of using a more complete

specification of a pattern better support the usability needs. The study demonstrates that the use of more complete specification of the patterns increases the effectiveness and efficiency of usability evaluations.

Various empirical studies evaluating the influence of team size, organization and support as regards the communication among the members of teams have been reported (i.e., [7], [5], [6]). Ali Babar and Kitchenham [7] reported on a controlled experiment to analyze the impact of group size on the outcome of a software architecture evaluation exercise. They analyzed how the group size affects both the quality of scenario profiles and the participants' satisfaction with the process and outcomes. The principal result of this study was that the size of the group affects the quality of the scenarios created. There were also disagreements as to the group size with which the subjects obtained scenarios with the best quality (i.e., groups of five participants) and the group size with which the participants were most satisfied (i.e., groups of three participants).

Ali Babar et al. [5] presented an experiment comparing distributed and face-to-face meetings within the software architecture evaluation process. The objective of this study was to assess the effectiveness of the proposed groupware-supported process in the development of high quality scenarios during the evaluation process. In a similar study, Ali Babar et al. [6] reported the results of an experiment assessing the use of LiveNet, a groupware tool that can be used to support the software architecture 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. The results of the first experiment showed that the quality of the scenario profiles developed by distributed teams using a groupware tool were significantly better than those developed by face to face teams. The results of the second study showed that the participants found the use of the groupware tool positive in distributed meetings.

Falessi et al. [32] reported the results of a controlled experiment and its replication [33] with the aim of analyzing the perceived utility of the information associated with Architectural Design Decisions Rationale Documentation (DDRD), an artifact with which to document architectural design decisions. The participants were requested to perform different activities (described using the DDRD Use Cases) and to then rank the categories of information in DDRDs (e.g., issue, decision, status, constraints, related requirements). The results showed that the perceived importance of the different information categories in DDRDs depend on the activity that the DDRD is helping to conduct [32], and that the DDRD should contain only the information required to perform that activity [33].

Ali Babar [8] presented the assessment of the Architectural Level Security Analysis Framework (ALSAF), performed with a pilot study and a quasi-experiment. The goal of the study was to identify security attributes and the security design patterns that are suitable to attain these attributes based on a given list of security properties. In this study, the control group only had access to the software requirement specification (SRS), whereas the treatment group had access to both the SRS and ALSAF. The results show that the participants using ALSAF obtained significantly better results as regards identifying security attributes and patterns, and also that they found ALSAF useful when performing these tasks.

Martens et al. [53] reported a series of three separated controlled experiments comparing the accuracy and required effort when applying different software architecture performance evaluation methods. The aim of the study was to compare three monolithic performance evaluation methods (i.e., SPE, CP and umlPSI) with the component based performance evaluation method (PCM). The results showed that, in terms of accuracy, PCM, SPE and CP produced similar results, and that umlPSI produced over estimations, whereas the application of PCM required more effort.

Although the intention of the aforementioned studies was to gather empirical knowledge through experiments or quasi experiments, it will be observed that the majority of them are focused on specific aspects of the architecture evaluation process or on assessing how a given treatment improves performance. There is, however, a lack of empirical validations of the methods and tools being proposed, through a comparison with the existing body of knowledge.

Finally, there are several works reporting experiences on the application of ATAM with different purposes (i.e., [10], [64], [74]). Reijonen et al. [64] presented an experience report describing the application of ATAM in eleven architecture evaluations in real industrial projects. In this work, the authors provide a detailed description of the application steps, the schedule followed in the evaluations, the problems confronted during the evaluation and the main benefits perceived by the stakeholders. Svahnberg and Martensson [74] reported their experiences in academic architectural evaluations in student projects. They applied a lightweight software architecture evaluation method adapted from SAAM and ATAM. The architecture evaluations were applied both to assess the architectures of the projects developed by the students and to teach software architecture evaluation. Barbaci et al. [10] presented a case study in which a product line architecture in the avionics domain was evaluated. These works contribute towards

demonstrating the feasibility of ATAM; all of them highlight the importance of conducting software architecture evaluations, emphasize the importance of the utility tree (although Reijonen et al. point out the difficulties found by the evaluators during its creation) and how critical the quality of scenarios is to the success of the evaluation.

2.3. Discussion

The analysis of the aforementioned studies has allowed us to identify some limitations in the empirical validation of software architecture evaluation methods, such as: i) the low number of empirical studies assessing the approaches being defined; ii) the fact that the quantitative and qualitative comparisons with existing methods has been neglected; and iii) the fact that the majority of the empirical studies tend to be isolated and not replicated.

The first limitation is in line with the findings of Ali Babar et al. [9], Falessi et al. [31] and Dyba et al. [29] in which the authors claim that the majority of the approaches being presented in the software architecture field lack empirical validation. The few available validations of the approaches being proposed are based on toy examples, case studies or experience reports, as discussed by Qureshi et al. [63], and few of them are validated through controlled experiments. Moreover, in this field we have found that most of the software architecture papers used incorrect terminology (e.g., they used the term experiment rather than experience report, as in the works of Niemela and Immonen [59] or Martesson [54]; or the term case study when documenting proof of concepts without methodical or data extraction descriptions [63]).

The second limitation concerns the lack of quantitative or qualitative comparisons with existing methods. The proposed methods and tools that are defined have not been compared with the existing alternatives in the software architecture evaluation body of knowledge. There are various frameworks (see Section 2.1) with which to classify software architecture evaluation methods; however, there is a lack of empirical studies in which the authors analyze how the methods perform as compared with similar ones.

The third limitation is in line with studies that have been performed in the Software Engineering field, such as that by Sjøberg et al. [72]. This work claims that only 20 out of 113 controlled experiments are replications. A replication is the repetition of an experiment to confirm findings or to ensure accuracy. There are two types of replications: close replications, also known as strict replications (i.e., replications that attempt to keep almost all the known experimental conditions much the same or at least very similar), and differentiated replications (i.e., replications that introduce variations in essential aspects of the experimental conditions, such as executions of replications with different kinds of participants) [52]. Both types of replications are necessary to achieve a greater validity of the results obtained from empirical studies. The problem of dealing with experimental replications has been addressed with the concept of the family of experiments. Although many empirical studies have been applied in the software architecture evaluation field, few families of experiments have been reported so far.

3. Software Architecture Evaluation Methods

The two software architecture evaluation methods evaluated in our family of experiments are: the ATAM [50] and our proposal QuaDAI [41], both of which are introduced in the following subsections.

Both ATAM and QuaDAI can be classified as early architecture evaluation methods, capable of deal with multi-attribute evaluations, following the taxonomy of architectural evaluation methods described by Roy and Graham [67]. Although ATAM was initially defined to assess general purpose software architectures, it can be used in an SPL environment to assess both the product line architecture and the product architectures at derivation time at various stages of SPL development (conceptual, before code, during development or after deployment) [23], [56], [68] taking into account multiple quality attributes, as is shown in various experience reports (e.g., [10], [64], [68], [74]). QuaDAI is, meanwhile, focused on the evaluation and improvement of product architectures once they have been derived from the product line architecture.

We opted for ATAM as the baseline for the following reasons:

- It is a widely-used software architecture evaluation method [54].
- It has been widely applied and validated [67] in both industry (e.g., [64], [10]) and academic environments (e.g., [74]).
- It is capable of addressing multiple attribute analysis [1].
- It performs tradeoff analyses among quality attributes and design decisions [50]. ATAM extend previous methods such as SAAM to cover the tradeoff among competing quality attributes [54].

- It is capable of reengineering the software architecture and employs activities to extract architectural styles or design patterns [67] whereas other architectural evaluation methods (e.g. ARID) are focused on evaluating the suitability of a portion of the architecture to be used by the developers to complete their tasks [22].

3.1. ATAM

The purpose of ATAM is to assess the consequences of architectural design decisions in the light of quality attributes [50]. ATAM assists 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 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. A summary of these phases and steps and the main generated artifacts is shown in Fig. 1.

Finally, the outputs of ATAM are: i) a prioritized statement of quality attribute requirements; ii) a mapping of approaches onto quality attributes; iii) a catalog of the architectural approaches identified and used; iv) risks and non-risks; v) quality-attribute-specific analysis questions; and vi) sensitivity points and tradeoff points [50].

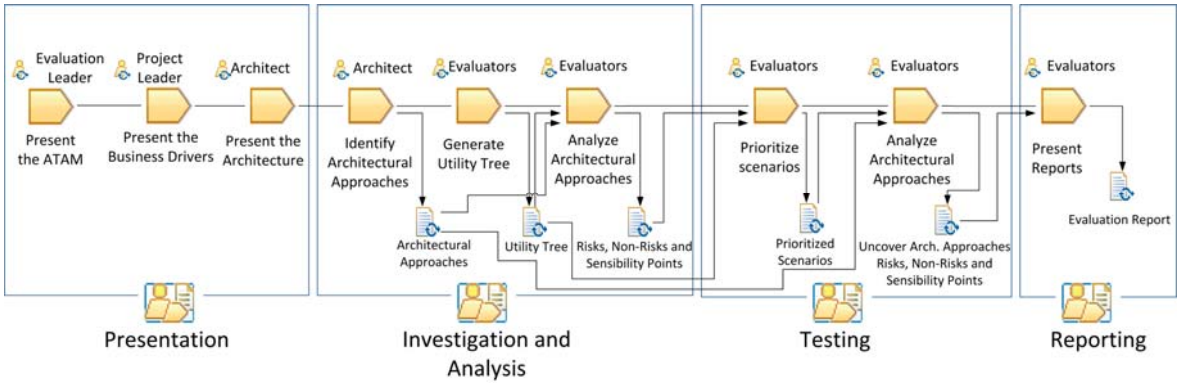

Fig. 1. Summary of the ATAM phases and activities

3.2. QuaDAI

QuaDAI is a generic, integrated method for the derivation and improvement of product architectures regardless the Architectural Description Language in which they are expressed or the domain. It is based on an artifact (the multimodel [41]) that represents the SPL viewpoints and a process consisting of a set of activities conducted by model transformations. QuaDAI has been designed by taking into account the weak points of existing architecture evaluation methods in order to improve their usability and effectiveness (e.g., the need for a highly experienced team: QuaDAI relies on knowledge reuse, which allows a less skilled evaluator to perform architecture evaluations using the domain expert's knowledge).

In QuaDAI, a multimodel permits the explicit representation of relationships among entities in different viewpoints. A multimodel is a set of interrelated models that represent 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 [11]. Such a model is known as a viewpoint model, 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 a final product is derived, the relationships drive the different model transformation processes that constitute the production plan [23] used to produce a product architecture.

The multimodel used to specify SPLs is composed of (at least) four interrelated viewpoints:

- **The variability viewpoint**, which 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 [23].
- **The architectural viewpoint**, which contains the architectural variability of the Product Line architecture that realizes the external variability of the SPL expressed in the variability viewpoint. This variability can be defined on the different architectural viewpoints. It is expressed by means of the Common Variability Language (CVL) [60], which is a generic language for expressing variability on a modeling language. Its main element is the architectural variation point.
- **The quality viewpoint**, which includes a quality model for SPL defined in [43]. This quality model extends the ISO/IEC 25000 standard (SQuaRE) [47], thus providing the quality assurance and evaluation activities in SPL development with support. The multimodel also permits the specification of the product line and the product specific NFRs as constraints defined over the quality model, affecting characteristics, sub-characteristics and quality attributes [43]. The NFRs can incorporate variability in terms of thresholds (i.e., the interval in which the product quality attribute levels can vary). The explicit representation of the NFRs in the multimodel provides a mechanism for the automatic validation of NFR fulfillment once the software artifacts have been obtained [40].
- **The transformation viewpoint** 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. 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 [18] and [28]. Architectural patterns specify the solutions to recurrent problems that occur in specific contexts [18]. 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 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. 5(a). It consists of the following activities:

- **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 quality, the variability and the architectural viewpoints 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. 5(b)). In this activity, the decision as to which architectural variation points should be resolved in the product architecture is made by considering: i) the composition relationships between features and architectural variation points; ii) the impact relationships between architectural variation points and NFRs; and iii) the impact relationships between features and NFRs. The transformation generates the CVL resolution model that is used to generate the first version of the candidate architecture, through a CVL transformation [40]. This activity comprises the configuration of the product, the consistency

checking of the configuration, taking into account the variability and quality constraints but also the inter-viewpoint relationships. This validation, together with a consistency checking of the obtained models allows us to assure that the obtained product architecture is well-formed [40]. Once derived, the product architecture should be evaluated in order to analyze the attainment of non-functional requirements. The QuaDAI derivation activity has been preliminary empirically validated through two case studies [40].

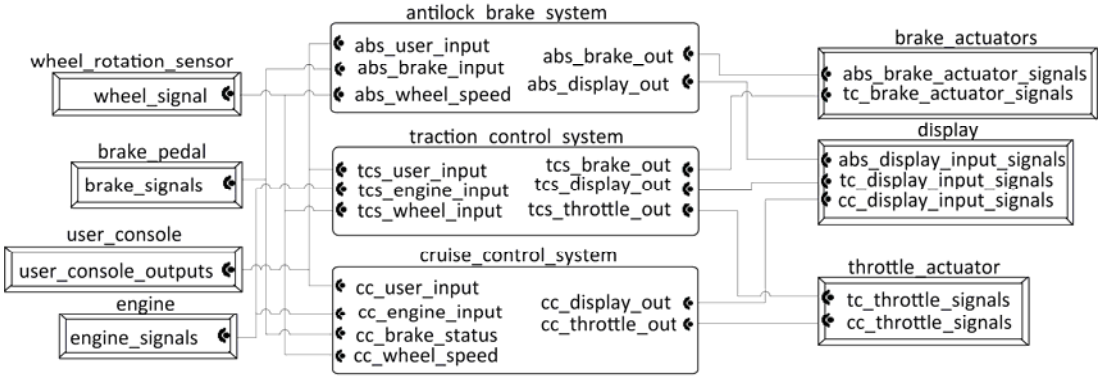

Fig. 2 Excerpt of a Product Line Architecture

The activities of the QuaDAI process are illustrated through the use of a running example: a SPL from the automotive domain that comprises the safety critical embedded software systems responsible for controlling a car. Fig. 3 shows the product architecture derived from the product line architecture (shown in Fig. 2) generated by the *Product Architecture Derivation* for the automotive example when the application engineer selects only the *antilock_brake_system* feature and introduces the product specific NFRs, which come from the system's requirements, demanding a fault tolerance of the ABS greater than 99.5% and restricting the ABS latency time to 5ms.

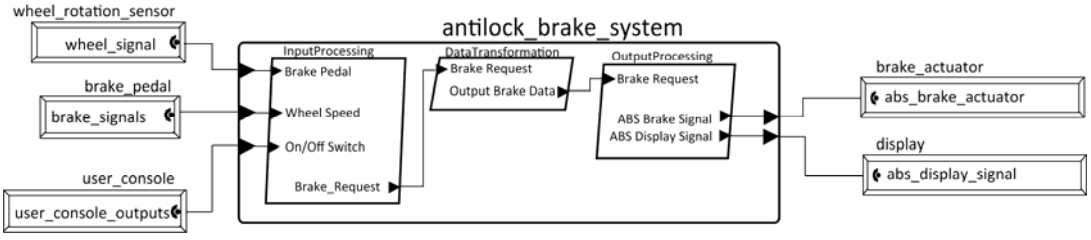

Fig. 3 Portion of the Product Architecture showing the ABS system

- **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 the product architecture in order to evaluate whether or not it satisfies the desired NFRs. This evaluation actually measures the degree of fulfillment of the NFRs defined during the configuration on the product architectural models [40].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. 5(b)). The method relies on analyzing the derived product architectural models instead of relying on a set of predicted values, based on a previous measurement process of a sample of products as in [65], [37], [70]. Those approaches could have scalability concerns, due to the exponential growth of the number of configurations as a function of the number of features and they also fail on managing those products that refine or extend the NFRs of the product line with delta requirements specific for the product under development. Following the automotive example *t*he evaluation for the architecture shown in Fig. 3 may conclude that the architecture meets the latency NFR but that the fault tolerance NFR is not achieved, and architectural transformations may thus be required.

- **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* automatically applies pattern-based architectural transformations to the product architecture. These architectural transformations can be applied to different architectural viewpoints, depending on the nature of the patterns being considered. 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 transformations to be applied. It generates a product architecture as output in an attempt to cover the NFRs prioritized by the architect (see Fig. 5(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 in order to improve the architecture quality attribute levels. These architectural transformations may help achieving the NFRs. In the automotive example, if the architect selects both the latency and the fault tolerance as being of equal importance (i.e., with a weight of 0.5 for each one) the transformation process will select the Triple Modular Redundancy pattern (TMR). The architecture resulting from the application of the TMR pattern is shown in Fig. 4**¡Error! No se encuentra el origen de la referencia.**.

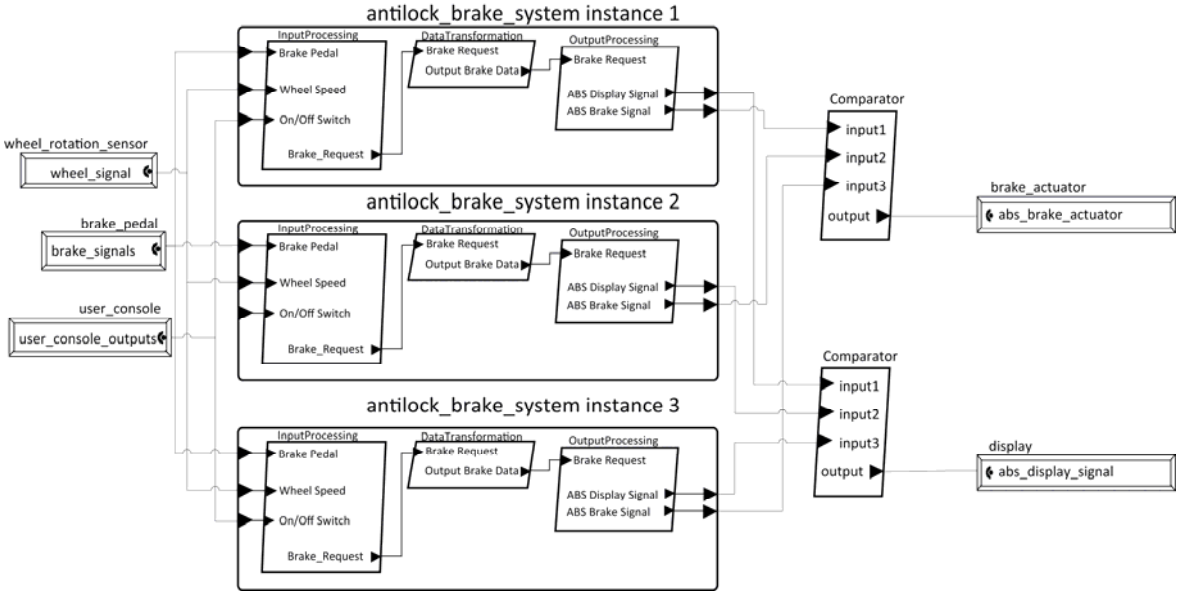

Fig. 4 Product architecture after applying the TMR pattern

The process iterates until the NFRs are achieved or when the architect detects that it is not possible to build the product with the set of NFRs selected in the configuration (Fig. 5a(1)). The evaluation process may result also in a renegotiation of the NFRs with the customer (Fig. 5a(2)). In this case, the product architecture should be re-evaluated to check the conformance with the new NFRs. Finally, in some cases the architect should vary some architectural variation points to modify the candidate product architecture. For instance, in some cases the first candidate architecture may imply the positive resolution of a set of architectural variation points that may lead to a quality attribute levels that are far above of a given NFR. Considering another combination of architectural variation points may also imply the fulfillment of that specific NFR but also other that were previously unfulfilled.

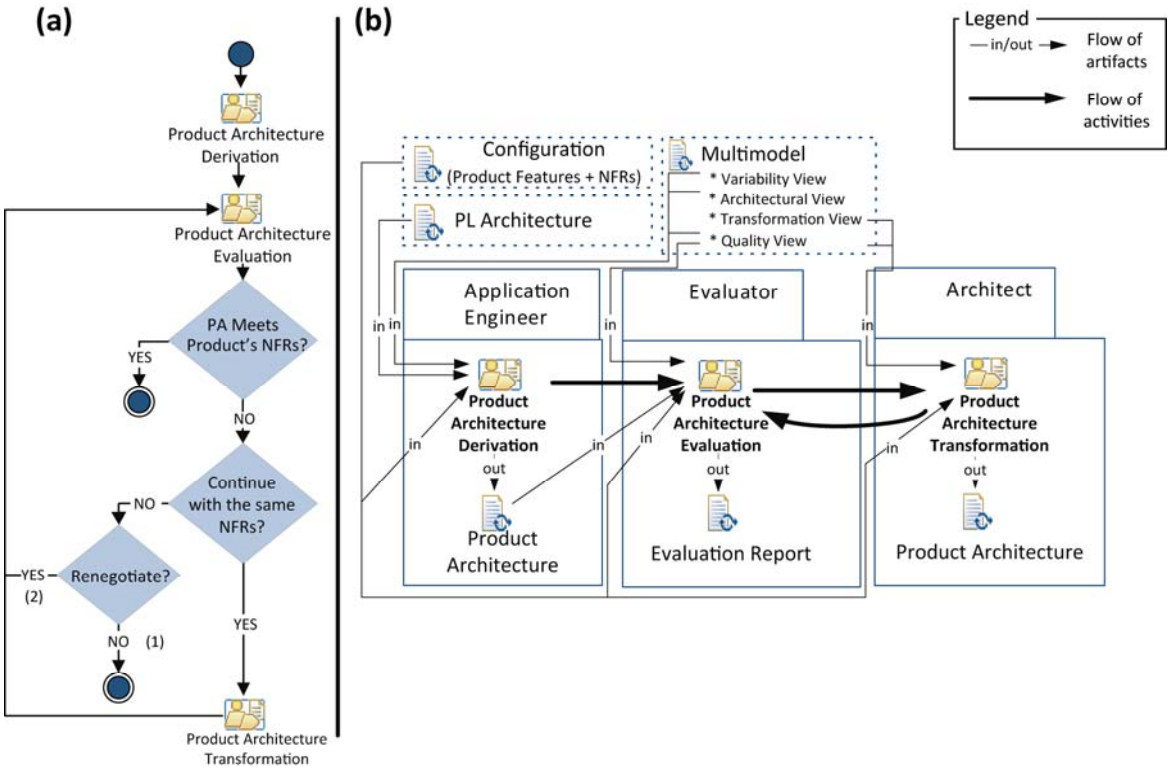

Fig. 5. Overview of the QuaDAI process

4. Overview of the Family of Experiments

In this section, we present the family of experiments conducted to empirically validate the evaluation and transformation activities of QuaDAI. The methodology adopted is an extension of the five-steps proposed by Ciolkowski et al. [20], in which the fifth step, family data analysis, has been replaced with "family data analysis and meta-analysis". Each experiment was designed according to the experimental process proposed by Wohlin et al. [78].

4.1. Step 1: Experiment Preparation

According to the Goal-Question Metric (GQM) paradigm [13], the goal of the family of experiments is to **analyze** the evaluation and transformation activities of QuaDAI and ATAM **for the purpose** of comparing them **with respect to** their effectiveness, efficiency, 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 the context** of undergraduate and postgraduate students in Computer Science.

4.2. Step 2: Context Definition

The context of the family of experiments is the quality evaluation of two software architectures carried out by novice evaluators. The context is defined by i) the software architecture to be evaluated; ii) the architectural evaluation method and iii) the selection of participants.

4.2.1. Software Architecture Evaluated

The software architectures to be evaluated in the family of experiments are two architectures from two domains: the software architecture of an Antilock Braking System (ABS System) from an automotive control system SPL and the software architecture of the Savi application (http://goo.gl/1Q49O), which is a mobile application for emergency notifications.

The architecture of the ABS System, represented through its *Component and Connector* view [21] expressed in AADL [34], was selected as experimental object O1[2], and the Savi architecture, represented through the *Deployment* view, was selected as experimental object O2. We also selected a set of four architectural patterns that can be applied to improve the quality attribute levels of interest for each of the software architectures from two pattern catalogues ([28] for the automotive domain and [57] for the emergency management domain). The corresponding selected patterns are intended to improve the specific NFRs specified in each experimental object. Some of these architectural patterns may impact on several quality attributes of interest, so as to force the subjects in performing tradeoffs. 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. Table 1 shows the details of the experimental objects used in the individual experiments. The rationale behind the selection of these two architectures is to have different problems, in different domains and that deal with aspects represented in different architectural viewpoints, so as to have an analysis that is not influenced by one specific domain or one specific architectural concern. In addition, these two architectures are comparable in terms of size and complexity.

Table 1 Experimental objects details

			Software Metrics		
	Architectural View	***NFRs***	***Reliability***	***Performance***	***Architectural Patterns***
Object O1: ABS System	Component & Connector	Reliability, Performance	Failure Probability	Latency Time	Watchdog, Homogenous Redundancy, Sanity Check and Triple Redundancy [28]
Object O2: Savi Application	Deployment View	Reliability, Performance	Uptime	Workload	Load Balancer, Symmetric Cluster, Asymmetric Cluster, Failover Cluster [57]

4.2.2. Architecture Evaluation Methods Compared

In this family of experiments 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. Before its execution, the authors first performed the domain expert role for the QuaDAI method, which includes i) the selection of the architectural patterns for each domain; ii) the selection of the quality attributes and the metrics that measure each quality attribute; iii) the execution of the tradeoff process between architectural patterns and quality attributes for the QuaDAI application and iv) the storage of the tradeoff results in the multimodel. These activities were carried by the authors since it was required for the subjects to have a full description of the problem, and since we did not expect the subjects to have a deep knowledge on the domain to identify the metrics and architectural patterns and be able to perform this tradeoff.

The activities selected from the ATAM method to be included as experimental tasks were the analysis of *Analysis of Architectural Approaches* from the investigation and analysis phase, the *Scenario Prioritization* and the second *Analysis of Architectural Approaches* from the testing phase. Before its execution, the authors first performed the architect role to identify the architectural approaches, along with the first step performed by the evaluators: the generation of the utility tree. These activities were carried by the authors since, as for QuaDAI, it was required for the subjects to have a full description of the problem, and since we did not expect the participants to have a deep knowledge on the domain to identify the architectural approaches by themselves from the scratch.

In order to have a fair comparison between both methods, we provided the multimodel in the case of QuaDAI containing the architectural transformations and their impact on the NFRs and the utility tree and the architectural approaches in the case of ATAM.

4.2.3. Participants Selection

The context of this family of experiments is the quality evaluation of software architectures from the perspective of novice architecture evaluators.

[2] An excerpt of the description of the architecture is shown in Appendix A.1.1

Although experienced architecture evaluators enhance the value of the evaluation [22], we focus on the profile of novice evaluators since one of our goals is to provide a software architecture evaluation method that will help less experienced evaluators to perform architecture evaluation, by reusing the domain expert's knowledge. The following groups of participants were therefore identified in order to facilitate the generalization of results:

- Undergraduate students, all Computer Science students at the Universitat Politècnica de València. These students attended the "Advanced Software Engineering" course from September 2012 to January 2013, during which time they had eight hours of lectures on software architectures and architecture evaluation.
- Master's students, enrolled on the Software Engineering Master's degree program at the Universitat Politècnica de València. These students attended the "Quality of Web Information Systems" course from February 2013 to July 2013. One of the main topics on this course is the quality assurance and control and includes more than eight hours of theoretical content concerning software architectures and architecture evaluation.
- Undergraduate students, all Computer Science students at the Universitat Politècnica de València. These students attended the "Software Quality" course from February 2013 to July 2013. One of the main topics on this course is the quality assurance and control and also includes more than eight hours of theoretical content concerning software architectures and architecture evaluation.
- Undergraduate students, all Computer Science students at the Università degli Studi della Basilicata. These students attended the "Software Engineering" course from March 2013 to June 2013. One of the main topics on this course is modeling of object-oriented systems using the UML. The students had experience in object-oriented programming and Web technology.

We have focused on the profile of final-year undergraduate and Master's students since it has been demonstrated that, under certain conditions, there is no great difference between this type of students and professionals [12]; [44], and that they can be considered as the next generation of practitioners [51].

We did not establish a classification of participants based on their architecture evaluation experience, since neither the undergraduates nor the Master's students had a previous background in conducting architectural evaluations.

4.3. Step 3: Experimental Tasks and Material

The experimental tasks were structured to allow the comparison of both methods, starting with the software architecture to be evaluated, a set of NFRs (which are documented in different ways depending on the method), and a set of patterns. We selected a set of NFRs (reliability and performance) which are critical in the automotive domain [17] and in the mobile applications domain [36]. Depending on the method, each task was composed of the method activities that help to achieve its purpose. After applying the method, the participants had to fill in a post-experimental questionnaire with subjective questions regarding the method.

4.3.1. ATAM Experimental Tasks

The experimental tasks carried out by the participants when applying ATAM included two measurement processes, the *analysis of architectural approaches* (both on the investigation and analysis and on the testing phase) and *the scenario prioritization activities* of ATAM (see Section 3.1). The system's software architecture, the business goals, the architectural approaches to be considered and the utility tree of the system were provided as input as a result of the activities performed by the authors since they were part of the problem description (see Section 4.2.2). The experiment consisted of three experimental tasks, which in the case of ATAM were structured as follows:

- The first experimental task consisted of a first measurement of the architecture to check the fulfillment of the NFRs described in the utility tree. This helps the participants to understand whether the architecture meets the NFRs. During this activity, the participants had first to examine the documentation of the metrics to be applied and then to calculate the metrics values by introducing the values required in an excel file that automate the metric calculation.
- The second experimental task consisted of three ATAM activities: i) the first *analysis of architectural approaches*, in which the participants had to analyze how the architectural approaches identified

support the scenarios and attributes on the utility tree; ii) the *prioritization of scenarios* activity, in which the participants had to assign priorities to the utility tree scenarios and; iii) the second *analysis of the architectural approaches*, in which the participants had to select the architectural pattern to be applied.

- Finally the third experimental task consisted of the final measurement of the modified architecture after the application of patterns in order to check the fulfillment of the NFRs described in the utility tree by following the same procedure than in the first measurement process.

4.3.2. QuaDAI Experimental Tasks

The experimental tasks carried out by the participants when applying QuaDAI included two executions of the *Product Architecture Evaluation* activity and the *Product Architecture Transformation* activity (see Section 3.2). The system's software architecture, the system's NFRs and the multimodel with the tradeoff among architectural transformations and quality attributes were provided as input as a result of the activities performed by the authors since, as in the case of the inputs of the ATAM method, it is part of the problem description (see Section 4.2.2). The experiment consisted of three experimental tasks, which in the case of QuaDAI were structured as follows:

- The first experimental task consisted of a first application of the QuaDAI's Product Architecture Evaluation activity, in which the participants performed a first measurement of the architecture to check the fulfillment of the NFRs. During this activity, the participants had first to examine the documentation of the metrics to be applied and then to calculate the metrics values by introducing the values required in an excel file that automate the metric calculation.
- The second experimental task consisted of the QuaDAI's Architectural Transformation activity, in which the participants also had to introduce the relative importance of the NFRs as weights ranging from 0 to 1. These values were introduced by the participant in the excel file that contains the multimodel and that returns which pattern was selected based on that information.
- Finally, the third experimental task consisted of a second application of the QuaDAI's Product Architecture Evaluation activity, in which the participants performed the final measurement of the modified architecture after the application of patterns in order to check the fulfillment of the NFRs by following the same procedure than in the first measurement process.

The experimental tasks only comprised one iteration of the method due to the need of defining a set of experimental tasks that allow us to compare the final results of the process.

4.3.3. Experimental Materials

The experimental material[3] was composed of a set of documents required to support the experimental tasks and the training sessions, along with the post-experimental questionnaire.

The training materials included: i) a set of slides containing the introduction to software architectures, architectural patterns, and software architecture evaluation; ii) a set of slides describing the QuaDAI method, with an example of its application which also introduced the use of the excel files which partially automate the metrics calculation; iii) a set of slides describing the ATAM method, along with an example of its application.

The documents supporting the experimental tasks included:

- Four kinds of booklets which covered the four possible combinations of both evaluation methods and experimental objects (QuaDAI-O1, QuaDAI-O2, ATAM-O1, ATAM-O2). The purpose of these booklets was to i) describe the experimental tasks to be performed; ii) describe the systems, the architecture of each system and the NFRs to be fulfilled; and iii) gather the data from each experimental task. An excerpt of the architectural description contained on the booklet can be found in Appendix A.1.1.
- Two appendixes (O1 and O2) containing the description of the architectural pattern to be applied. The description of each pattern contains its name, the context in which the pattern can be applied, the description of the problem to be solved, the pattern structure and the

[3]All the materials are available for download at http://users.dsic.upv.es/~jagonzalez/IST/family.html

consequences in terms of benefits and drawbacks. Two examples of these patterns can be found in Appendix A.1.3.

- Two appendixes (O1 and O2) containing the description of the software metrics that measure the systems' NFRs (Both for ATAM and for QuaDAI). An example of these metrics can be found in Appendix A.1.4.
- Two appendixes (O1 and O2) containing the description of the architecture after the application of each pattern (Both for ATAM and for QuaDAI). An example of the resulting architecture after the application of a pattern is shown in Appendix A.1.5.
- Two Excel files (O1 and O2) which automate the calculations of the application of the different metrics (Both for ATAM and for QuaDAI). The complexity of the calculations required that the calculations of each metric were partially automated. The participants had to fill in the data required by each metric (which was provided in the booklet) and they then obtained the final result, which they had to evaluate.
- Two appendixes containing a detailed explanation of each evaluation method (QuaDAI and ATAM). The QuaDAI appendix included guideline to help the participants in the definition of the importance of the quality attributes.
- Two Excel files (O1 and O2) which included the selection of alternative architecture transformations based on the quality attributes' relative importance selected by the participant, to be used during the application of the QuaDAI method.

The post-experimental questionnaire contained a set of closed-questions that allowed the participants to express their opinion of the method's ease of use, usefulness and their intention to use that method in the future. The closed questions included in the questionnaire can be found in Appendix A.2. The order of the questions in this questionnaire was shuffled in order to prevent systemic response bias, and the questions were formulated to become negative statements on the left-hand side so as to avoid monotonous responses [45]. We also included two open questions in order to obtain the participants' feedback as regards the changes that they would make to improve the methods and their reasons for using a given method in the future.

4.4. Step 4: Individual Experiments

The family of experiments is summarized in Fig. 6. The original experiment (UPV1) [41] was replicated [42] so as to obtain more evidence for the results obtained in the experiment and to verify the remaining issues. The second experiment (i.e., UPV2) was differentiated internal replication of the original experiment performed in different settings and the third experiment (i.e., UPV3) as a differentiated internal replication of the second experiment (i.e., UPV2).

The fourth experiment (UNIBAS) was an external replication of the third experiment (i.e., UPV3) performed at the Università degli Studi della Basilicata in Italy so as to verify the findings obtained in the three previous experiments and to avoid any author bias that may have been present in the previous studies.

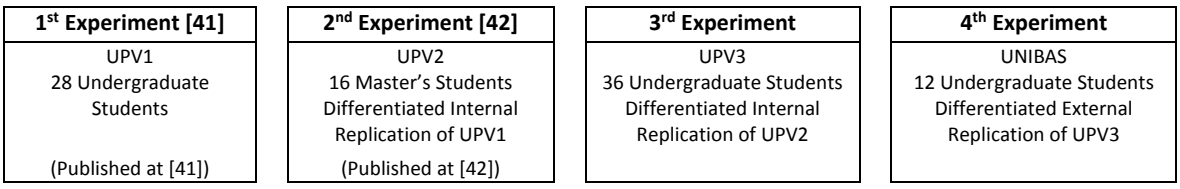

Fig. 6. Overview of our family of experiments

4.5. Step 5: Family Data Analysis and Meta-Analysis

The results of each individual experiment were collected using the booklets and the questionnaire containing the closed-questions, and they were then analyzed. For testing the hypotheses we applied parametric one-tailed t-tests for testing these variables that were normally distributed and the non-

parametric Mann Whitney test when the data was not normally distributed. For testing whether or not the data was normally distributed we applied the Shapiro-Wilk test since in each individual experiment the sample size was less than 50 [24].

We also performed a meta-analysis, based on the Hunter-Schmidt method [46], based on the point biserial correlation *r*, in order to aggregate the results, since the experimental conditions were very similar for each experiment. This analysis, which is detailed in Section 7.2, enabled us to obtain stronger results and to extract more general conclusions with regard to each individual experiment.

5. Design of Individual Experiments

In this section, we describe the characteristics of each experiment in the family of experiments. To avoid redundancies, we only discuss some clarifications of the original experiment with regard to the information presented in the previous section and the differences between the experiments. We conclude the section by discussing issues related to the documentation used in the external replication and the means of communication used by the experimenters.

5.1. The Original Experiment (UPV1)

5.1.1. Planning

Context of the experiment: we used both of the experimental objects (O1 and O2) described in Section 4.2.1 and applied the software architecture evaluation methods described in Section 4.2.2. We selected 31 undergraduate students as participants.

Selection of Variables: The **independent variable** of interest in this family of experiments is the use of each architecture evaluation method with nominal values: ATAM and QuaDAI.

There are two objective dependent variables:

- *Effectiveness* of the method, which was calculated as a function of the *Euclidean Distances* between the NFR values attained by the architecture being evaluated by the participant and the optimal set of values that it was possible to attain when selecting the architectural pattern that best fits the NFRs for each experimental object.
- *Efficiency*, which is calculated as the ratio between the effectiveness and the total time spent on applying the evaluation method.

Effectiveness is calculated by applying formula (1) to normalized *Euclidean* distances, were p is the vector of NFRs' values as obtained by the participant. 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. Formula (3), calculates the distance between two n-dimensional vectors of NFRs values, p and q Normalization is required 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. The values for effectiveness range from 0 (i.e., the minimum possible effectiveness, when the distance *D* is the maximum observed for all the participants) to 1 (i.e., the maximum effectiveness, when the distance *D* is 0).

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

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

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

For example, if the optimal values for a given experiment object O_i are $q = (0.5, 50)$ and a given participant obtains an architecture whose NFRs are $p = (0.7, 51)$ the *Euclidean* distance $D = \sqrt{(0.7 - 0.5)^2 + (51 - 50)^2} = 1.02$. If the maximum distance for this specific experimental object is 1.5; $Norm\big(D(p, Optimal(O_i)\big) = {}^{1.02}/_{1.5} = 0.68$ and the participants effectiveness is calculated as: $Effectiveness(p) = 1 - 0.68 = 0.32$.

There are also three **subjective dependent variables**, which are based on the Technology Acceptance Model (TAM) [27], since TAM is one of the most widely applied theoretical models when analyzing user acceptance and usage behavior of emerging information technologies, and has empirical support through validations and replications [77]. The perceived efficacy [27] of the method can be broken down into the following subjective dependent variables:

- *Perceived Ease of Use***,** which refers to the degree to which evaluators believe that learning and using a particular method will be easy.
- *Perceived Usefulness*, which refers to the degree to which evaluators believe that using a specific method will increase their job performance within an organizational context.
- *Intention to Use*, which refers to the extent to which an evaluator intends to use a particular method. This last 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 13 closed-questions: 3 questions for perceived ease of use (PEOU), 6 questions for perceived usefulness (PU) and 4 for intention to use (ITU). The closed-questions were formulated by using a 5-point Likert scale, using the opposing statement question format. In other words, each question contains two opposite statements representing the maximum and minimum possible values (5 and 1), where the value 3 is considered to be a neutral perception. The aggregated value of each subjective variable was calculated as the arithmetical mean of the answers to the questions associated with each subjective dependent variable.

Hypothesis formulation: We formulated the following null hypotheses, which were formulated in a one-tailed manner, since we wanted to analyze the effect of the use of QuaDAI on the subjective variables.

Each null hypothesis and its alternative are presented as follows:

- **$H1_0$:** 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 evaluators applying 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 intention to use of QuaDAI and ATAM / **$H5_a$:** QuaDAI is perceived as more likely to be used than ATAM.

Experimental design: The experiment was planned as a balanced within-participant design with a confounding effect, signifying that the same participants applied both methods with both experimental objects in a different order. We established four groups (each of which applied one method to one object) and the participants were randomly assigned to each group.

Table 2 shows the experimental design schema used in all the individual experiments. The within participants experimental design is intended to minimize the impact of learning effects on the results, since none of the participants repeat any treatment or experimental object during the execution. Other factors which may also have been present needed to be controlled, since they might have influenced the results, i.e., the complexity of experimental objects. The comprehension of the architecture to be evaluated, the NFRs, the metrics evaluating these NFRs and the architectural patterns may have affected the application of both methods. We attempted to alleviate the influence of this factor by selecting two representative software systems with software architectures, NFRs, software metrics and architectural patterns of a reasonable

complexity. The complexity of the patterns and metrics selected made them suitable for application in the time slot available for the execution of the experiments.

Table 2 Experimental design

	Groups (sample size = 4n participants)			
	G1 (n participants)	*G2 (n participants)*	*G3 (n participants)*	*G4 (n participants)*
1st Session	ATAM applied in O1	ATAM applied in O2	QuaDAI applied in O1	QuaDAI applied in O2
2nd Session	QuaDAI applied in O2	QuaDAI applied in O1	ATAM applied in O2	ATAM applied in O1

Instrumentation: the documents presented in Section 4.3 were used to support the experimental tasks (4 data gathering documents, 4 appendices and 1 questionnaire and 3 Excel files) and the training material (3 slide sets).

5.1.2. Operation and Execution

This section describes the experimental operation, including the preparation, the execution, the data recording and the data validation.

With regard to the operation of the experiment, the experiment was planned to be conducted in three sessions (Table 3 shows the details for each day). On the first day, the participants were given complete training on the methods to be applied and also on the tasks to be performed in the execution of the experiment. On the second and third days, the participants were given an overview of the training before applying one evaluation method to an experimental object (O1 or O2). We established a slot of 90 minutes with no time limit for any of the methods to be applied. However, we allowed the participants to continue the experiment even though these 90 minutes had passed in order to avoid a possible ceiling effect [71].

With regard to the experiment execution, the experiment took place in a single room, and no interaction between participants was allowed. The questions that arose during the session were clarified by those conducting the experiment.

With regard to the data validation, we verified that one of the participants had not completed the 2nd session and that it was therefore necessary to eliminate this data point. Since we had 30 participants distributed in four groups, it was necessary to discard two participants (who did not represent outliers, they were simply selected randomly) in order to maintain the balanced design, shown in Table 2 (i.e., having exactly the same number of participants in each group), consisting of a total of 28 participants, with seven samples in each group.

Table 3 Schedule of the first experiment

1st session (120 min)	Training on Software Architecture Evaluation using ATAM and QuaDAI			
2nd session (120 min)	Software Architecture Evaluation using ATAM and QuaDAI (short training)			
	QuaDAI in O1	QuaDAI in O2	ATAM in O1	ATAM in O2
	QuaDAI Questionnaire		ATAM Questionnaire	
3rd session (120 min)	Software Architecture Evaluation using ATAM and QuaDAI (short training)			
	ATAM in O2	ATAM in O1	QuaDAI in O2	QuaDAI in O1
	ATAM Questionnaire		QuaDAI Questionnaire	

5.2. The Second Experiment (UPV2)

This experiment (first replication) was different to the original experiment in three aspects:

- Participant Selection: The participants were 19 Master students. They attended the "Quality of Web Information Systems" course, and whose profile is described in Section 4.2.3.
- One level of an NFR in the experimental object O2 was also changed since in this experimental object it was easier to find the best solution (100% of the participants had selected the best pattern when dealing with O2 in the original study) as compared to the experimental object O1 (only 71% of the participants had selected the best pattern, regardless of the method).
- Control questions: We included a set of control questions[4] in the experimental material in order to analyze the comprehension of the patterns and the metrics being applied. These questions

[4] The control questions are included in the booklet (which is available at http://users.dsic.upv.es/~jagonzalez/IST/family.html) and should be answered by the participants after

helped the participants to focus on understanding the patterns and metrics and allowed us to control their comprehension of the problem. These questions did not influence the experiment execution and results; their purpose was solely to control the comprehension of the patterns and metrics.

With regard to the preparation of the experiment, the experiment was also planned to be conducted by following the schedule shown in Table 3. As in the original experiment, the experiment took place in a single room and no interaction between participants was allowed. With regard to the data validation, in order to maintain a balanced design, it was necessary to discard the data from three participants (who did not represent outliers, they were simply selected randomly), consisting of a total of 16 participants - 4 samples in each group.

5.3. The Third Experiment (UPV3)

This third experiment (second replication) was a replication of UPV2. The difference between this experiment and UPV2 was the participants selected. The participants were 40 undergraduate students. They attended the "Software Quality" course, and whose profile is described in Section 4.2.3.

The preparation and execution of the experiment were the same as those for UPV2 since the same three day planning was followed. With regard to the data validation, we verified that three participants had not completed the 2nd session and that it was therefore necessary to eliminate their first exercise. Since we had 37 participants distributed in four groups, it was necessary to discard one additional participant (who did not represent an outlier, and was simply selected randomly) so as to maintain the same number of samples per group, consisting of a total of 36 participants, with nine samples in each group.

5.4. The Fourth Experiment (UNIBAS)

The fourth experiment (third replication in our family of experiments) is an external replication of UPV3. UNIBAS was different from UPV3 in three respects:

- Participants Selection: The participants were 12 third-year Computer Science undergraduate students. They attended a "Software Engineering" course.
- Experimental Tasks: we also included a third NFR and a new software metric to be applied in order to make the tradeoff more complex and less obvious. This also included a new NFR in the calculation of the effectiveness, following the expressions described in Section 5.1.1.
- Experimental Material: The material was translated from Spanish into English, which was the language in which the replication was conducted. English was not the mother tongue of the participants. This may have led to construct and external validity threats because of the participants' familiarity with the English language. However, all of the participants had to pass an English language exam to be enrolled in the second year of their Bachelor program, and the participants' knowledge of English was therefore almost homogenous, thus mitigating the possible threats mentioned above. For the first time, the participants used the electronic version of the material rather than printed versions of the documents.

The preparation and execution of the experiment were slightly different to those of the other experiments. A few days before the experiment sessions, the participants attended three training sessions (180 minutes in total). In the first session (60 minutes), software architecture evaluation concepts were presented, while ATAM and QuaDAI were introduced in the latter two training sessions, respectively. In order to better explain the software architecture evaluation approaches that are the object of our study, two running examples were presented to the participants in these latter two sessions, which lasted about 60 minutes each. It is worth mentioning that the experimental schema used was the same as that used in the other experiments (see Table 3).

5.5. Documentation and Communication

Issues such as documentation [73] and communication between experimenters [76] may influence the success of a replication. Deficiencies in documentation and laboratory packages are one of the biggest

reading the pattern description for the questions regarding the patterns and before the first measurement for the questions regarding the metrics.

sources of problems, and make it difficult to use replication to advance knowledge. As a possible solution, the authors propose better laboratory packages and the use of knowledge sharing mechanisms.

With regard to the documentation, the experimenters in the three original experiments translated all the material initially written in Spanish into English. This material included the post-experimental questionnaire, all the annexes documenting the patterns, the metrics, the software architectures resulting from the application of patterns, and the excel spreadsheets automatizing the application of each metric. We also included a set of slides in the training material so as to explain the execution procedure.

In the document, we also discussed the rationale behind the design choices made in the original experiment, highlighting all the information that was useful to reproduce the experimental conditions. The experimenter involved in the external replications was also provided with previous publications concerning the original experiment [41], [42]. The groups of experimenters additionally exchanged the training material in order to reproduce the same experimental setting as used in UPV1, UPV2 and UPV3. Although documentation is a key factor in being able to carry out a replication, communication among experimenters is even more important [76].

The interactions between the groups of experimenters were mainly by e-mail, and instant messaging tools were also occasionally used. The exchange of documentation was performed by using file sharing tools in the cloud.

6. Analysis of the Results

In this section, we discuss the results of each individual experiment by quantitatively analyzing the data according to the hypotheses stated. All the results presented were obtained by using the SPSS v20.

In this analysis, we used descriptive statistics, boxplots and statistical tests in order to analyze the data collected from each individual experiment. In particular, since the sample size was less than 50, it was necessary to apply the Shapiro-Wilk test to check whether the data was normally distributed. The Shapiro-Wilk test allowed us to select the tests to be applied in order to check the hypotheses. When the data was normally distributed (Shapiro-Wilk p-value ≥ 0.05) we applied the parametric one tailed t-test for independent samples [48]. However, when the data could not be assumed to be normally distributed, we applied the non-parametric Mann-Whitney test [24]. The subjective variables were analyzed separately for each method, by comparing whether the mean of the responses to the questions related to a variable were significantly greater than the Likert neutral value. In our case, the ordinal scales ranged from 0 to 5 and the neutral value corresponded to 3.

We applied these tests because they are a set of robust, sensitive and accepted statistical tests that are widely applied in the Software Engineering community [55]. As is usual, in all the tests we decided to accept a probability of 5% of committing a Type-I-Error [78], i.e., rejecting the null hypothesis when it is actually true.**¡Error! No se encuentra el origen de la referencia.**Table 4 shows a summary of the overall results of the architecture evaluations performed in each individual experiment. Mean and standard deviations have also been used as descriptive statistics for the qualitative subjective variables PEOU, PU and ITU. Even though we did not further analyze *Duration* we included the variable in this summary so as to give a first idea of the complexity of the experimental tasks. In the UNIBAS experiment, duration was higher since we included a third NFR that make the process less obvious and probably due to the participant's former level of experience.

Table 4 Summary of the family results

		UPV1 (N=56)		UPV2 (N=32)		UPV3 (N=72)		UNIBAS (N=24)	
		Mean	SD	Mean	SD	Mean	SD	Mean	SD
Effectiveness	QuaDAI	**0.684**	0.361	**0.797**	0.209	**0.708**	0.351	**0.631**	0.277
	ATAM	0.626	0.388	0.461	0.448	0.462	0.404	0.297	0.285
Efficiency	QuaDAI	**0.029**	0.013	**0.023**	0.007	**0.025**	0.015	**0.011**	0.006
	ATAM	0.019	0.018	0.014	0.015	0.015	0.014	0.006	0.007
Duration	QuaDAI	**25.36**	7.258	**34.63**	7.974	**30.11**	8.671	**69.83**	23.486
	ATAM	31.11	9.154	39.19	11.356	33.03	9.032	73.17	29.735
Perceived Ease of Use	QuaDAI	**3.98**	0.876	**4.312**	0.714	**3.59**	0.659	**2.667**	1.054
	ATAM	3.50	0.816	4.020	0.714	**3.59**	0.658	2.444	0.729
Perceived Usefulness	QuaDAI	**3.804**	0.832	**4.167**	0.860	**4.07**	0.498	**3.569**	0.329
	ATAM	3.723	0.730	3.927	0.672	3.85	0.528	3.458	0.342
Intention to Use	QuaDAI	**3.654**	0.698	**4.063**	0.710	**3.935**	0.631	**3.458**	0.673
	ATAM	3.547	0.844	3.906	0.831	3.740	0.638	3.417	0.685

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 [19]. The cells highlighted in bold type in Table 4 show the best

values for each of the statistics. The overall results have allowed us to interpret that the participants achieved their best performance with QuaDAI in almost all the statistics under analysis.

The following subsections detail the analysis of each dependent variable (Effectiveness, Efficiency, Perceived Ease of Use, Perceived Usefulness and Intention to Use) and the hypotheses testing.

6.1. Effectiveness

Fig. 7 shows the boxplots for the effectiveness variable per participant and per method for each experiment in our family. These boxplots show that QuaDAI was relatively more effective than ATAM when evaluating the architectures in each experimental object. Probably this can be explained by the reuse of the knowledge stored in the multimodel. It was easier for the participants to interpret the NFRs and prioritize the quality attributes so as to obtain the correct pattern to be applied rather than the subjective evaluation of the patterns with regard to the scenarios on the utility tree.

We checked the statistical significance of these tests by performing the Mann-Whitney non-parametric test so as to verify H1 in UPV1, UPV2 and UPV3, since the Shapiro-Wilk test results evidenced that they were not normally distributed (p-value=0.000 both for QuaDAI and ATAM in UPV1, p-value=0.000 for QuaDAI and p-value=0.001 for ATAM in UPV2 and finally, p-value=0.000 both for QuaDAI and ATAM in UPV3). The 1-tailed t-test was then used for the verification of H1 in UNIBAS since it was normally distributed. The p-values obtained for these tests were 0.906 for UPV1, 0.036 for UPV2, 0.003 for UPV3 and 0.008 for UNIBAS. These results therefore support the rejection of the null hypothesis $H1_0$ in experiments UPV2, UPV3 and UNIBAS (p-value < 0.05) and the acceptance of its alternative hypothesis, signifying that the effectiveness of the participants when applying QuaDAI was significantly greater than their effectiveness when applying ATAM.

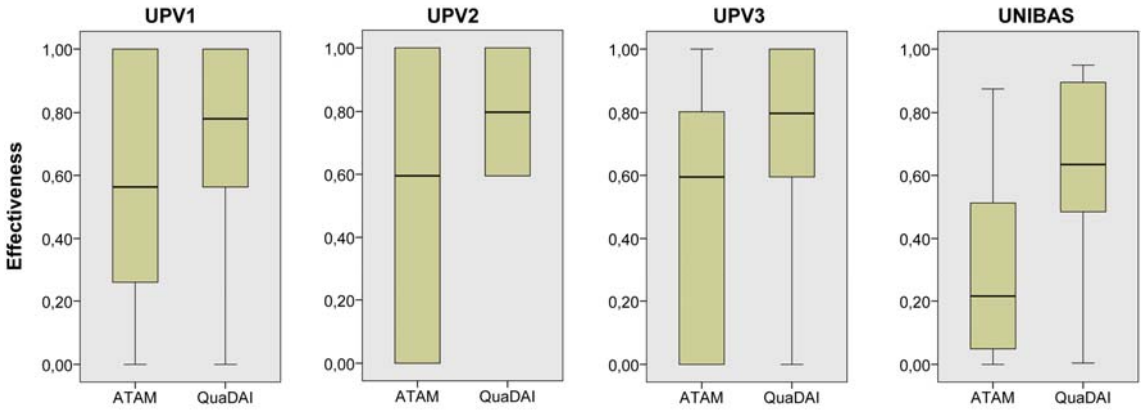

Fig. 7. Effectiveness variable boxplots

6.2. Efficiency

Fig. 8 shows the boxplots for the efficiency variable per participant and per method for each experiment[5]. These boxplots show that QuaDAI was relatively more efficient than ATAM when evaluating the architectures in each experimental object. This can be due to the effectiveness of the participants (the *efficiency* variable was calculated as *effectiveness/duration*) but also because they performed faster, since the selection of the architectural pattern was automated once they established the priority of the quality attributes.

We checked the statistical significance of these results by performing the Mann-Whitney non-parametric test so as to verify H2 in UPV2, UPV3 and UNIBAS, since the Shapiro-Wilk test evidenced that they were not normally distributed (p-value=0.016 for ATAM in UPV2, p-value=0.000 for ATAM in UPV3 and finally, p-value=0.001for ATAM in UNIBAS). The 1-tailed t-test was then used for the verification of H2 in UPV1 since it was normally distributed. The p-values obtained for these tests were 0.030 for UPV1, 0.043 for UPV2, 0.06 for UPV3 and finally 0.045 for UNIBAS. These results therefore support the rejection of the null hypothesis $H2_0$ in all the experiments (p-value < 0.05) and the acceptance of its alternative hypothesis, signifying that the efficiency of the participants when applying QuaDAI was significantly greater than their efficiency when applying ATAM.

[5] We did not provide specific treatment to the outliers; they are only evident for the efficiency variable since those two participants finished the experimental tasks in less time than the rest.

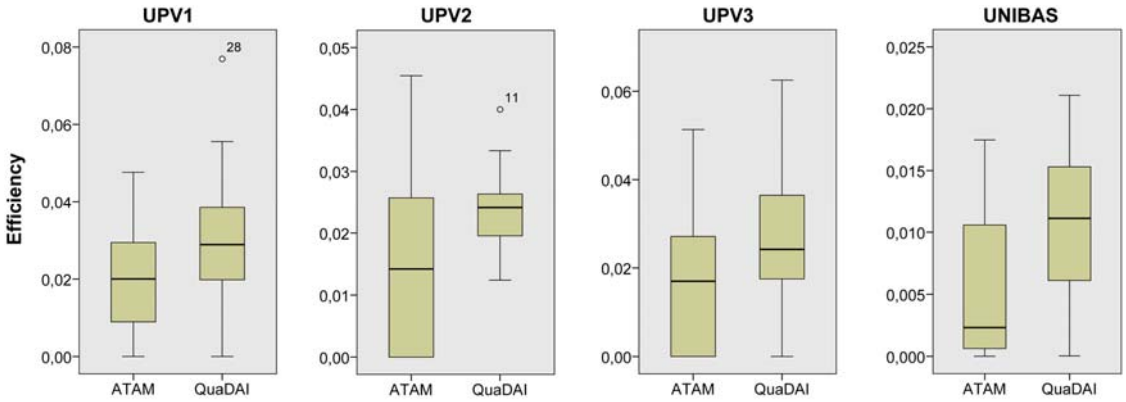

Fig. 8. Efficiency variable boxplots

6.3. Perceived Ease of Use

Fig. 9 shows the density-plots for PEOU per method for each experiment. The mean of each population is shown with two vertical lines in each density-plot. These density-plots show that, with the exception of UPV3 in which the participants perceived both methods to be equally easy to use, the participants perceived QuaDAI to be easier to use than ATAM. This can be owing to the fact that the participants perceived as easier to decide the priority of the quality attributes once they had finished the measurement process rather than evaluating the different patterns and how these patterns support the scenarios. This was confirmed by the participants in their response to the open questions of the questionnaire: *"I find a simple and intuitive way to evaluate. The use of the spreadsheet facilitates the process"*, "*It is a method easy to learn and apply"*.

We checked the statistical significance of these results by performing the one sample Wilcoxon test with a test value equal to 3 for each method separately so as to verify H3 for UPV1/QuaDAI, UPV1/ATAM, UPV2/QuaDAI and UPV3/ATAM, since the Shapiro-Wilk test evidenced that they were not normally distributed (p-value=0.014 for UPV1/QuaDAI, p-value=0.027 for UPV1/ATAM, p-value=0.026 for UPV2/QuaDAI and finally p-value=0.046 for UPV3/ATAM). The two-tailed t-test with a test value equal to 3 (the Likert scale's neutral value) was then used to verify H3 for UPV2/ATAM, UPV3/QuaDAI, UNIBAS/QuaDAI and finally for UNIBAS/ATAM, since they were normally distributed. The p-values obtained for these tests were 0.000 for UPV1/QuaDAI, 0.003 for UPV1/ATAM, 0.001 for UPV2/QuaDAI, 0.000 for UPV2/ATAM, 0.000 for UPV3/QuaDAI, 0.000 for UPV3/ATAM, 0.297 for UNIBAS/QuaDAI and 0.023 for UNIBAS/ATAM. These results therefore support the rejection of the null hypothesis $H3_0$ in UPV1, UPV2, UPV3 ($p\text{-value} < 0.05$) and the acceptance of its alternative hypothesis, signifying that the participants perceived QuaDAI to be easier to use than ATAM.

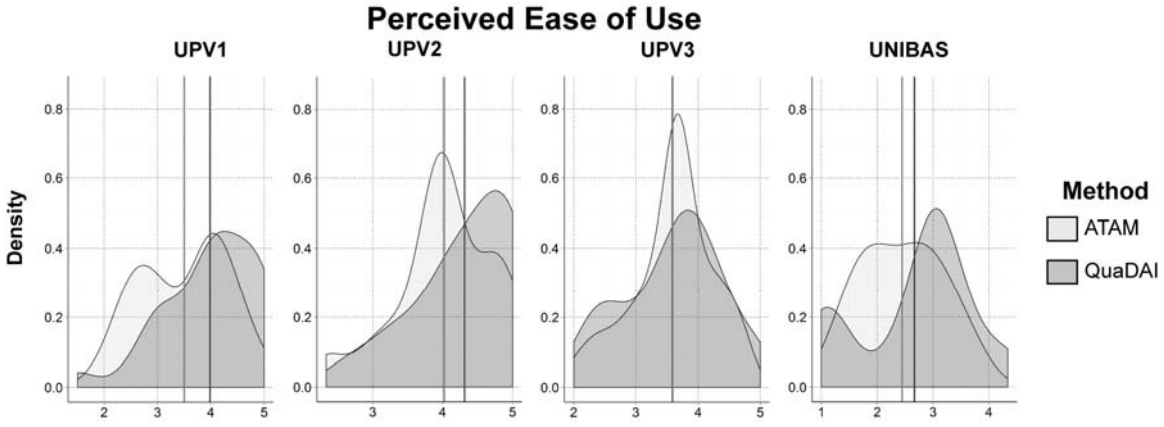

Fig. 9. Perceived ease of use variable density-plots

6.4. Perceived Usefulness

Fig. 10 shows the density-plots for PU per method for each experiment. The mean of each population is shown with two vertical lines in each density-plot. These density-plots show that the participants perceived QuaDAI to be more useful than ATAM. This can be owing to the fact that the participants perceived they perceived as more useful the use of a tool (the spreadsheet) that helps them to take the decision. Since they are novice architecture evaluators, it seems more useful for them to use a method that allows reusing the domain expert's knowledge.

We checked the statistical significance of these results by performing the one sample Wilcoxon test with a test value equal to 3 for each method separately so as to verify H4 for UPV1/QuaDAI, UPV3/QuaDAI, and UNIBAS/QuaDAI, since the Shapiro-Wilk test evidenced that they were not normally distributed (p-value=0.001 for UPV1/QuaDAI, p-value=0.008 for UPV3/QuaDAI and finally p-value=0.005 for UNIBAS/QuaDAI). The two-tailed t-test with a test value equal to 3 (the Likert scale's neutral value) was then used to verify H4 for UPV1/ATAM, UPV2/QuaDAI, UPV2/ATAM, UPV3/ATAM, and UNIBAS/ATAM, since they were normally distributed. The p-values obtained for these tests were 0.000 for UPV1/QuaDAI, 0.000 for UPV1/ATAM, 0.001 for UPV2/QuaDAI, 0.001 for UPV2/ATAM, 0.001 for UPV3/QuaDAI, 0.000 for UPV3/ATAM, 0.002 for UNIBAS/QuaDAI and 0.001 for UNIBAS/ATAM. These results therefore support the rejection of the null hypothesis $H4_0$ in each experiment (p-value < 0.05) and the acceptance of its alternative hypothesis, signifying that the participants perceived QuaDAI to be more useful than ATAM.

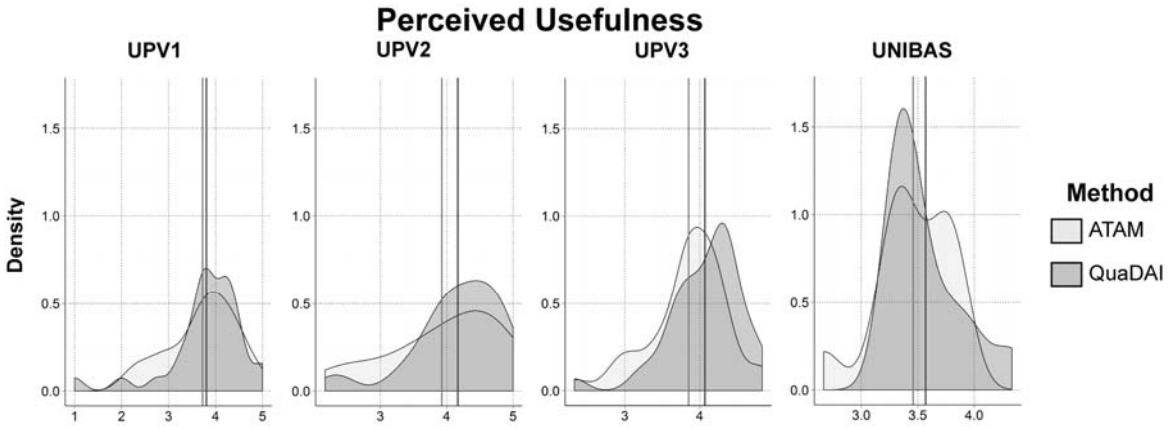

Fig. 10. Perceived usefulness variable density-plots

6.5. Intention to Use

Fig. 11 shows the density-plots for the ITU per method for each experiment. The mean of each population is shown with two vertical lines in each density plot. These density-plots show that the participants perceived QuaDAI to be more likely to be used than ATAM. This can be owing to the fact that since they are novice architecture evaluators, they perceive that the method provides them mechanisms with which to carry out software evaluations in a model-driven context. This was confirmed by the participants in their response to the open questions of the questionnaire: *"I have the intention of using this method for the evaluation of architectures since incorporates the mechanisms to detect the improvements to be made in software architectures"*, "*I will use it since allows prioritizing different characteristics and take the decision based on this prioritization"*.

We checked the statistical significance of these tests by performing the one sample Wilcoxon test with a test value equal to 3 for each method separately so as to verify H5 for UPV1/QuaDAI, UPV2/QuaDAI, UPV3/QuaDAI and UPV3/ATAM, since the Shapiro-Wilk test evidenced that they were not normally distributed (p-value=0.024 for UPV1/QuaDAI, p-value=0.022 for UPV2/QuaDAI, p-value=0.000 for UPV3/QuaDAI and p-value=0.005 for UPV3/ATAM). The two-tailed t-test with a test value equal to 3 (the Likert scale's neutral value) was then used to verify H5 for UPV1/ATAM, UPV2/ATAM, UNIBAS/QuaDAI and UNIBAS/ATAM, since they were normally distributed. The p-values obtained for these tests were 0.000 for UPV1/QuaDAI, 0.000 for UPV/ATAM, 0.001 for UPV2/QuaDAI, 0.001 for UPV2/ATAM, 0.000 for UPV3/QuaDAI, 0.000 for UPV3/ATAM, 0.003 for UNIBAS/QuaDAI, 0.059 for UNIBAS/ATAM. These results

therefore support the rejection of the null hypothesis $H5_0$ in UPV1, UPV2, UPV3 (p-value < 0.05) and the acceptance of its alternative hypothesis, signifying that the participants perceived QuaDAI to be more likely to be used than ATAM.

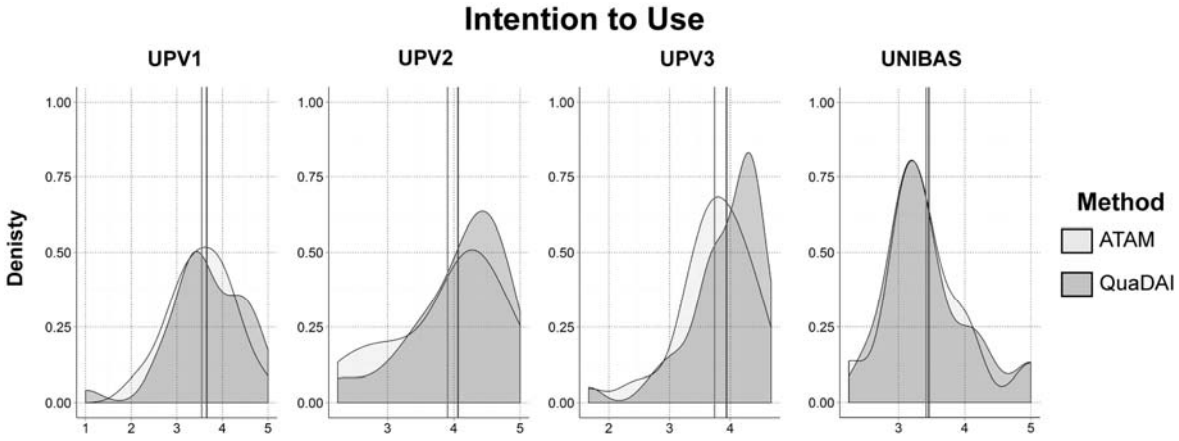

Fig. 11. Intention to use variable density-plots

7. Consolidated Data Analysis

This section provides a summary of the results obtained. We present an analysis of the results in the context of our family of experiments, followed by the results of a meta-analysis that aggregates the empirical findings obtained in the individual experiments.

7.1. Summary of Results

Once the experiments had been carried out, we performed a global analysis of the results in order to determine whether the main goal of our family of experiments had been attained.

The main result is the empirical validation of a method in the field of software architecture evaluation. In this family of experiments we have gained empirical evidence on how a method may help novice architects in performing software architecture evaluations as opposed to a method widely used in industrial environments. This evidence is a contribution to the body of knowledge on architecture evaluation methods since it provides factual data about which software architecture evaluation method is more suitable under certain conditions.

These results may be of interest to the software architecture community in general and to novice architects in particular (since we have tested its utility for guiding novice architects on performing architectural evaluations) but also for researchers in the area who want to replicate the experiments (the research package has been made public). Finally, the validation strategy could be relevant (and could be reused) by other researchers to validate other existing architecture evaluation methods. As far as we know, this is the first family of experiments on comparing architecture evaluation methods.

We also analyzed all the results of the individual experiments to search for differences. A summary of the results obtained in each individual experiment is provided in Table 5.

Table 5 Summary of the results of the family of experiments

Experiment	Type of participants	Num. of participants	Hypotheses rejected
UPV1	Undergraduate students	28	$H2_0$, $H3_0$, $H4_0$, $H5_0$
UPV2	Master Students	16	$H1_0$, $H2_0$, $H3_0$, $H4_0$, $H5_0$
UPV3	Undergraduate students	36	$H1_0$, $H2_0$, $H3_0$, $H4_0$, $H5_0$
UNIBAS	Undergraduate students	12	$H1_0$, $H2_0$, $H4_0$

The individual results show that in all the experiments the participants attained their best results with the QuaDAI and that all the alternative hypotheses were accepted in at least two experiments. In the context of this family of experiments, QuaDAI proved to be more effective and efficient than ATAM. The participants

were also more satisfied when they applied QuaDAI, and they found it easier to use, more useful and more likely to be used than ATAM in a Model-Driven Software Product Line Development context.

With regard to the Effectiveness variable (see Table 4 and Fig. 7), we detected that after the application of the QuaDAI evaluation and transformation activities, the participants obtained software architectures that were closer to the required NFR values that should have been attained. However, in the UPV1 experiment the results were not found to be statistically significant. This may be owing to the fact that in the first experiment the experimental object O2 was much easier than in experiment O1, and this may have affected this result (there were differences only in the experimental object O1, and these were not sufficient to lead to the statistical significance). What is more, the differences in the UNIBAS experiment were even more important; this could have been owing to the addition of a third NFR, making the decision even more complicated.

With regard to Efficiency (see Table 4 and Fig. 8), we detected that the participants were also faster when they applied QuaDAI than when using ATAM. This may be owing to the fact that when applying ATAM the participants had to consider all the architectural approaches (i.e., patterns) and how they affect the NFRs of interest, whereas when they applied QuaDAI they had to decide on the relative importance of the different NFRs, and this decision was made automatically based on that relative importance.

With regard to the Perceived Ease of Use variable (see Table 4 and Fig. 9), we found that QuaDAI achieved mean values higher than ATAM. In general, in all the experiment the mean values for PEOU were above the Likert neutral value established at 3 points (except in the UNIBAS experiment, where was found to be non-statistically significant). In the UNIBAS experiment, Perceived Ease of Use was found to be non-statistically significant perhaps due to the fact that in this experiment the materials were not in the participants' mother tongue –Italian-, which may have influenced the subjective variables owing to the complexity of the questionnaire questions. We also observed that there was a slight difference between the Master's students (who perceived both methods to be easier to use) as compared to the undergraduate students. This could have been owing to the former level of experience and maturity of the participants, who have a better appreciation of the difficulties involved in the software architecture evaluation process.

With regard to the Perceived Usefulness variable (see Table 4 and Fig. 10), we found that the QuaDAI achieved mean values higher than ATAM, signifying that the participants perceived QuaDAI to be more useful than ATAM. It is also important to note that both scores are good results for both methods since all of them were above the Likert's neutral value established at 3 points.

With regard to the Intention to Use variable (see Table 4 and Fig. 11 we found that QuaDAI achieved mean values higher than ATAM, signifying that the participants perceived QuaDAI to be more likely to be used than ATAM. It is also important to highlight that both scores are good results for both methods since all of them were above the Likert's neutral value established at 3 points. In the case of the UNIBAS experiment, this variable was not found to be statistically significant.

The variability in all these results and its significance may be due to the number of participants in each experiment and the heterogeneity of the profiles among experiments. In addition the first experiment it also suffered from differences on the complexity of the experimental objects, which only could be spotted once the experiments had been carried out. In the future, and once we have a stable experiment definition, we plan to execute new replication not only with bigger groups but also with more experienced participants, in order to analyze the impact that the profile could have on the results.

In summary, the results support the hypothesis that QuaDAI would perform better than ATAM in the specified context for which had been designed which is the evaluation of product architectures that occur after the product architecture derivation in model-driven SPL development context. According to the previously discussed results, we can conclude that QuaDAI can be considered as a promising approach with which to perform product architecture evaluations in model-driven SPL development processes after the architecture derivation. Feedback on how to improve the approach was also obtained (e.g., the difficulties that some participants had when assigning the relative importance to the set of NFRs or the possible reuse of the measurement results to select from among architectural transformations). Running a family of experiments (including replications) rather than a single experiment provided us with more evidence of the external validity, and thus the generalization of the study results. Each replication provided further evidence of the confirmation of the hypothesis. We can thus conclude that the general goal of the empirical validation has been achieved.

7.2. Meta-Analysis

Although several statistical methods have been proposed for the aggregation and interpretation of the results that were obtained from interrelated experiments (e.g., [38], [46], [58], [69]), we used meta-analysis

since it allowed us to extract more general conclusions. Meta-analysis consists of a set of statistical techniques with which to combine the effect sizes of each experiment so as to obtain a global effect of a factor. The estimation of effect sizes can be used after the comparison of the different studies to evaluate the average impact across studies of an independent variable on the dependent variable. Since measures may originate from different settings and may be non-homogeneous, a standardized measure should be obtained for each experiment: these measures must be combined to estimate the global effect size of a factor. In our study, we considered that the architectural evaluation method was the main factor in the family of experiments.

The meta-analysis was performed by using the Meta53 [69] and applying the Hunter-Schmidt method [46]. Since all the variables under study had at least one experiment in which the data was not normally distributed we used the point biserial correlation *r*. Correlations are the best-known effect sizes, and they describe the direction and strength of the relationships between two variables within a range of -1.0 and 1.0 [69]. The point biserial correlation r for each experiment can be calculated by applying the formula (4) [58], where Z is the quartile (z-score) of the p-value obtained by the Mann-Whitney/Wilcoxon tests and *N* is the sample size in each case. Since we have different population sizes across the studies, the best way in which to estimate the effect size for the family of experiments is to calculate the correlation $\bar{r}$ weighted by the number of participants in each study [46]. This metric assigns more weight to the large *N* studies and is calculated using formula (5) [46]. For studies in the Software Engineering field, the effect size calculated using point-biserial *r* correlation is rated as follows: small (0-0.193), medium (0.194-0.456), or large (above 0.456) [49]. The Hunter-Schmidt Method also allows a chi-square significance test of homogeneity across the studies to be performed. This test is calculated by applying formula (6) [46]. The result of homogeneity test is the chi-square with k-1 degrees of freedom, for a family of k experiments, and where the weighted variance across studies ${S_r}^2$ is calculated by applying formula (7) [46]. If the chi-square is not significant, this means that there is strong evidence that there is no true variation across studies, but if it is significant, then the variation may still be negligible in magnitude [46].

$$r = \sqrt{\frac{Z^2}{N}} \tag{4}$$

$$\bar{r} = \frac{\sum r_i * N_i}{\sum N_i} \tag{5}$$

$${\chi_{k-1}}^2 = \frac{\sum N_i}{(1-\bar{r}^2)^2} * {S_r}^2 \tag{6}$$

$${S_r}^2 = \frac{\sum (N_i (r_i - \bar{r})^2)}{\sum N_i} \tag{7}$$

Table 6 summarizes the results of the meta-analysis. The overall effect of each variable is reported with the weighted mean $\bar{r}$, its significance, and the homogeneity of the studies with regard to that effect. In each case, the significance of this effect size was strongly significant, and the population was found to be homogeneous for the Effectiveness and Efficiency variables and for the Perceived Usefulness of QuaDAI, and heterogeneous for the remaining variables (i.e., Perceived Ease of Use, ATAM's Perceived Usefulness and for Intention to Use). The effect size obtained was medium for Effectiveness and Efficiency, and large for the subjective dependent variables (i.e., Perceived Ease of Use, Perceived Usefulness and Intention to Use). This was probably as a result of the number of experiments used in the data meta-analysis.

Table 6 Effect sizes and homogeneity test results

Variable	*N*	*Significance*	*Test of Homogeneity*

			Global Effect Size (Weighted Mean $\bar{r}$)		Chi-square	Degrees of freedom	Result (p-value)
Effectiveness		184	Medium (0.27197)	Yes (p<0.001)	6,7648	3	Homogeneous (p=0,080)
Efficiency		184	Medium (0.32346**)**	Yes (p<0.001)	0,4693	3	Homogeneous (p=0,926)
PEOU	***QuaDAI***	92	Large (0.70681)	Yes (p<0.001)	47,0646	3	Heterogeneous (p<0.001)
	ATAM	92	Large (0.55076)	Yes (p<0.001)	51,4611	3	Heterogeneous (p<0.001)
PU	***QuaDAI***	92	Large (1.00000)	Yes (p<0.001)	3,5851	3	Homogeneous (p=0,310)
	ATAM	92	Large (0.98694)	Yes (p<0.001)	27,6145	3	Heterogeneous (p<0.001)
ITU	***QuaDAI***	92	Large (0.93938)	Yes (p<0.001)	36,8937	3	Heterogeneous (p<0.001)
	ATAM	92	Large (0.87296)	Yes (p<0.001)	17,4899	3	Heterogeneous (p<0.001)

The results of the cluster analysis, in which the effect size of each individual experiment is also analyzed, are shown in Table 7. For each experiment, this table reports the population effect size through the unweighted mean r after applying Mullen and Rosenthal's cluster analysis method [58]. This effect has also been classified as Small (S), Medium (M) and Large (L) following the same classification [49] for effect sizes as those applied to the weighted effect size in Table 6.

Despite the fact that the first experiment contributed to the overall results of the meta-analysis to a lesser extent, these results have a significant positive effect, and we can thus reject the null hypotheses which were formulated for each dependent variable (i.e., "there are no significant differences between QuaDAI and ATAM"). The meta-analysis therefore strengthens all the alternative hypotheses, providing promising results as regards QuaDAI's performance.

Table 7 Individual experiments effect sizes

			PEOU		PU		ITU	
Experiment	***Effectiveness***	***Efficiency***	***QuaDAI***	***ATAM***	***QuaDAI***	***ATAM***	***QuaDAI***	***ATAM***
UPV1	S (0.0146)	M (0.2684)	L (0.9622)	M (0.4374)	L (1.0000)	L (0.8079)	L (0.8880)	L (0.7684)
UPV2	M (0.3678)	M (0.3585)	L (1.0000)	L (1.0000)	L (1.0000)	L (0.8011)	L (1.0000)	L (0.8273)
UPV3	M (0.3502)	M (0.3230)	L (0.6584)	L (0.8495)	L (1.0000)	L (1.0000)	L (1.0000)	L (1.0000)
UNBAS	L (0.5098)	M (0.4069)	S (-0.1348)	S (-0.6799)	L (1.0000)	L (1.0000)	L (0.7965)	L (0.7967)

8. Threats to the Validity

In this section, we explain the main issues that may have threatened the validity of the experiments, by considering the four types of threats proposed in [25].

8.1. Internal Validity

The threats to internal validity are relevant in those studies that attempt to establish causal relationship. In our case, the main threats to the internal validity of the family of experiments were: Learning effect, participants' experience, information exchange among participants, author's bias, author influence, order of methods in the training sessions and the understandability of the documents.

The learning effect was alleviated by defining two experimental objects, ensuring that each participant applied each method in a different experimental object and considering all the possible combinations of both the method order and the experimental objects by following a within participants experiment design which is intended to minimize the impact of learning effects.

There were no differences in the participants' experience since none of them had any experience in architecture evaluations. We confirmed this by asking the participants about their previous experience with architecture evaluation methods. This was the rationale behind providing the training sessions on both methods in each experiment since it was our intention to balance the participants' knowledge of architecture evaluation methods according to the novice evaluator profile. The participants were also provided with an introduction to the tasks and the problems they would have to solve via training sessions.

The information exchange was minimized by, on the one hand using different experimental objects, and on the other hand by monitoring the participants while they performed the tasks. However, since the experiment was designed to take place on more than one day, the participants might have been able to exchange information during the time between the sessions. In order to alleviate this situation the participants were asked to return the material at the end of each session.

The author's bias may have influenced the results in the Spanish experiments (i.e., UPV1, UPV2 and UPV3) since the training sessions were conducted by one author of the method. In these experiments, the author's influence was alleviated by not disclosing the authorship of the QuaDAI method to the participants. However, in the UNIBAS experiment this threat was alleviated since both the training session and the experiment were conducted by an external researcher. In addition the authors may also have influenced the results of the studies since the first activities of both methods were carried out by them resulting on the artifacts to be used by the participants on their experimental tasks. However this bias is mitigated since in both cases the set of artifacts were leveled, and, for each experimental object, contained in essence the same information for both methods. Finally, the authors may have also influenced the results through the addition of the measurement processes to the ATAM experimental tasks. These measurement processes help the subjects identifying the whether the architecture meets the NFRs or not. It is probable that the subjects will take more time if they apply ATAM without applying these measurement tasks. We will investigate this issue in future experimentations.

We attempted to alleviate the understandability by writing the experimental materials for the first three experiments (i.e., UPV1, UPV2 and UPV3) in the participants' mother tongue. However, in the UNIBAS experiment the experimental material was in English rather than in the participants' mother tongue (Italian), and this may have influenced the final results of this replication. Finally, we cleared up any misunderstandings about the experimental materials that appeared during the experimental sessions in all the experiments in the family.

8.2. External Validity

This refers to the approximate truth of conclusions involving generalizations within different contexts. The main threat to external validity is the representativeness of the results. The representativeness of the results might be affected by the evaluation design, the participant context selected and the size and complexity of the tasks.

We believe that the results obtained are applicable to SPLs even though the context at the experiments was not specific for SPL development (this study is focused on validating a subset of QUADAI that provide activities for evaluating and improving software architectures that can (or not) be obtained as a result of a derivation activity in a SPL environment). However, the strategy and the mechanisms provided by QuaDAI to express and validate NFRs make this method suitable to model-driven SPL development.

The evaluation design might have had an impact on the generalization of the results owing to the kind of architectural models and quality attributes to be evaluated. We attempted to alleviate this by selecting two architectures with the same size and complexity from two different domains (i.e., the automotive and mobile applications domains). The architectural models representing the architectural views under study (the *Component and Connector* view for O1 and the *Deployment* view for O2) had a similar number of entities (8 vs 6). With regard to the NFRs, we attempted to select two different and representative NFRs (related to reliability and performance) and their associated metrics, that had also the same complexity (for each experimental object one of the metrics was linear and the second was logarithmic/exponential). With regard to the architectural patterns, the domain experts selected four different patterns for each experimental object with the intention of improving the NFRs of interest.

With regard to the participant's experience, the experiments were conducted with students with no previous experience in architectural evaluations, and who received only limited training in the evaluation methods. However, since they were final year students they can be considered as novice users of architectural evaluation methods, and the next generation of practitioners [51]. The results could thus be considered to be representative of novice evaluators. As further work, we intend to conduct more experiments involving participants with different participant profiles (e.g., practitioners or students with a higher level of knowledge and skills in architecture evaluation) and different experimental objects in order to improve the representativeness of our results. In addition, more external replications should be conducted by other experimental conductors to reinforce the results of the UNIBAS external replication.

The size and complexity of the tasks might have also affected the external validity. We decided to use relatively small tasks that would be applied in few representative architectural models since a controlled experiment requires participants to complete the assigned tasks in a limited amount of time. In addition, the

experiments were designed for executing just one iteration of the experimental tasks and thus the results are valid only in this context.

8.3. Construct Validity

The construct validity of the family of experiments may have been influenced by both the measures that were applied during the quantitative analysis and the reliability of the questionnaire. We attempted to alleviate this threat by using measures that are commonly applied in architecture evaluation and optimization techniques. In particular, the Effectiveness was measured using the Euclidean distance, which has commonly been used to measure the goodness of a solution with regard to a set of opposed NFRs with different purposes (e.g., [26], [75]). The subjective variables are based on the Technology Acceptance Method (TAM), a well-known and empirically validated model for the evaluation of information technologies [27].

The reliability of the questionnaire was tested by applying the Cronbach test. Table 8 shows the Cronbach's alphas obtained for each set of the closed-questions intended to measure the three subjective dependent variables (i.e., PEOU, PU, and ITU). Except in UNIBAS, all the values obtained for PU and ITU were higher than the acceptable minimum threshold (α=0.70) [55]. This can be explained by the fact that in UPV1, UPV2 and UPV3 the materials were in Spanish, the participants' mother tongue, whereas in the UNIBAS experiment the materials were in English, and this difference may have had more influence in the case of the questionnaire.

Table 8 Cronbach's alphas for the questionnaires' reliability

Dependent Variable	*UPV1*	*UPV2*	*UPV3*	*UNIBAS*
Perceived Ease of Use	Acceptable (0.824)	Acceptable (0.890)	Acceptable (0.738)	Acceptable (0.748)
Perceived Usefulness	Acceptable (0.870)	Acceptable (0.898)	Acceptable (0.758)	Non-Acceptable (0.011)
Intention to Use	Acceptable (0.831)	Acceptable (0.814)	Acceptable (0.731)	Non-Acceptable (0.666)

8.4. Conclusion Validity

The main threats to the conclusion validity of the family of experiments were the data collection and the validity of the statistical tests applied. With regard to the data collection, we applied the same procedure to each individual experiment when extracting the data, and ensured that each dependent variable was calculated by applying the same formula. With regard to the validity of the statistical tests applied, we alleviated this threat by applying a set of commonly accepted tests employed in the empirical SE community [55].

9. Conclusions and Further Work

The lack of integrated methods to support the derivation, evaluation, and quality improvement of software architectures motivated us, in a previous work, to propose the Quality-Driven Architecture Derivation and Improvement (QuaDAI) [41], a model-driven approach with which to derive, evaluate and eventually transform software architectures in an SPL environment.

In this paper, we have reported the results of a family of experiments aimed at evaluating participants' effectiveness, efficiency, perceived ease of use, perceived usefulness and intention to use with regard to the use of QuaDAI as opposed to a widely-used industrial architectural evaluation method: the Architecture Tradeoff Analysis Method (ATAM).

The results of the quantitative analysis showed that, under the conditions described above, QuaDAI was more effective and efficient than ATAM (although in the first experiment the differences in the participant's effectiveness did not prove to be statically significant, perhaps owing to the difference in difficulty between the experimental objects in that experiment). In addition, with regard to the evaluators' perceptions, the participants perceived QuaDAI to be easier to use, more useful and more likely to be used than ATAM (although again in the fourth experiment the differences in the perceived ease of use and intention to use were not found to be statistically significant, and this was perhaps influenced by the fact that in the last replication the materials were not in the participants' mother tongue, which may have influenced the subjective variables owing to the complexity of the questionnaire questions). The results were supported by a meta-analysis that was performed in order to aggregate the empirical findings from each individual experiment. The results of the meta-analysis allow us to conclude that there is a significant positive effect on all the variables under study associated with the use of QuaDAI as an architecture evaluation method for evaluating the derived product architecture.

From a research perspective, the family of experiments was a valuable means to obtain feedback with which to improve QuaDAI (e.g., improve the mechanism used to assign the relative importance to the set of NFRs or reuse the results from the measurement process to select architectural transformations). To the best of our knowledge, this is the first empirical study to provide evidence of the usefulness of a software architecture evaluation method for a model-driven software development process. In our family of experiments what we did a first validation of the method to obtain some feedback and to analyze ourselves how the participants performed. We agree that students may not be as helpful as working with practitioners. However we were running the initial validation of the method. Now when we have a first version of the method, and when we have a tool for supporting the configuration and derivation stages of the method our plan is to perform realistic evaluations with practitioners

From a practical perspective, we are aware that this study provides only preliminary results on the usefulness of QuaDAI as a software architecture evaluation method to be applied to architectures obtained (or not) as a result of the derivation of product architectures in model-driven SPL development processes. Although the experimental results provided good results as regards the performance of our software architecture evaluation method for model-driven SPL development, these results need to be interpreted with caution since they are only valid within the context established in this family of experiments. It is now necessary to analyze whether the same results will be obtained with more experienced participants and practitioners and with new experimental objects, and with a wider set of pattern and non-functional requirements. It is therefore necessary to carry out more empirical studies in order to test our proposal in other settings. Nevertheless, this study has value as a first study to test the integration of QuaDAI evaluations in model-driven software development processes.

The validation through the family of experiments had required us to find out a method with which to compare QuaDAI, to set up two well-defined experimental objects from two different, representative domains, together with an example to illustrate the training process. The experimental process had also taught us that, even though we ran pilot studies, experimental materials and experimental tasks need to be readjusted to level the difficulty among the experimental objects. We have also experienced that minor curricula differences among groups may also change the experimental results and sometimes is difficult to figure out the reasons that justify the differences on the variables under study.

As future work, we intend to perform more replications to minimize the effect of possible threats to validity for our study. In particular, these replications will consider: more external replications, new kinds of participants such as practitioners from industry with different levels of experience in software architecture evaluations, and finally, software architectures originating from different domains and using different and wider sets of quality attributes, software metrics and architectural patterns.

Acknowledgements

The authors would like to thank all the participants in the experiments for their selfless involvement in this research. This research is supported by the MULTIPLE project (MICINN TIN2009-13838) and the ValI+D program (ACIF/2011/235).

Appendix A. Excerpts from the Experimental Material

This appendix presents excerpts from all the different materials. Appendix A.1 presents an excerpt from the ATAM and QuaDAI O1 booklets. We show the experimental material of one of the experimental objects since having the whole set of materials of one object it will assist in the comprehension of the experimental tasks. The materials from the second experimental object (i.e., O2) are available for download at http://www.dsic.upv.es/~jagonzalez/IST/family.html. Finally, Appendix A.2 presents the post-experimental questionnaire used to measure the subjective variables.

A.1. Examples of Software Architecture, Patterns and Metrics

A.1.1. Software Architecture to be Evaluated

The first software architecture to be evaluated is from an Antilock Braking System (ABS). The goal of the ABS Systems is to control the brake actuators of a car. The system monitors the brake pedal sensor and activates the brake actuator as soon as the driver presses the brake pedal. In addition, so as to prevent wheel slippage when the brake actuator is activated, the system monitors four wheel rotation sensors, one on each wheel. Each sensor sends signals while the wheel is rotating. If the system detects wheel slippage, the brake actuator will be deactivated, and it will be activated again after a short period of time.

Fig. 12 shows the (ABS) system architecture. The input sensors are on the left. We can see the processing and control software components inside the ABS system, and the actuators are on the right. In this case, apart from the sensors and actuators, the driving console has also been considered. The driving console includes the switches used to activate or deactivate the different systems and the signals that indicate to the user that the ABS has been activated.

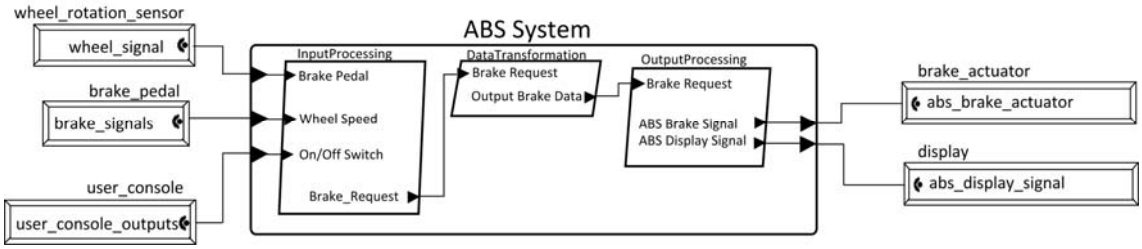

Fig. 12. ABS System architecture

A.1.2. ATAM Utility Tree

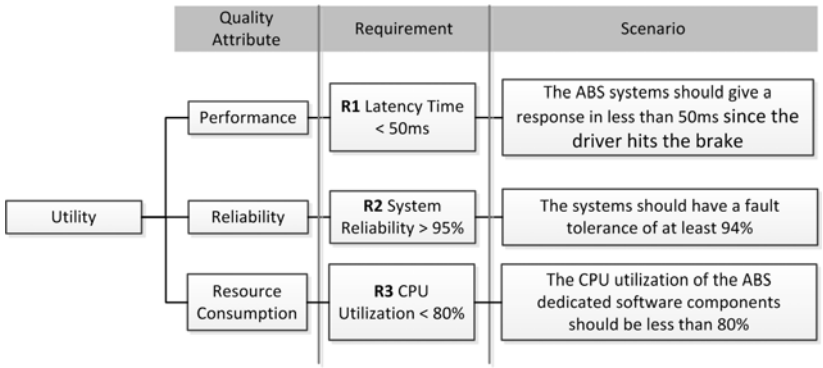

Fig. 13. *ABS System utility tree*

A.1.3. Architectural Patterns

Pattern	**Triple modular redundancy pattern [28]**
Context	The Triple Modular Redundancy Pattern (TMR, for short) is used to enhance reliability and safety in situations in which there is no fail-safe state. The TMR pattern has an odd number of channels (three) which operate in parallel. The computational results or resulting actuation signals are compared, and if there is a disagreement, then a two-out-of-three majority wins policy is invoked.
Problem	The problem addressed by the Triple Modular Redundancy Pattern is the same as that of the Homogeneous Redundancy Pattern—that is, to provide protection against random faults (failures) with the additional constraint that when a fault is detected, the input data should not be lost, nor should additional time be required to provide a correct output response.
Pattern Structure	The pattern has a replicated structure consisting of three channels operating in parallel, as shown in Fig. 14. Each channel contains a set of objects that process incoming data in a series of transformational steps. The channels do not cross-check each other at strategic points. Rather, the three channels in the set operate completely in parallel, and only the final resulting outputs are compared. The comparator implements a winner-take-all policy such that the two channels producing the correct results win.
Consequences	The Triple Modular Redundancy Pattern can only detect random faults. Since the channels are homogeneous, then by definition any systematic fault in one channel must be present in both of the others. Because the channels execute in parallel, the source data is also replicated in each channel. In the case of an error, only the erroneous channel's output is discarded, and the other channels' output is used, so the failure does not result in the loss of data, nor does it require that the output be re-calculated. This pattern adds to the computation time, thus affecting the system's behavior in the general case. TMR has a rather high recurring cost because the hardware and software in the channels must be replicated. The TMR pattern is common in applications in which reliability needs are very high and worth the additional cost involved in replicating the channels.

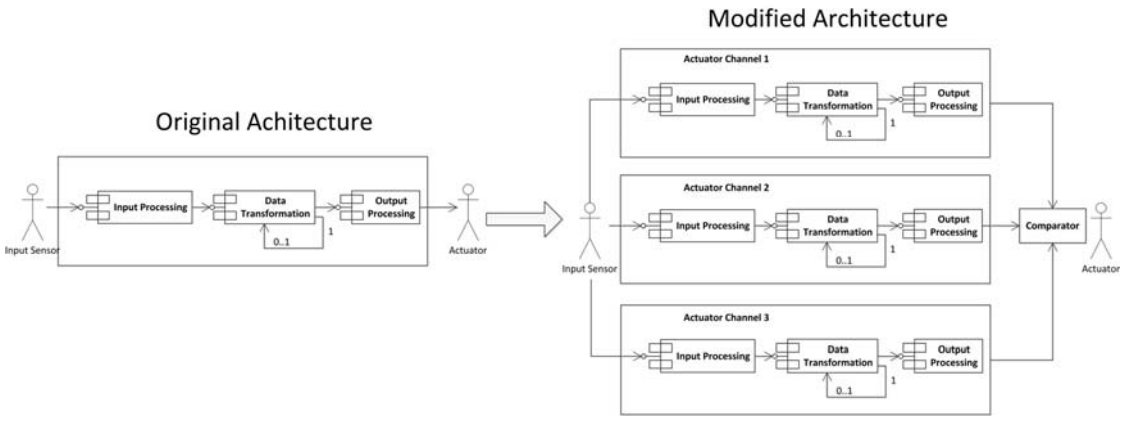

Fig. 14. Triple modular redundancy pattern structure

Pattern	**Watchdog pattern[28]**
Context	The watchdog pattern is a lightweight pattern that provides minimal coverage against faults. The watchdog pattern merely checks that the internal computational processing is proceeding as expected. This means that its coverage is minimal, and a broad set of faults will not be detected.
Problem	Real-time systems are those that are predictably timely. In the most common (albeit simplified) view, the computations have a deadline by which they must be applied. If the computation occurs after that deadline, the result may either be erroneous or irrelevant. If the output appears too late, then the system cannot be controlled; the system is said to be in an unstable region.
Pattern Structure	The structure of the watchdog pattern is shown in Fig. 15, in which we can see its simplicity. The Actuator Channel operates pretty much independently of the watchdog, sending a liveness message to the watchdog every so often. This is called stroking the watchdog. The watchdog uses the timeliness of the stroking to determine whether a fault has occurred. Most watchdogs check only that a stroke occurs in a particular period of time and do not pay attention to what happens if the stroke occurs too quickly. Some watchdogs check that the stroke occurs neither too quickly nor too slowly.
Consequences	The Watchdog Pattern is a very lightweight pattern that is rarely used alone in safety-critical systems. It is best at identifying timebase faults, particularly when an independent timebase drives the Watchdog. It can also be used to detect a deadlock in the actuation channel. Since its coverage it so minimal, its effect on the reliability is weak, and it is therefore rarely used alone.

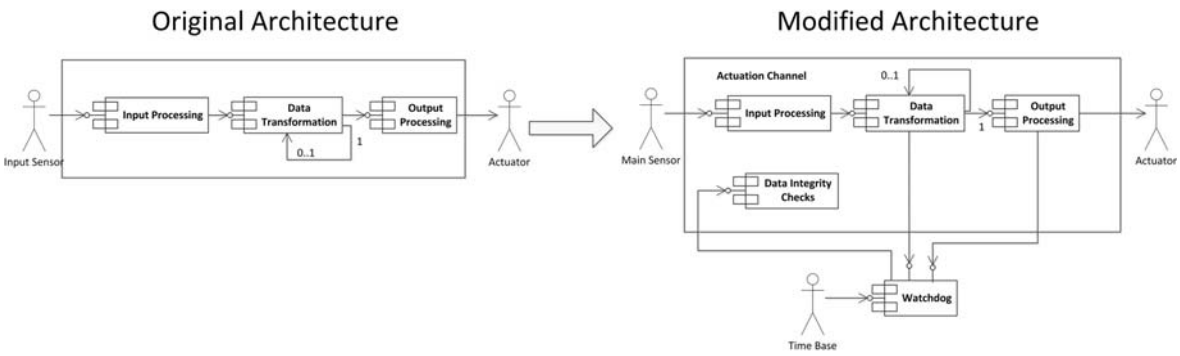

Fig. 15. Watchdog pattern structure

A.1.4. Metrics

Metric	**Latency Time**
Description	The latency time is defined as the time that has elapsed between receiving an input event and the generation of the output by the system.
Calculation	For its calculation we should consider the flows of data and events that follow the reception of an input event in a sensor until the actuator state is changed. We shall consider only the latency time in the best case. $$Latency\ Time = \sum_{Components\ in\ flow} Latency\ Time\ (Component)$$
Interpretation	The latency time gives a positive value as a result. The lower the value is, the better the latency time of the system. It can be measured using different time units (e.g., ms, sec)

A.1.5. Resulting Architecture after the Application of Patterns

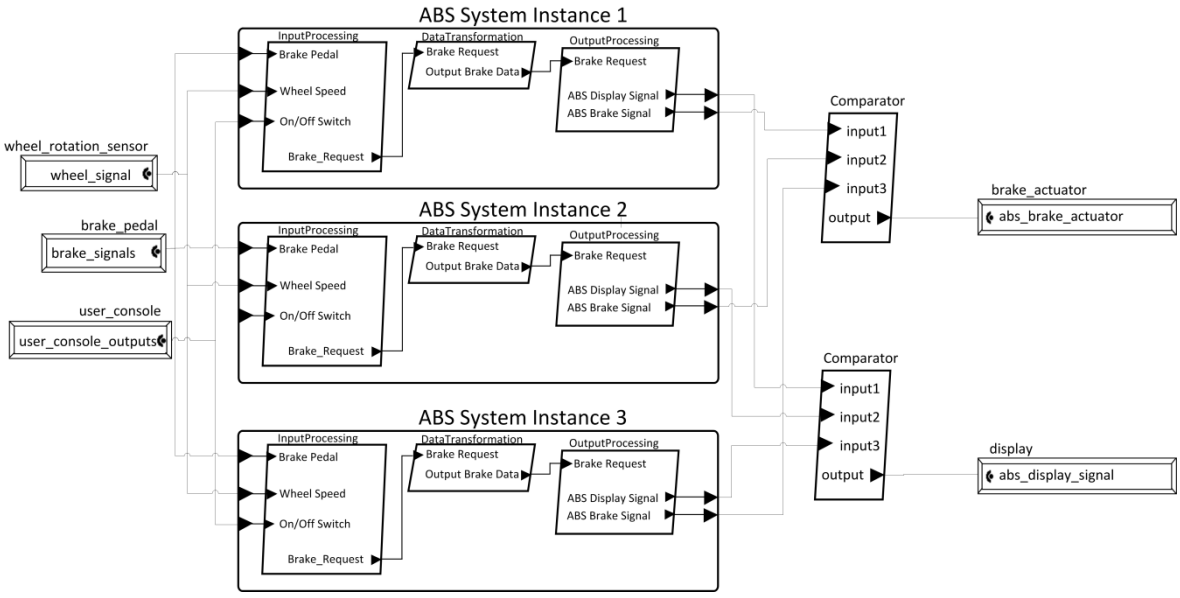

Fig. 16. Resulting architecture after the application of the triple modular redundancy pattern

A.2. **Subjective variables evaluation questionnaire**

Table 9 Closed-questions evaluating each subjective dependent variable

Question	Positive Statement (5 Points)	Negative Statement (1 Point)
PEOU-1	The architecture evaluation method is simple and easy to follow.	The architecture evaluation method is complex and difficult to follow.
PEOU-2	In general, the software architecture method is easy to understand.	In general, the software architecture method is difficult to understand.
PEOU-3	The software architecture method is easy to learn.	The software architecture method is difficult to learn.
PU-1	I believe that this method would reduce the time and effort required for the evaluation of software architecture.	I believe that this method would increase the time and effort required for the evaluation of software architecture.
PU-2	In general, the software architecture method is useful.	In general, the software architecture method is useless.
PU-3	I believe that this software architecture evaluation method is useful as regards obtaining architectures that fulfill the quality requirements.	I believe that this software architecture evaluation method is useless as regards obtaining architectures that fulfill the quality requirements.
PU-4	I believe that this method incorporates the mechanisms required to detect the improvements to be made to software architectures.	I believe that this method lacks the mechanisms required to detect the improvements to be made to software architectures.
PU-5	In general, I believe that this method efficiently supports the software architecture evaluation.	In general, I believe that this method does not efficiently support the software architecture evaluation.
PU-6	The use of this method will improve my performance when evaluating software architectures.	The use of this method will not improve my performance when evaluating software architectures.
ITU-1	If I need to use a software architecture method in the future, I believe that I will consider this method.	If I need to use a software architecture method in the future, I do not believe that I will consider this method.
ITU-2	I believe that it would be easy to become skilled in using this method.	I believe that It would be difficult to become skilled in using this method.
ITU-3	I intend to use this method in the future.	I have NO intention of using this method in the future.
ITU-4	I would not recommend using this software architecture evaluation method.	I would not recommend using this software architecture evaluation method.

References

[1] Abrahão, S., Gravino, C., Insfrán, E., Scanniello, G., Tortora G., 2013. Assessing the Effectiveness of Sequence Diagrams in the Comprehension of Functional Requirements: Results from a Family of Five Experiments. IEEE Transactions on Software Engineering, Vol. 39, Issue 3, pp. 327-342.

[2] Ali-Babar, M., Gordon, I., 2004. Comparison of scenario-based software architecture evaluation methods. In 11th Asia-Pacific Software Engineering Conference, Bussan, Korea, pp. 600-607.

[3] Ali-Babar, M., Zhu, L., Jeffery, R., 2004. A framework for classifying and comparing software architecture evaluation methods. In 15th Australian Software Engineering Conference, Melbourne, Australia, pp. 13-16.

[4] Ali-Babar, M., Kitchenham B., 2007. Assessment of a framework for comparing software architecture analysis methods. In 11th International Conference on Evaluation and Assessment in Software Engineering, Keele, England, pp. 12-20.

[5] Ali-Babar, M., Kitchenham, B., Jeffery, R., 2007. Comparing distributed and face-to-face meetings for software architecture evaluation: A controlled experiment. Empirical Software Engineering, Vol. 13, Issue 1, pp. 39-62.

[6] Ali-Babar, M., Winkler, D., Biffl, S., 2007. Evaluating the usefulness and ease of use of a groupware tool for the software architecture evaluation process. In 1st International Symposium on Empirical Software Engineering and Measurement, Madrid, Spain, pp. 430-439.

[7] Ali-Babar, M., Kitchenham B., 2007. The impact of group size on software architecture evaluation. A controlled experiment. In 1st International symposium on Empirical Software Engineering and Measurement, Madrid, Spain, pp. 420-429.

[8] Ali-Babar, M., 2008. Assessment of a framework for designing and evaluating security sensitive architecture. In 12th International Conference on Evaluation and Assessment in Software Engineering. Bari, Italy.

[9] Ali-Babar, M., Lago, P., Van Deursen, A., 2011. Empirical research in software architecture: opportunities, challenges, and approaches. Empirical Software Engineering. Vol. 16, Issue 5, (October 2011) pp. 539-543.

[10] Barbacci, M., Clements, P., Lattanze, A., Northrop, L., Wood, W., 2003. Using the architecture tradeoff analysis method (ATAM) to evaluate the software architecture for a product line of avionics systems: A case study. Tech. Report CMU/SEI- 2003-TN-012, Software Engineering Institute, Carnegie Mellon University.

[11] Barkmeyer, E.J., Feeney, A.B., Denno, P., Flater, D.W., Libes, D.E., Steves, M.P, Wallace, E.K., 2003. Concepts for automating systems integration, NISTIR 6928, National Institute of Standards and Technology, U.S. Dept. of Commerce.

[12] Basili, V.R., Shull, F., Lanubile, F., 1999. Building knowledge through families of experiments. IEEE Transactions on Software Engineering, Vol. 25, Issue 4, pp. 456–473.

[13] Basili, V.R., Rombach, H.D., 1998. The TAME project: towards improvement-oriented software environments. IEEE Transactions on Software Engineering, Vol. 14, Issue 6, pp. 758–773.

[14] Botterweck, G., O'Brien, L., Thiel, S., 2007. Model-driven derivation of product architectures. In 22th International Conference on Automated Software Engineering, New York, USA, pp. 469-472.

[15] Bosch, J., 2000. Design and Use of Software Architectures. Adopting and Evolving Product-Line Approach. Addison-Wesley, Harlow.

[16] Breivold, H.P., Crnkovic, I., Larsson, M., 2012. A systematic review on software architecture evolution research. Information and Software Technology, Vol. 54, Issue 1 (January 2012), pp. 16-40.

[17] Broy, M., 2006. Challenges in automotive software engineering. 28th international conference on Software engineering New York, USA, pp. 33–42.

[18] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M., 1996. Pattern-oriented software architecture, Vol. 1: A System of Patterns. Wiley.

[19] Carifio, J., Perla, R.J., 2007. Ten common misunderstandings, misconceptions, persistent myths and urban legends about Likert scales and Likert response formats and their antidotes. Journal of Social Sciences, Vol. 3, Issue 3, pp. 106-116.

[20] Ciolkowski, M., Shull, F., Biffl, S., 2002. A family of experiments to investigate the influence of context on the effect of inspection techniques. In 6th International Conference on Empirical Assessment in Software Engineering, pp. 48–60.

[21] Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., Stafford, J., 2011. Documenting software architectures, views and beyond. Addison-Wesley, Boston, USA.

[22] Clements, P., Kazman, R., Klein, M., 2002. Evaluating software architecture, methods and case studies. Addison-Wesley, Boston USA.

[23] Clements, P., Northrop, L., 2007. Software product lines: practices and patterns, Addison-Wesley, Boston.

[24] Conover, W.J., 1998. Practical nonparametric statistics, 3rd Edition. Wiley.

[25] Cook, T., Campbell, D., 1979. Quasi-experimentation: Design & analysis issues for field settings. Houghton Mifflin Company, Boston.

[26] Datorro, J., 2005. Convex optimization & Euclidean distance Geometry. Meboo Publishing.

[27] Davis, F.D., 1989. Perceived usefulness, perceived ease of use and user acceptance of information technology. MIS Quarterly, Vol. 13, Issue 3, pp. 319-340.

[28] Douglass, B.P., 2002. Real-time design patterns: robust scalable architecture for real-time systems. Addison-Wesley, Boston.

[29] Dyba, T, Kitchenham, B., Jorgensen, M., 2005 Evidence-based software engineering for practitioners. IEEE Software Vol. 22, Issue 1, pp. 58–65.

[30] Etxeberria, L., Sagardui, G., 2005. Product line architecture: New issues for evaluation. In 9th International Conference on Software Product Lines, LNCS 3714, pp. 174–185. Springer.

[31] Falessi, D., Ali-Babar, M., Cantone, G., Kruchten, P., 2010. Applying empirical software engineering to software architecture: challenges and lessons learned. Empirical Software engineering, Vol. 15, Issue 3, pp. 250-276.

[32] Falessi, D., Cantone, G., Kruchten, P., 2008. Value-based design decision rationale documentation: principles and empirical feasibility study. In 7th Working IEEE/IFIP Conference on Software Architecture, Vancouver, Canada, pp. 189-198.

[33] Falessi, D., Capilla, R., Cantone, G., 2008. A value-based approach for documenting design decisions rationale: a replicated experiment. In 3rd International Workshop on Sharing and Reusing Architectural Knowledge, Leipzig, Germany, pp. 63-69.

[34] Feiler, P.H., Gluch, D.P., Hudak, J., 2006. The Architecture Analysis & Design Language (AADL): An introduction. Tech. Report CMU/SEI-2006-TN-011. Software Engineering Institute, Carnegie Mellon University.

[35] Fisher, R.A., 1915. Frequency distribution of the values of the correlation coefficient in samples of an indefinitely large population. Biometrika, Vol. 10, Issue 4 (May 1915), 507–521.

[36] Flinn, J., Park, S.Y., Satyanarayanan, M., 2002. Balancing performance, energy, and quality in pervasive computing. 22th International Conference on Distributed Computing Systems, Vienna, Austria, pp. 217 – 226.

[37] Ghezzi, C., Sharifloo, A.M., 2013. Model-based verification of quantitative non-functional properties for software product lines. Information and Software Technology. Vol. 55, Issue 3, pp. 508–524.

[38] Glass, G.V., Mcgaw, B., Smith, M.L., 1981. Meta-analysis in social research. Sage Publications.

[39] Golden, E., John, B.E., Bass, L., 2005. The value of a usability-supporting architectural pattern in software architecture design: A controlled experiment. In 27th International Conference on Software Engineering, St. Louis, Missouri, USA, pp. 460-469.

[40] Gonzalez-Huerta, J., 2014. Derivación Evaluación y Mejora de la Calidad de Arquitecturas Software en el Desarrollo de Líneas de Producto Software Dirigido por Modelos. PhD Thesis, Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València. http://hdl.handle.net/10251/36448.

[41] Gonzalez-Huerta, J., Insfrán, E., Abrahão, S., 2013. Defining and validating a multimodel approach for product architecture derivation and improvement. In 16th International Conference on Model Driven Engineering Languages and Systems, Miami, USA, pp. 388-404.

[42] Gonzalez-Huerta, J., Insfrán, E., Abrahão, S., 2013. On the effectiveness, efficiency and perceived utility of architecture evaluation methods: A replication study. In 18th National Conference in Software Engineering and Databases, Madrid, Spain, pp. 427-440.

[43] Gonzalez-Huerta, J., Insfrán, E., Abrahão, S., McGregor, J.D., 2012. Non-functional requirements in model-driven software product line engineering. In 4th International Workshop on Non-functional System Properties in Domain Specific Modeling Languages, Insbruck, Austria, pp. 63-78.

[44] Höst, M., Regnell, B., Wohlin, C., 2000. Using students as subjects – a comparative study of students and professionals in lead-time impact assessment. In 4th Conference on Empirical Assessment and Evaluation in Software Engineering, pp. 201–214.

[45] Hu, P.J., Chau, P.Y.K., 1999. Examining the technology acceptance model using physician acceptance of telemedicine technology. Management Information Systems, Vol. 16, Issue 2, pp. 91–113.

[46] Hunter, J.E., Schmidt, F.L., Jackson, G.B., 1982. Meta-analysis: cumulating research findings across studies. Sage Publications.

[47] ISO/IEC 25000:2005, 2005. Software Engineering. Software product quality requirements and evaluation SQuaRE.

[48] Juristo, N., Moreno, A.M., 2001. Basics of software engineering experimentation. Kluwer Academic Publishers.

[49] Kampenes, V., Dybå, T., Hannay, J.E., Sjøberg, D.I.K., 2007. A systematic review of effect size in Software Engineering experiments. Information and Software Technology, Vol.49, Issue 11–12 (November 2007), 1073–1086.

[50] Kazman, R., Klein, M., Clements, P., 2000. ATAM: A method for architecture evaluation. Tech. Report CMU/SEI-2000-TR-004, ADA382629, Software Engineering Institute, Carnegie Mellon University.

[51] Kitchenham, B.A., Pfleeger, S.L., Hoaglin, D.C., Rosenber, J., 2002 Preliminary guidelines for empirical research in software engineering. IEEE Transactions on Software Engineering, Vol. 28, Issue 8, pp. 721-734.

[52] Lindsay, R.M., Ehrenberg, A.S.C., 1993. The design of replicated studies. The American Statistician, Vol. 47, pp. 217–228.

[53] Martens, A., Koziolek, H., Prechelt, L., Reussner, R., 2011. From monolithic to component-based performance evaluation of software architectures: A series of experiments analyzing accuracy and effort. Empirical Software Engineering, Vol. 16, Issue 5 (October 2011), pp. 587-622.

[54] Martensson, F., 2006. Software architecture quality evaluation. Approaches in an industrial context. Ph. D. thesis, Blekinge Institute of Technology, Karlskrona, Sweden.

[55] Maxwell, K., 2002. Applied statistics for software managers. Software Quality Institute Series, Prentice Hall.

[56] McGregor, J.D., 2010. Testing a Software Product Line. Revised Lectures of 2nd Pernambuco Summer School on Software Engineering, Pernambuco, Brazil, pp. 104–140.

[57] Microsoft MSDN, 2003. Performance and reliability patterns. Version 1.1.0. http://msdn.microsoft.com/en-us/library/ff648802.aspx (last accessed 07/2013).

[58] Mullen, B., Rosenthal, R., 1985. Basic meta-analysis: procedures and programs. L. Earlbaum Associates.

[59] Niemelä, E., Immonen, A., 2007. Capturing quality requirements of product family architecture. Information and Software Technology, Vol. 49, Issue 11-12 (November 2007), pp. 1107–1120. doi:10.1016/j.infsof.2006.11.003.

[60] Object Management Group 2012. Common Variability Language (CVL) OMG Revised Submission.

[61] Perovich, D., Rossel, P.O., Bastarrica, M.C., 2009. Feature model to product architectures: Applying MDE to Software Product Lines. In IEEE/IFIP & European Conference on Software Architecture, Helsinki, Finland, pp. 201-210.

[62] Prechelt, L., Unger-Lamprecht, B., Philippsen, M., Tichy, W.F., 2002. Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance. IEEE Transactions on Software Engineering, Vol. 28, Issue 6, June 2002, pp. 595-606.

[63] Qureshi, N., Usman, M., Ikram, N., 2013. Evidence in software architecture, a systematic literature review. In 17th International Conference on Evaluation and Assessment in Software Engineering, Porto de Galinhas, Brazil, pp. 97-106.

[64] Reijonen, V., Koskinen, J., Haikala, I., 2010. Experiences from scenario-based architecture evaluations with ATAM. In 4th European Conference on Software Architecture, Copenhagen, Denmark, pp. 214–229.

[65] Rhein, A. von, Apel, S., Kästner, C., 2013. The PLA model: on the combination of product-line analyses. 7th International Workshop on Variability Modelling of Software-intensive Systems, Pisa, Italy

[66] Roos-Frantz, F, Benavides, D., Ruiz-Cortés, A., Heuer, A., Lauenroth, K., 2011. Quality-aware analysis in product line engineering with the orthogonal variability model. Software Quality Journal, Vol. 20, Issue 3-4, pp. 519-565. DOI: 10.1007/s11219-011-9156-5.

[67] Roy, B., Graham, T.C.N., 2008. Methods for evaluating software architecture: A survey. School of Computing, Technical Report 2008-545, Queen's University http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.211.8188 (last accessed 05/2014).

[68] Ryu, D., Lee, D., Baik, J., 2012. Designing an Architecture of SNS Platform by Applying a Product Line Engineering Approach. In 11th International Conference on Computer and Information Science, Shanghai, China, pp. 559–564.

[69] Schwarzer, R., 1987. Meta-analysis programs. http://userpage.fu-berlin.de/~health/meta_e.htm (last accessed 09/2013).

[70] Siegmund, N., Rosenmuller, M., Kästner, C., Giarrusso, P.G., Apel, S., Kolesnikov, S.S., 2011. Scalable prediction of non-functional properties in software product lines. 15th International Software Product Line Conference, Munich, Germany, pp. 160–169.

[71] Sjøberg, D.I.K., Anda, B., Arisholm, E., Dybå, T., Jørgensen, M., Karahasanovic, A., Vokác, M., 2003. Challenges and recommendations when increasing the realism of controlled software engineering experiments. Empirical Methods and Studies in Software Engineering Experiences from ESERNET 2001–2003, LNCS 2765, pp. 24–38.

[72] Sjøberg, D.I.K., Hannay, J.E., Hansen, O., Kampenes, V.B., Karahasanovic, A., Liborg, N., Rekdal, A.C., 2005. A survey of controlled experiments in Software Engineering. IEEE Transactions on Software Engineering, Vol. 31, Issue 9, 733–753.

[73] Shull, F., Mendoncça, M.G., Basili, V., Carver, J., Maldonado, J.C., Fabbri, S., Travasos, G.H., Ferreira, M.C., 2004. Knowledge-sharing issues in experimental software engineering. Empirical Software Engineering, Vol. 9 (January-February 2004), pp. 111-137. doi:10.1023/B:EMSE.0000013516.80487.33.

[74] Svahnberg, M., Martensson, F., 2007. Six years of evaluating software architectures in student projects. Journal of Systems and Software, Vol. 80, Issue 11, pp. 1893-1901.

[75] Taher, L., Khatib, H.E., Basha., R., 2005. A framework and QoS matchmaking algorithm for dynamic web services selection. In 2nd International Conference on Innovations in Information Technology, Dubai, UAE, pp. 1-10.

[76] Vegas, S., Juristo, N., Moreno, A., Solari, M., & Letelier, P., 2006. Analysis of the influence of communication between researchers on experiment replication. In Proceedings of the 2006 ACM/IEEE International symposium on empirical software engineering, New York, New York, USA, pp. 28-37. doi:10.1145/1159733.1159741.

[77] Venkatesh, V., 2000. Determinants of perceived ease of use: integrating control intrinsic motivations, and emotion into the Technology Acceptance Model. Information Systems Research, Vol. 11, Issue 4, pp. 342–365.

[78] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Andwesslen, A., 2000. Experimentation in software engineering – An Introduction. Kluwer.