The Value of Design Rationale Information

D. Falessi, *Member, IEEE*, L.C. Briand, *Fellow, IEEE*, G. Cantone, R. Capilla, Member, IEEE, P. Kruchten *Senior Member, IEEE*

Abstract- A complete and detailed Design Rationale Documentation (DRD) could support many software development activities, such as an impact analysis or a major redesign. However, a full DRD is too onerous for systematic industrial use as it is not cost-effective to write, maintain, or read. The key idea investigated in this paper is that DRD should be adopted only to the extent required to support activities particularly difficult to enact or in need of significant improvement in a particular context. The aim of this paper is to empirically investigate the possibility to customize the DRD by documenting only the information that will probably be required for executing an activity. This customization strategy relies on the hypothesis that the value of a DRD information item depends on its category (e.g. Assumptions, Related requirements, etc.) and on the activity it is meant to support. This hypothesis is investigated through two controlled experiments involving a total of seventy-five post-graduate master students as experimental subjects. The value of DRD information was shown to significantly depend on its category and, within the same category, on the activity it supports. Furthermore, our results show that, on average among activities, documenting only the information that has been required at least half of the time (i.e., the information that will probably be required in the future) generates a customized DRD containing about half the information of a full documentation. Such a significant reduction of information to document is expected to mitigate the effects of inhibitors that are currently preventing practitioners from documenting the rationale of their design decisions.

Index Terms— Empirical software engineering, software architecture, design decisions, value-based software engineering, software maintenance.

1. Introduction

During any software development process, most architectural design decisions are not explicitly documented with their rationale, as they are often embedded in the models the architects build [1]. Consequently, useful knowledge associated to the decision-making activities is lost forever [2,3]. In cases where the design erodes, the problem of knowledge vaporization [4], often due to a lack of Design Rationale Documentation (DRD), leads to high maintenance cost, as new design decisions cannot rely on previous ones.

Although the use of design rationale is recognized as one of the most promising steps for advancing the state of the art of software architecture design and maintenance [4], its widespread industrial use has been hindered by socio-technical inhibitors, in particular the effort to produce and maintain additional documentation [5]. According to Gorton [6], "Generating architecture documentation is nearly always a good idea. The trick is to spend just enough effort to produce only documentation that will be useful for the project's various stakeholders. This takes some upfront planning and thinking." In other words, DRD should be tailored to support activities that are the most in need of improvement in a given context.

The term *information item* refers to a single piece of rationale information regarding a design decision. The taxonomy of DRD information proposed by Tyree and Akerman [1] provides a *categorization* of information items for a design decision.

The key idea we wanted to investigate is that design rational documentation should only be introduced to the extent required to support subsequent activities that are particularly challenging to perform, or in significant need of improvement in a particular context. The purpose of this paper is to report on an empirical study of the possibility to customize the design rational documentation by restricting the documentation to only the information elements that are highly likely to be required to perform a subsequent activity.

Our research questions are:

- **R.Q. 1**: Is the value of an information item significantly affected by its category and the activity it supports?
- **R.Q. 2**: How much effort could be saved by adopting a value-based DRD?

The empirical procedure consists of two controlled experiments performed in two different geographical locations and both involving trained, graduate students. The current paper combines the data of the first experiment with that of its replica in order to gain in statistical power and be able to apply more sophisticated analysis techniques. More specifically, we make use of Multiple Correspondence Analysis [7], a technique dedicated to large contingency tables and their interpretation.

The remainder of this paper is structured as follows: Section 2 presents the related work and introduces the concepts used in this study. Section 3 discusses the costs and benefits of using design rationale and presents our key idea. Section 4 describes two experiments (a controlled experiment and an exact replica) with the goal to assess empirically the feasibility and efficiency of documenting only the information that is valuable for the intended use or purpose. Section 5 reports the empirical results and their discussion. The paper concludes in Section 6.

2. Related Work

2.1 DRD Approaches and Tools

There are many definitions of design rationale [8]; one of the most comprehensive definitions has been proposed by Jintae Lee: "Design rationales include not only the reasons behind a

- *D. Falessi is with the Simula Research Laboratory, and University of Rome – Tor Vergata, DISP, PO Box 134, 1235 Lysaker, Norway. E-mail: d.falessi@ieee.org*
- *L. C. Briand is with the Simula Research Laboratory, and the University of Oslo, PO Box 134, 1235 Lysaker, Norway. E-mail: briand@simula.no.*
- *G. Cantone is with University of Rome – Tor Vergata, DISP, Viale del Politecnico 1, 00133 Rome, Italy. E-mail: cantone@uniroma2.it.*
- *R. Capilla is with the Rey Juan Carlos University, DCS, Madrid, Spain. E-mail: rafael.capilla@urjc.es*
- *P. Kruchten is with the University of British Columbia, ECE, Vancouver, Canada. E-mail: pbk@ece.ubc.ca*

design decision but also the justification for it, the other alternatives considered, the tradeoffs evaluated, and the argumentation that led to the decision."[5] The rationale can be classified into several types; most of the times these types are not mutually exclusive. Burge and Brown in [9] propose the following types of rationale: argumentation, history, device, process, and active document. In particular, in the argumentation-based DRD, design rationale is used to represent the arguments that characterize a design, such as, issues raised, alternative responses to these issues, and arguments for and against each alternative. Prominent argument-based design rationale techniques are gIBIS [10], DRL [5], and QOC [11].

Lee states that much of the design rationale is embedded in design specifications or part of meeting discussions [5]. Addressing all the issues that match the different dimensions of design rationale is also difficult because different stakeholders are interested in different concerns. This interest in different concerns is similar in principle to our value-based DRD approach which is based on the premise that "what you represent depends on what you want to do with it" [5].

Tang et al. evaluated the importance of DRD as perceived by industrial software architects [12]. Our paper shares with Tang et al. the concept that "practitioners recognize the importance of documenting design rationale […] however they have indicated barriers to the use and documentation of design rationale" [12]. In particular, they investigated which type of information is generally more likely to be used by practitioners. We have tried to go a step further by investigating the level of support, provided by each category of information, for specific activities.

Referring to the knowledge capturing problem, Tyree and Akerman proposed a framework to document design decision rationale for system architectures [1]. In the present study, we use such a documentation template as an example of DRD and we investigate the level of usefulness of each category of information item in this template in support of different software development activities.

Capilla et al. described an approach for modeling and documenting the evolution of architectural design decisions that is characterized by sets of mandatory and optional attributes that can be tailored according to different users' needs as well as to different organizations [13]. Customized information is used to adapt part of the information captured for the design decisions to the specific needs of different stakeholders and organizations.

Kruchten et al. had suggested a set of different activities supported by DRD [14]; we investigate some of these (e.g., impact analysis) in the present paper.

Van der Ven's et al. describe that different types of stakeholders adopt DRD to enact specific activities; in particular they presented a use-case model that arose from industrial needs [15].

Farenhorst et al. recently investigated by means of a large-scale survey the behavior of architects in terms of their daily activities, and how important they consider the various types of support for sharing architectural knowledge [16]. Their results indicate that architects mainly consume architectural knowledge, but neglect to document and share such knowledge themselves. Such a result calls for supporting architectural knowledge sharing by effectively balancing the costs and benefits of design rationale information.

Jansen et al. presented the Architectural Design Decision Recovery Approach (ADDRA) for recovering architectural design decisions after the fact [17]. In particular, ADDRA uses architectural deltas to provide the architect with clues about these design decisions. This DRD approach has the advantage of requiring little effort from the DRD producer.

Lee and Kruchten divided the documentation activity into three steps: flagging information (identification of possible significant information related to a decision that is being made), filtering (excluding some of the information selected in the previous step), and forming (merging the information produced in the previous step to create a useful and comprehensive DRD) [18]. Their key idea is that only the flagging step needs to be enacted near the decision-making process; so most of the effort required by the DRD can be postponed according to the decision-maker availability.

2.2 Value-based Approaches

We consider all the above-mentioned DRD approaches as value-neutral because they do not relate to business context; rather they aim to maximize the benefits for the knowledge consumer, by imposing the burden on the knowledge producer to document all potential useful information. To date, "much of current software engineering practice and research is done in a value-neutral setting, in which every requirement, use case, object, test case, and defect is equally important" [19]. Consequently, "a resulting value-based software engineering agenda has emerged, with the objective of integrating value considerations into the full range of existing and emerging software engineering principles and practices, and of developing an overall framework in which they compatibly reinforce each other" [19]. In the present work, we apply value-based software engineering principles to documentation, and we propose a value-based approach to DRD consisting in prioritizing the information to document according to the activity to support, i.e., the activities particularly hard to enact without DRD.

The idea of applying a value-based approach to requirements traces has been proposed in several studies including [20,21,22,23]. We share with such past studies the aim and vision though the object being tailored is different (requirements trace vs. DRD). Moreover, "How much traceability is enough?" [24] and "When and How does Requirements Traceability Deliver More Than it Costs?" [25], were addressed in panel discussions held at two important international conferences: COMPSAC06 and RE06.

More than twenty years ago, Basili and Rombach provided practical guidelines for a successful reuse strategy [26]. Our work matches their vision in which the model capturing the experience (i.e., DRD) should be easily tailored to specific project characteristics.

The present work perfectly matches the agile modeling principles [27] and in particular "The TAGRI Principle of Software Development: They Ain't Gonna Read It" as a way to cut the documentation effort [28]. While Ambler poses some relevant questions regarding the different ways stakeholders use the documentation [29], in this paper we provide quantitative results to such questions. Moreover, we agree on the view that the goodness of documentation depends from the situation.

2.3 DRD Benefits

The role of design rationale in software engineering has been extensively discussed by Burge et al. [30]. DRD can support several software architecture related activities: architectural review, review for a specific concern, change impact analysis, studying the chronology, adding decisions, cleaning up the system, spotting the subversive stakeholder, cloning architectural knowledge, detection and interpretation of patterns [14]. To explain the meaning of such DRD uses, let us consider, for instance, the activity 'Impact analysis', which is concerned with the management of requirement changes. In practice, changes in requirements or business goals are very frequent. Martin Fowler [31] emphasized the unpredictability of requirements: "What might be a good set of requirements now, is not a good set in six months time. Even if the customers can fix their requirements, the business world isn't going to stop for them." In the case of a requirements change, it is crucial to understand which decisions (and eventually which system artifacts) are still valid and which ones have to be re-worked (i.e., redesigned and/or re-implemented).

Several researchers assessed the support provided by DRD on given activities. Karsenty assessed the QOC approach in the maintenance of a nine-month old software project [32].

Brathall et al. presented a controlled experiment to evaluate the importance of DRD when predicting change impact on software architecture evolution [33]. Results show that DRD clearly improves effectiveness and efficiency.

Zimmermann et al. presented a proactive approach to modeling and reusing architectural knowledge for enterprise application development [34]. Their approach has already shown to be practical for BPM requirement models and the SOA architectural style: they observed initial effort savings and quality improvements on an early adoption project.

Falessi et al. analyzed the value of DRD with respect to effectiveness and efficiency of individual/team decision making in the presence of requirement changes [35]. The main goal was to estimate whether the availability of the DRD would improve the correctness of design decisions. The study was a controlled experiment in which fifty Master students were the experiment subjects in a controlled environment. Figure 1 summarizes the results of the study; we can see how in case of requirement changes the correctness of the decisions improves when the DRD is available for decision makers, both for individual participants and teams. Once assessed that DRD is beneficial, the present step is to provide realistic means to reduce the inhibitors for DRD usage.

3. Balancing the Costs and Benefits of Design Rationale Information

3.1 DRD Inhibitors

Although several studies empirically demonstrated that the use of design rationale documentation brings numerous benefits [32,33,35], such type of documentation is not widely adopted in practice, as discussed above. Let us highlight the main inhibitors to the adoption of DRD:

- **Bad timing and delayed benefit.** The period in which design decisions are made is often critical for the overall project success. People involved in design decisions are usually busy trying to perform, as well as possible, other more recognized and essential tasks and to meet their related deadlines. In such circumstances, documenting rationale is perceived to be less important and is eventually dismissed. Our experience shows that when we suggest documenting the design decision rationale in an appropriate time frame, the most common answer is: "We are already under pressure to meet the deadline, investing additional time in documentation would make the situation worse".

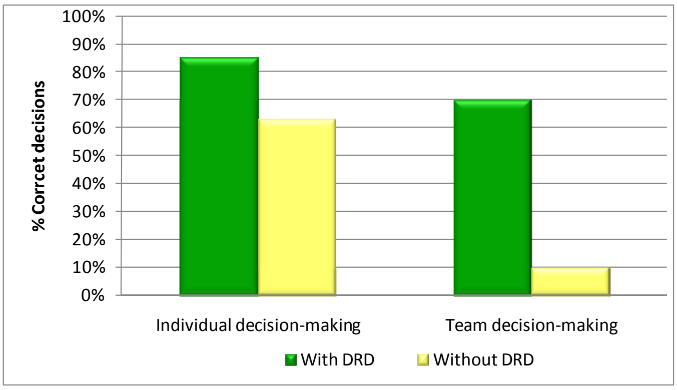


Figure 1: Correctness of individual decisions and team decisions with and without Design Rationale Documentation.

- **Information predictability.** DRD consumers and producers are often different persons. People who are responsible for evolving a software project are usually not the original designers, who meanwhile have moved to better, greener pastures. Hence, the documentation producer needs to forecast which information the consumers will need in the future. As a result, the producer documents all the information that could be useful.
- **Overhead.** Several DRD techniques already exist [9,10,17,18]; however, they usually focus on maximizing the consumer benefits rather than minimizing the producer effort. Consequently, some people must spend substantial effort on documentation and maintenance activities. Supposedly, the overhead required to capture design rationale information can be regained by assisting future maintenance and evolution activities. This, however, is only realistic if we refine the type of information to be captured in order to minimize the associated effort. Shum and Hammond in [36] pointed out that without a good Return On Investment (ROI), the documentation management system would not be used or ultimately it would be counterproductive.
- **Unclear benefits.** Decision-makers often do not know how the DRD will support specific activities.
- **Lack of motivation**. This may arise from the absence of direct benefits or a lack of personal interest. People in charge of documenting and maintaining DRD artifacts (the decision makers) are not very motivated because they do not directly benefit from DRD. Lee in [5] had already highlighted the existing problem that DRD producers and consumers differ. Moreover, experts may not be interested in making their valuable knowledge explicit as they may

perceive to be their personal property. In other words, some experts may see no clear advantage in documenting design rationale.

- **Lack of maturity.** Only few tools are currently available to support DRD and the majority of them are still immature [9,10,17,18].
- **Potential inconsistencies.** DRD and designs should be kept up to date and aligned to avoid potential inconsistencies when the design is modified or when decisions change.

3.2 Key Idea: a Value-Based Customization

To overcome some of the inhibitors previously mentioned, we present a customization strategy for DRD. We classify the different types of information items describing design rationale according to its value; i.e., the level of support to a particular software engineering activity. Such classification, shown in Figure 2, is defined as follows:

- **Useless**: The information does not provide any support to the activity.
- **Optional**: The information facilitates the activity but is not required.
- **Required**: The information is required for performing the activity.

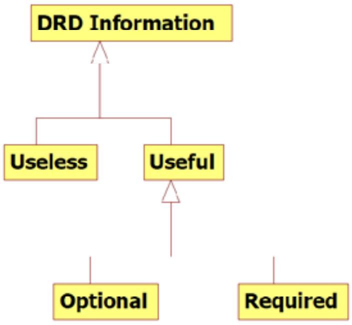


Figure 2: The different values of information.

In the past, researchers tried to maximize the DRD consumer's benefit by forcing the producer to document all the information potentially useful for all activities. The DRD is about making the decision-making process reusable. In the software reuse area, it is agreed that focusing the investment on the most valuable asset to reuse is a key factor for a successful reuse strategy [37,38,39]. Therefore, it is unrealistic to target all possible elements for reuse, i.e., documenting all the information items of a DRD. While in past studies we have evaluated the cost and benefits of DRD [35,40], in the present study we empirically evaluate the feasibility of a tradeoff between its costs and benefits. The key idea is to compromise between the cost to the producer and the benefit to the consumer by achieving a value-based customization (see Figure 3) which consists in documenting only the information valuable for the intended use or purpose, i.e., the information that (will probably) support the readers in a specific activity. We do not imagine a tailored DRD to be a panacea; on the contrary we expect to obtain both advantages and disadvantages from it. The main disadvantage is that the activities to be supported have to be known when architects make design decisions. Even thought this is an important issue, given the presence of several inhibitors, it is reasonable to consider DRD as an economic investment; thus, DRD needs to be motivated by the presence of a real business case. Therefore, DRD should be introduced to support activities that are particularly difficult to enact with the usual procedures and documentation. Moreover, even if a tailored DRD targets specific activities, it is likely to partially support other activities as well.

Though potentially interesting, as there has been no study investigating the value of DRD information for the different phases of the software development lifecycle or for the different types of decisions, we defer such investigation to future work.

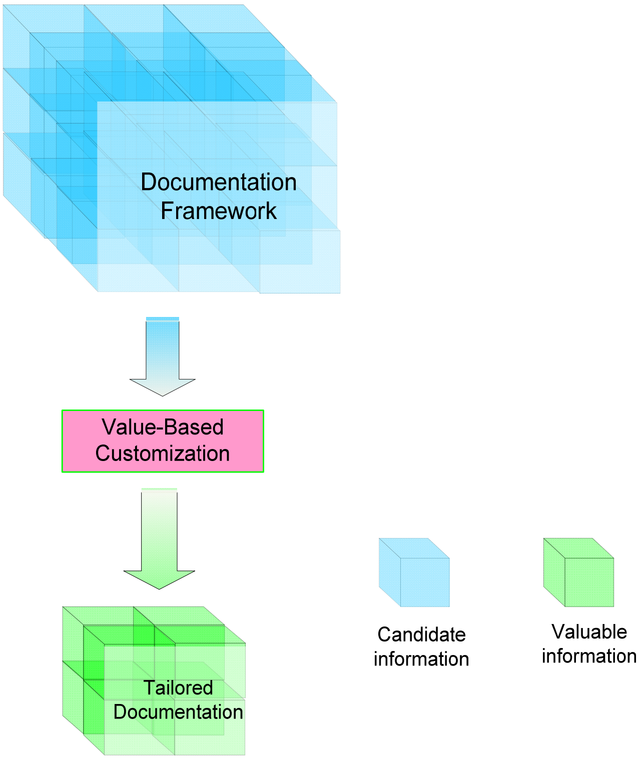


Figure 3: value-based customization.

4. Experiment Planning

4.1 Goals

Our research questions are:

R.Q. 1*: Is the value of an information item significantly affected by its category and the activity that it supports?*

The empirical evaluation of the practical support of a method to architectural activities is challenging [24, 28]. Hence, our strategy is to measure the practical support of a design rationale information item as the frequency with which it is required for enacting a given activity. A value-based customization of design rationale documentation relies on the following hypothesis: the value of an information item is affected by its category and the activity it aims to support. If that is true, architects can reduce the number of information items to document by selecting the ones expected to support a given activity of interest.

R.Q. 2: *How much effort can be saved by adopting a value-based DRD?*

Once assessed the feasibility of a value-based approach, then it is important to assess its efficiency in terms of saved documentation effort. In fact, reducing the effort required to document design rationale is essential to overcome the DRD inhibitors. In this study we measured the effort required by a DRD as the number of information items to document.

Therefore, the reduced effort is computed as the difference between the number of information items of a full DRD versus a customized DRD.

4.2 Research methodology

The research methodology consists of a controlled experiment and its exact replica.

In general, one classical method for identifying cause-effect relationships is to conduct controlled experiments where only independent variables vary [56] and other factors are either controlled or their effects mitigated. Our decision to adopt an experiment stems from the many uncontrolled, confounding factors that could blur the results in an industrial context. Thus, in order to assess the feasibility and potential benefits of a value-based DRD we enacted a controlled experiment involving master-students in computer and electrical engineering at the University of Rome Tor Vergata (Italy).

In general, the purpose of replication is to acquire additional observations, possibly under different conditions, that may lead to increased or weakened confidence in the results of the original experiment. Its main benefit is to help mature the software engineering body of knowledge by addressing both conclusion and external validity problems in existing experiments. Regarding internal validity, an exact replica aims to confirm that the variables taken into account (i.e., the replicated ones) are the ones influencing results [41]. In fact, an exact replica providing a result different from the original experiment would suggest the existence of variables influencing results that are beyond the knowledge of the researcher. Regarding conclusion validity, an exact replica increases statistical power by providing further data to analyze, together with the original experiment.

Therefore, in order to increase statistical power (required by the high number of treatments) and our confidence in the results, we enacted an exact replica [41] of the controlled experiment, at the University of Rey Juan Carlos (URJC) in Madrid (Spain).

Though we did not change the experimental settings, there are some minor differences among the experiment and its replica: the subjects' background was computer engineering, and computer science in the experiment and replica, respectively. Moreover the experimental material was translated from Italian to Spanish.

4.3 Experimental units

Fifty graduate students belonging to a Master's course in computer engineering at the University of Rome TorVergata (Rome) participated in the experiment. Twenty-five graduate students belonging to a Master's course in computer science at the King Juan Carlos University (URJC) participated in the replica.

All the students, during previous bachelor and master courses, attended extensive teaching on different phases of the software lifecycle, including requirements engineering and software architecture analysis and design. Some of the students had industrial experience or worked as private consultants.

Because the use of students as subjects can be considered as one of the main threats to validity of the present study, we provide a careful discussion of this issue in Section 5.3.5.

4.4 Variables

The *output and dependent variable* is the **Value** of an information item of a design rationale documentation; Value represents the level of support as perceived by the subjects for enacting an activity. We measure the Value by a 3-point ordinal scale (Useless, Optional, or Required), for each specific information item, after having executed an activity.

Among the *independent variables*, we considered the following two *factors*:

- **The activity to enact by subjects (Activity).** It comprises the following five levels:
  - o *Detecting wrong aspect in decisions***.** Which characteristics of the chosen decision are wrong?
  - o *Detecting wrong requirements understanding.* Which characteristics of the problem to address have been misunderstood?
  - o *Checking design (verification and evaluation)*. Do you approve the decision made?
  - o *Detecting conflicts between new requirements and an old decision.* Is the old decision still valid for the new set of requirements?
  - o *Evaluating impact***.** What is the impact of new requirements on the system?
- **The category of the information item (Category).** In the present study we assigned 13 levels to this factor (Table 1) as proposed by Tyree and Akerman in [1] for documenting design decisions. Let us also note that according to Tyree and Akerman, an information item always belongs to a single category.

Other variables were controlled to avoid confounding effects, such as the level of experience of the participating subjects, experimental materials, and complexity of the experimental objects.

4.5 Tasks

Subjects received a set of design decisions (five decisions each) and a set of requirements for a software-intensive system. Based on this material, each subject enacted a total of twenty-five activities: all five activities on five decisions. For each of the five decisions, the subjects had to execute the following steps:

- Understand the activity to enact.
- Write the start time.
- Read the DRD related to a specific decision.
- Execute the activity (write the requested answer).
- Write the final time.
- For each DRD category, describe how much support is provided to the activity: Useless, Optional, or Required.

4.6 Experimental Material

To emulate a realistic situation, we devised an artificial software system design which is similar to another, realistic system successfully used in another experimental study [35]. The project is about a public transportation system with ambient intelligent characteristics [42], such as heterogeneous sensors. Since we had: (1) twenty-five subjects, (2) up to five competence types (as required by our experiment object), and (3) the need to make the design as balanced as possible, we decided to use *five* activities. Among all possible relevant activities [43], we selected the five activities having the lowest level of (expected) validity threats given the available experiment time and the subjects' experience.

Table 1: Categories of information of design decision rationale documentation as proposed in [1].

Issue	Describe the architectural design issue you're addressing, leaving no questions about why you're addressing this issue now. Following a minimalist approach, address and document only the issues that need addressing at various points in the life cycle.
Decision	Clearly state the architecture's direction—that is, the position you've selected.
Status	The decision's status, such as pending, decided, or approved.
Assumptions	Clearly describe the underlying assumptions in the environment in which you're making the decision—cost, schedule, technology, and so on. Note that environmental constraints might limit the alternatives you consider.
Constraints	Capture any additional constraints to the environment that the chosen alternative might pose.
Positions	List the positions (viable options or alternatives) you considered. These often require long explanations, sometimes even models and diagrams. This isn't an exhaustive list. This section also helps ensure that you heard others' opinions; explicitly stating other opinions helps enroll their advocates in your decision.
Argument	Outline why you selected a position, including items such as implementation cost, total ownership cost, time to market, and required development resources' availability.
Implications	A decision comes with many implications. Clearly understanding and stating your decision's implications can be very effective in gaining buy-in and creating a roadmap for architecture execution.
Related decisions	It's obvious that many decisions are related; you can list them here. However, we've found that in practice, a traceability matrix, decision trees, or metamodels are more useful. Metamodels are useful for showing complex relationships diagrammatically.
Related requirements	Decisions should be business driven. To show accountability, explicitly map your decisions to the objectives or requirements. You can enumerate these related requirements here, but we've found it more convenient to reference a traceability matrix. You can assess each architecture decision's contribution to meeting each requirement, and then assess how well the requirement is met across all decisions.
Related artifacts	List the related architecture, design, or scope documents that this decision impacts.
Related principles	If the enterprise has an agreed-upon set of principles, make sure the decision is consistent with one or more of them. This helps ensure alignment along domains or systems.
Notes	Because the decision-making process can take weeks, we've found it useful to capture notes and issues that the team discusses during the socialization process.

Concerning the requirements changes driving the activities, we selected common change causes such as: (1) variations in industrial strategic partnerships, (2) changes in customer requests resulting from experience using the previous version of the product, and (3) technology advances.

We adopted a total of twenty-five decisions; such a high number is expected to mitigate the influence of the peculiarities of the adopted decisions on the empirical results. These twenty-five decisions are all, in some sense, architectural decisions; for instance:

- The selection of a communication protocol depends on the topology of the nodes, the specific communication mechanism (e.g., publish-subscribe or event-driven) and architectural style (e.g., blackboard or client-server).
- The selection of a data storage mechanism depends on the type of DMBS, the communication protocol, and the architectural pattern (e.g., MVC).

Further drivers for architectural decisions included: available budget, desired compatibility, maintainability, scalability, and security. Constraints, requirements (new and old), rationale, and related decisions (and their status) are described in the provided DRD.

Table 2 shows the form that subjects filled out during the experiment for a given decision. The first column describes the activity to execute. The following three columns describe the initial time, the output of the activity (Answer, column 3), and the final time when the activity was completed, respectively. Columns 5 to 17 describe thirteen levels of Value: columns identify the Category; rows identify the Activity. Thus, each cell in the columns 5 to 17 describes the Value, as perceived by subjects (Useless, Optional, or Required), of a given Category (column), for supporting a given Activity (row).

Table 2: Form that subjects filled in during the experiment. Each cell in the columns 5 to 17 describes the value, as perceived by subjects (Useless, Optional, or Required), of a given DRD category (column), for supporting a given activity (row).

Order	Answer			Value of DRD information												
Activity	Initial Time	Answer	Final Time	Issue	Decision	Status	Assumptions	Constraints	Positions	Argument	Implications	Related decisions	Related requirements	Related artifacts	Related principles	Notes
1																
2																
3																
4																
5																

4.7 Design

We designed the experiment to make the best possible use of the available subjects. In particular, using an artificial project related to ambient intelligence allowed us to model five different areas of expertise: authentication, human interface, operating system, communication protocol, and data storage. Then, subjects expressed their preferences for each area, according to their previous experience and level of confidence. Afterwards, we assigned one area of expertise to each subject and the same number of subjects per area, by satisfying subjects' preferences to the maximum extent possible. This procedure aimed at ensuring that subjects had a realistic ability and level of confidence in the task to perform.

We then prepared five design decisions for each area of expertise. Finally, we designed the experiment to balance:

- Number of activities: all treatments (i.e., activities) have been applied the same number of times on all decisions. This mitigates the influence of activity characteristics and of decisions' peculiarities on the empirical results.
- Order of activities: all activities have been applied the same number of times in all the available orders (i.e., first to fifth). This mitigates the influence of the order in which the treatments are applied on the results.

4.8 Execution Preparation

The training phase was performed in three sessions of a total of five hours. During the training we taught the concepts of design decisions and design rationale, and the importance of capturing such knowledge. Then, we described the basis and the steps of the controlled experiment and how it would be carried out. We clearly explained almost all the experiment characteristics however we hided the experimenters' expectations.

We carefully checked for the attendance of the subjects to all the training sessions. As a result, five out of thirty-two students were excluded because they did not attend one or more training sessions. In the end, we had twenty-seven students that could be considered properly trained for the experiment. Because the experiment design is balanced only in case of twenty-five subjects, then we randomly selected two students as spares, to compensate for possible subject absence in the experiment.

4.9 Execution Deviations

While conducting the experiment and the replica, no particular deviation from their plan was observed. Regarding the experiment, on one occasion where a subject was absent, we replaced him with one of the spare subjects, thus preserving the experiment balance.

4.10 Data Set Collection and Validation

In the original experiment, 50 subjects performed five activities (1 activity on 5 decisions) thus yielding 250 activity executions. Because for each activity execution we had a set of 16 answers to collect (initial time, answer, the Value of each of 13 information categories, and final time), the experiment produced around 4,000 data items, 3,250 of which represents values of DRD information (13 information on 250 activity executions). Similarly because in the replica we had 25 subjects enacting 25 activities (5 activities on 5 decisions), the replica produced 10,000 data items, 8,125 of which represents values of DRD information. The total amount of data is therefore 14,000; 11,375 of which represent values of DRD information.

To detect and correct mistakes during the data transcription, we re-dictated and checked the data three times. Moreover, we applied some sanity checks based on automatic semantic analysis techniques. For example, for each form, the "initial time" for enacting an activity is checked to: i) precede the "final time" of the same activity, and ii) follow the "final time" of the previous activity. As a result we excluded few invalid data points from any further analysis.

5. Results and Interpretation

5.1 R.Q. 1: Is the Value of an information item significantly affected by its category and the activity that it supports?

5.1.1 Analysis Procedure

Because the original experiment and its replica do not differ much, we merged their data to maximize statistical power. However, we also investigated the consistency of their results as discussed in last paragraph of this subsection.

Multiple Correspondence Analysis (MCA) [7] is a standard and very convenient statistical procedure to visualize in a two-dimensional space the associations among the levels of three or more categorical variables; in our case, Activity, Category, and Value. Though rarely applied in software engineering, MCA is particularly useful for three or less variables and when variables show many levels as it can provide a global view of the data facilitating interpretation. MCA produces a *symmetrical variable plot* where each point represents a given level of a categorical variable. A key principle of MCA is that the distance among points on that two-dimensional plot is a meaningful measure of the degree of association of the respective variables' levels. Therefore, when levels of different variables appear close in the plot it means that they are strongly associated. Moreover, when all the levels of the *same* variable appear close to each other, this implies there is a small difference among them and that, as a result, the variable has a low impact. We meet the conditions [7] under which MCA is an appropriate and useful technique as we are only considering three discrete variables but with many levels. Furthermore, we have a minimum of eight observations in each level combination of these variables.

In order to analyze interaction effects between Activity and Category, we applied interactive coding analysis [7], which consists in creating a new variable having as levels all the combinations of levels for Value and Activity. Given the high number of combinations (13x5=65), in order to help visualization, we reported these interaction results across 13 distinct figures (Figure 5), each figure reporting on one Category level at a time and its combinations with the five Activity levels. So each figure reports eight points: the three levels for Value (VR, VU, VO) plus the five levels for Activity when combined with a particular Category level. The more spread these five points, the stronger the interaction effect between Activity and the figure's specific Category level.

For statistically testing the impact of Activity and Category on Value we applied logistic regression for ordinal response variables (Value in our case) with Activity, Category, and their interaction term as explanatory variables [44]. The significance of the effect of each explanatory variable is tested using the standard Likelihood Ratio Chi-square test [44]. We meet the data requirements as the sample size is considerably large with 11,000 observations.

In order to analyze the difference between the controlled experiment and its replica results we apply two independent MCA analyses based on their two respective data sets. We then plot the results of both analyses in one MCA Symmetric Variable Plot to assess the variation in positions for all Category and Activity levels. The higher the distance for each pair of points corresponding to the same level but a different experiment, the more inconsistent the results. What we mean to assess is whether the main conclusions are the same for both experiments.

5.1.2 Results

Table 3 describes the results of MCA in a standard form: the coordinates of each level of each variable in the symmetric variable plot, as depicted in Figure 4. The "Required" Value level (VR) is in the top-left quadrant and the closer the Category and Activity levels to this point, the stronger their statistical association with it. For example, CatRR is strongly associated with VR, therefore suggesting that "related requirements" is a very important piece of design rationale information. Similarly, category CatRA is closely associated with VU, thus suggesting that "Related artifacts" is not a relevant piece of information. Figure 4 also shows Activity levels to be much closer to the center, thus showing that Activity does not explain nearly as much of the variation in Value as Category does.

Table 4 reports the results of the Likelihood ratio Chi-square test to assess the significance of the impact of Activity, Category, and their interaction on Value. Because p-values related to Category, Activity, and their interaction, are far less than 0.05, then both their main and interaction effects on Value are significant at a 95% confidence level.

Table 3: Abbreviations and results of Multiple Correspondence Analysis.

Variable	Treatment	Abbreviation	F1	F2
Category	Issue	CatIs	-28.225	3.257
	Decision	CatD	-34.549	46.667
	Status	CatS	-4.362	-18.965
	Assumptions	CatAs	-8.528	-33.270
	Constraints	CatC	18.700	0.828
	Positions	CatP	-15.078	-12.607
	Argument	CatAr	-19.953	-3.181
	Implications	CatIm	8.872	-3.979
	Related decisions	CatRD	-9.133	-5.353
	Related requirements	CatRR	-20.438	15.138
	Related artifacts	CatRA	37.141	9.952
	Related principles	CatRP	22.678	-30.388
	Notes	CatN	52.876	31.902
Activity	1: Wrong solution space	A1	-3.973	4.461
	2: Wrong problem space	A2	-6.480	6.694
	3: Decision verification	A3	9.388	8.056
	4: Conflicts detection	A4	-0.010	-2.492
	5: Impact evaluation	A5	1.064	-16.715
Value	Required	VR	-74.422	41.726
	Optional	VO	-3.504	-77.388
	Useless	VU	83.001	26.514

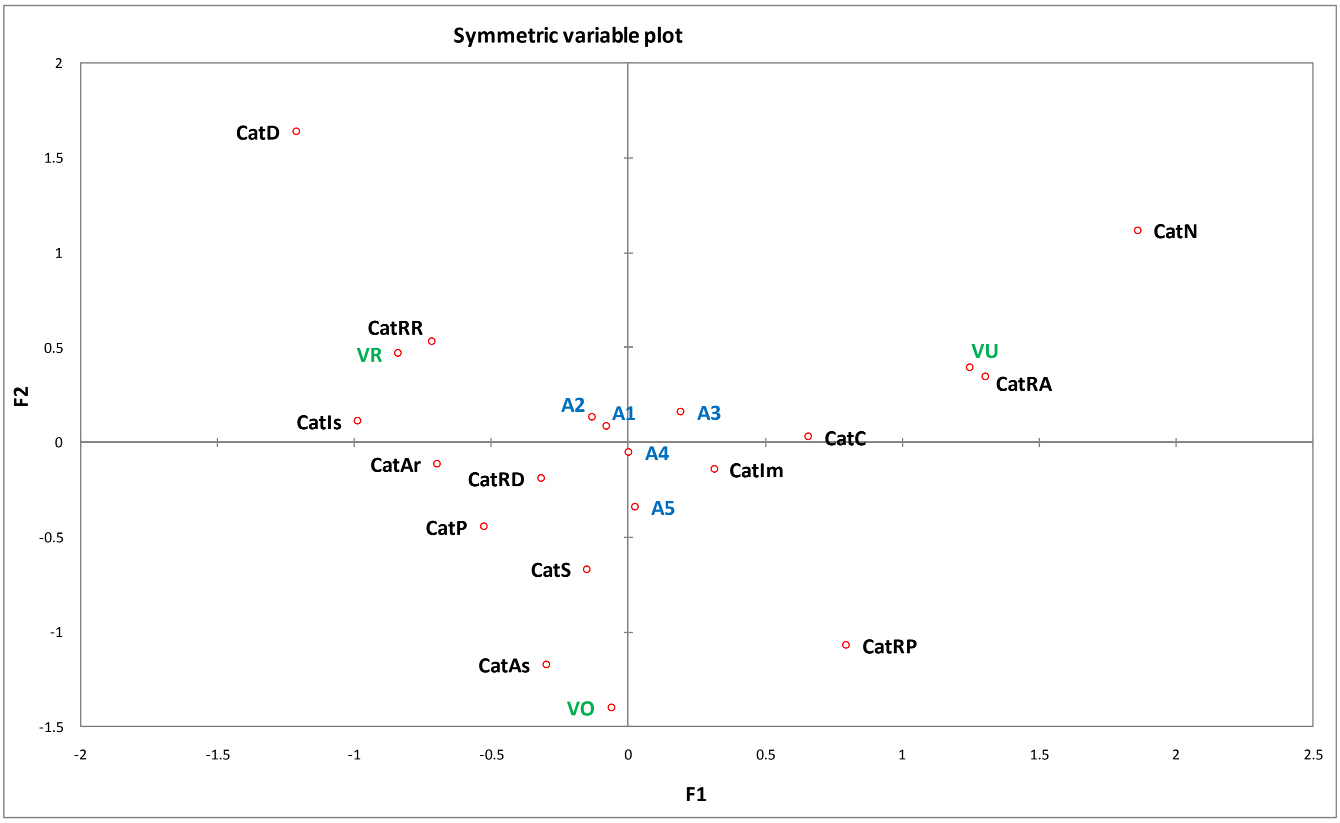
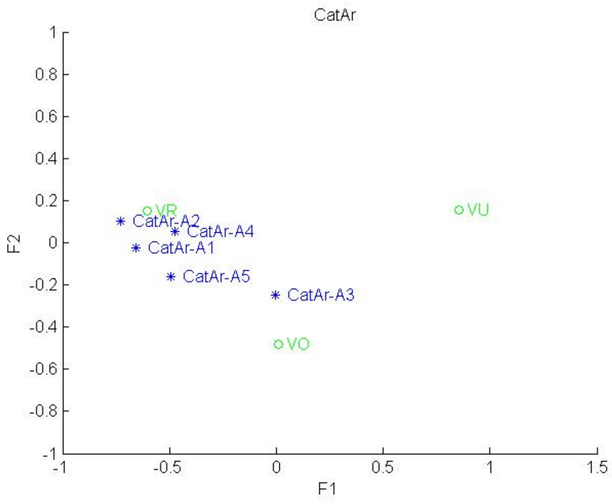
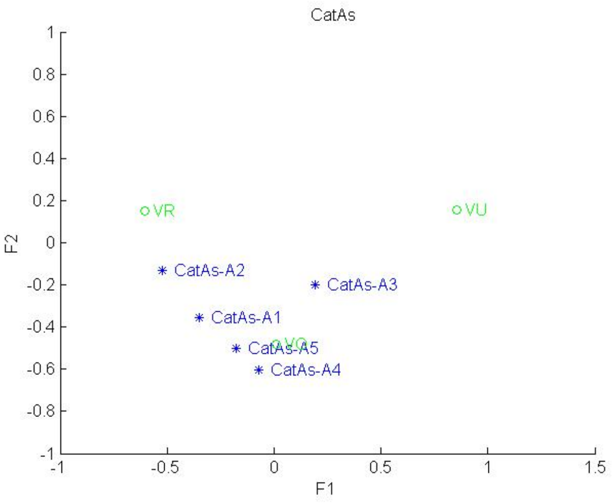
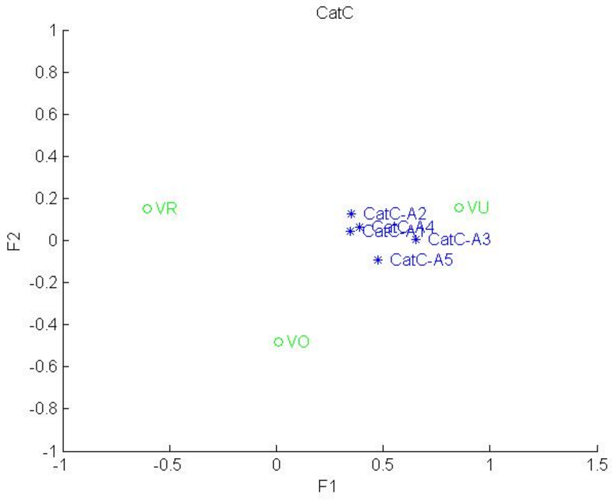
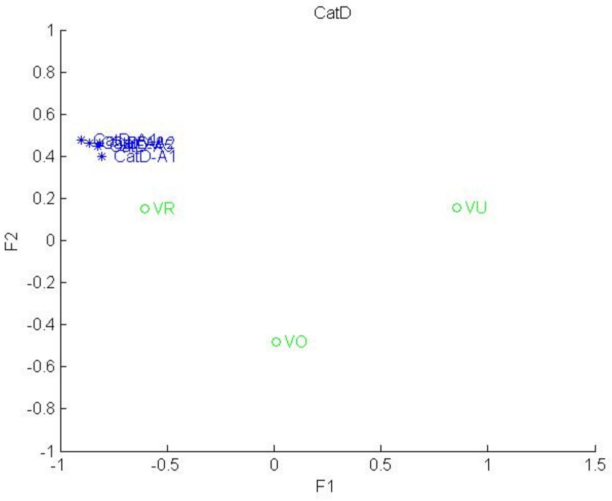


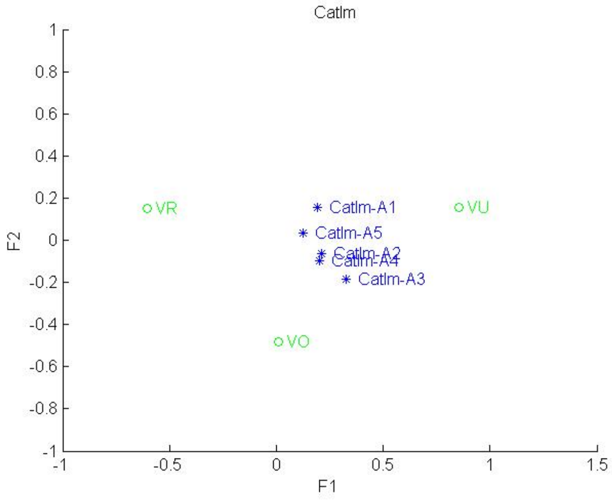
Figure 4: Graphical results of Multiple Correspondence Analysis.

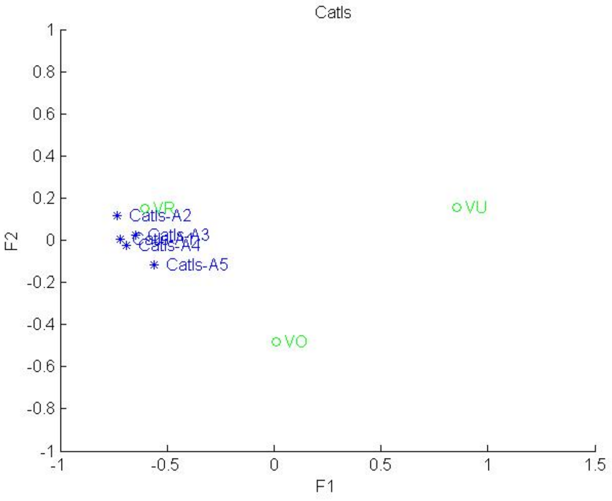

CatAr
F2
F1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
1.5
VR
VU
VO
CatAr-A2
CatAr-A4
CatAr-A1
CatAr-A5
CatAr-A3

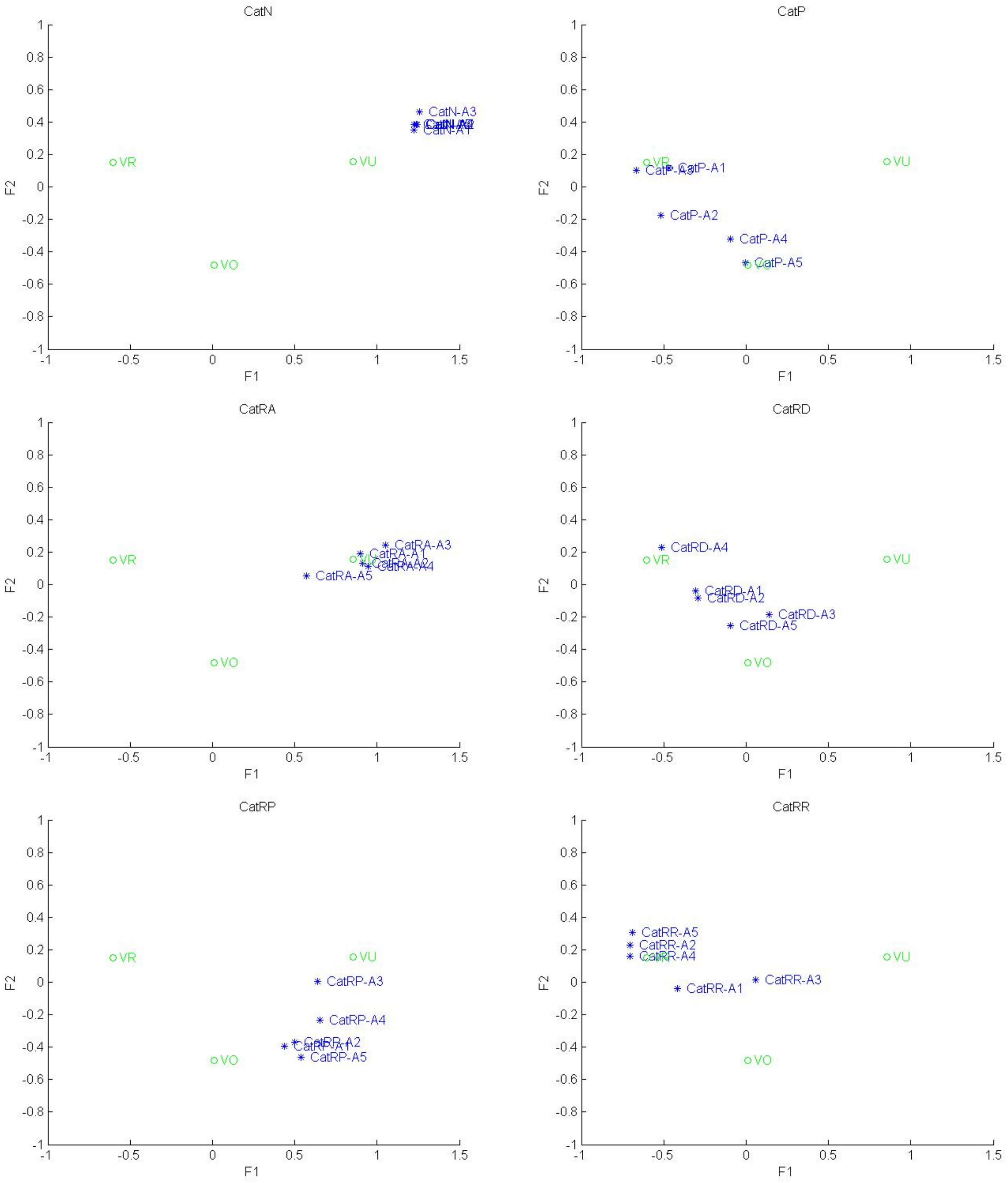

CatAs
F2
F1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
1.5
VR
VU
VO
CatAs-A2
CatAs-A3
CatAs-A1
CatAs-A5
CatAs-A4


CatC
F2
F1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
1.5
VR
VU
VO
CatC-A2
CatC-A4
CatC-A1
CatC-A3
CatC-A5


CatD
F2
F1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
1.5
CatD-A1
VR
VU
VO


CatIm
F2
F1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
1.5
VR
VU
VO
CatIm-A1
CatIm-A5
CatIm-A2
CatIm-A4
CatIm-A3


CatIs
F2
F1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
1.5
VR
VU
VO
CatIs-A2
CatIs-A3
CatIs-A4
CatIs-A5


CatN
CatP
CatRA
CatRD
CatRP
CatRR
F1
F2
-1
-0.5
0
0.5
1
1.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
VR
VU
VO
CatN-A1
CatN-A2
CatN-A3
CatN-A4
CatP-A1
CatP-A2
CatP-A3
CatP-A4
CatP-A5
CatRA-A1
CatRA-A2
CatRA-A3
CatRA-A4
CatRA-A5
CatRD-A1
CatRD-A2
CatRD-A3
CatRD-A4
CatRD-A5
CatRP-A1
CatRP-A2
CatRP-A3
CatRP-A4
CatRP-A5
CatRR-A1
CatRR-A2
CatRR-A3
CatRR-A4
CatRR-A5

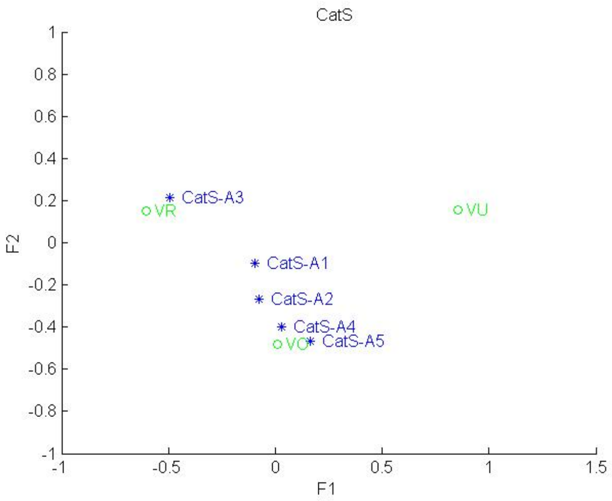


Figure 5: Interaction effect analysis through Multiple Correspondence Analysis.

Table 4: Likelihood Ratio Chi-square test results of the effect of Category, Activity, and their interaction on the Value of an information item.

Source	Nparm	DF	Likelihood Ratio Test	
			ChiSquare	P-value
Activity	4	4	45	<< 0.001
Category	12	12	4251	<< 0.001
Category*Activity	48	48	403	<< 0.001

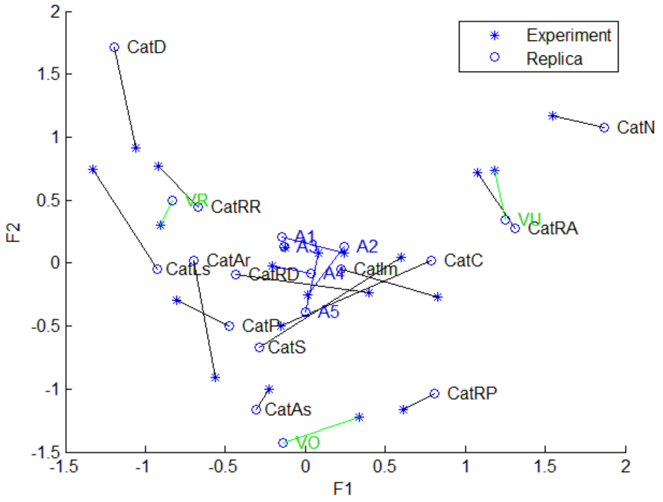


Figure 6: Difference between Experiment and Replica through Multiple Correspondence Analysis.

5.1.3 Discussion

Both Category and Activity have statistical impact on Value. In Figure 4, the distance between different levels of Category and Activity and different levels of Value vary significantly. The statistical analysis in Table 4 confirms statistical significance.

However, according to Figure 4, all the levels of Activity are close to each other and near the center; this means that Category is a much stronger predictor than Activity on Value. This is confirmed by the statistical analysis in Table 4; in fact, the Chi-Square value of Activity is much lower than that of Category.

There is a significant interaction effect between Activity and Category on Value. The statistical analysis in Table 4 shows that the interaction term between Activity and Category is statistically significant. In Figure 5, for six of the Category levels, there is a difference in position for different activities: Assumptions, Argument, Positions, Related Decisions, Related Principles, and Status. However, we note that when there is an interaction effect, Value changes only from Required or Useless to Optional (or vice versa) and never from Useless to Required (or vice versa). This means that, even when present, the interaction effect has a limited effect on Value.

Categories of design rationale documentation can be divided in three groups according to the resulting value:

- **High and independent from the activity to support**. Categories that are very valuable, independently from the activity to support, include the description of the chosen alternative (CatD) and of the related requirements (CatRR).
- **High or low according to the activity to support**. Some categories are only relevant for some of the activities to support. For instance, the category Positions (i.e., the description of the considered alternatives) provides value only in case the activity to support is Decision verification or Wrong solution space. On the contrary, Positions is irrelevant to support Impact evaluation (Activity 5). This makes intuitive sense as, given new requirements, it is irrelevant to know the set of considered alternatives because what is relevant is only the chosen alternative and whether it meets the new requirements.
- **Low and independent from the activity to support**. Finally, one category shows little usefulness regardless of the activity to support: Notes. This result shows that the proposed framework to document DRD is comprehensive and no further comments need to be added for documenting a design decision.

Obviously, the experiment results are influenced by the specific set of activities considered, the type of subjects, and so on. However, the main aim of the paper is to analyze whether there is an impact of Category and Activity on Value. In other words, we want to determine whether DRD consumers require specific information depending on the activities to be supported. Therefore, one may want to prioritize the DRD information to collect according to activities that are the most in need of improvement.

There are only minor differences between the results of the controlled experiment and the exact replica. Based on Figure 6, the position of the 13 categories did not significantly vary. This is especially the case for those categories that were most strongly associated with Value levels and further away from the center. For example, CatD and CatRR remain close to VR. Visible variations are for those categories near the center that have no strong association with any Value level. This result suggests, for the main associations, that the results are consistent across the two experiments.

5.2 R.Q. 2: How much effort can be saved by adopting a value-based DRD?

5.2.1 Analysis and Results

In the absence of evidence showing actual differences of required effort among information items, we assume that the number of items of a design rationale documentation to be a good indicator of the effort entailed. To compute the expected effort savings in adopting a value-based DRD, we compute the percentage of information items considered relevant and valuable for a given activity (value-based DRD).

As explained above, a value-based DRD is composed of valuable information items only. An information item is defined as valuable or expensive according to a threshold on Value; in particular, a threshold on a given frequency of Required among subjects' scores. Adopting a medium threshold (frequency of Required = 50%) results in documenting only the information categories that are more likely than not to support the readers (i.e., the categories of information that have been perceived at least the half of the time as required by subjects). Though a medium threshold makes sense, we also analyze the expected effort savings in relation to other thresholds. Obviously, the lower the threshold, the more the number of information items included in the value-based DRD, the higher the required effort, the less the probability that a subject would require an information that is not included in the DRD.

Figure 7 describes the number of information categories in the value-based DRD according to both the threshold on Value (Lowest to Highest) and the activity to be supported (e.g., Impact evaluation). Figure 7 is composed of a histogram and a table. In the histogram, columns show, for each supported activity, the number of DRD information categories associated with each frequency threshold for Required. The table provides the exact number of information categories for each percentage threshold. For instance, a value-based DRD for the Decision verification activity (third column) with a medium threshold (frequency of Required = 50%, third row) requires only four information categories out of 13.

5.2.2 Discussion

According to the third row in Figure 7, the number of information items included in a value-based DRD, tailored for a medium threshold, varies from 4 (third and fifth columns) to 8 (first column) depending on the activity to support. Therefore, the effort savings in documenting only the valuable information (i.e., the ones that is more likely to support the reader in a given activity) varies from a maximum of 70% (*Decision verification* and *Impact evaluation*) to a minimum of 40% (*Checking wrong solution space*).

On average across activities, a value-based DRD includes only 6 information items out of 13, which is 46% of a full documentation.

The significant reduction of information to document is therefore expected to mitigate the effects of inhibitors that are currently preventing practitioners from documenting the rationale of their decisions. Though these results are generalizable only to the adopted five activities, they have practical relevance since these activities have a widespread importance.

5.3 Threats to Validity

It is not possible to eliminate all the threats to validity; experimenters should prioritize them according to their context and research question to address [45]. In particular, there is always a tradeoff between realism and control [46]. Because of our research questions and the circumstances, we prioritized control over realism. In fact, our purpose was, to reveal the magnitude of differences in the values of an information item according to its category and the activity that it supports and, not to provide a fine-grained value. In the following subsections we provide insights regarding the validity of the aforementioned results based on the possible threats to validity as suggested in [45,47].

As in our previous study [35], a key choice was to use single decisions as the empirical objects rather than the whole set of decisions; this is not contradictory with the current trend to consider software architecture as a set of design decisions [48,49]. In fact, breaking down the decision process was a wile that helped us to face some challenges as described in the following paragraphs.

5.3.1 Threats in the Application of Empirical Software Engineering to Software Architecture

We describe here some of the relevant challenges in applying empirical software engineering to software architecture [47].

Fuzzy boundaries: To avoid misunderstandings regarding the term software architecture and design we clearly defined the boundary of our study: single design decisions and specific areas of expertise (authentication, human interface, operative system, communication protocol, and data storage).

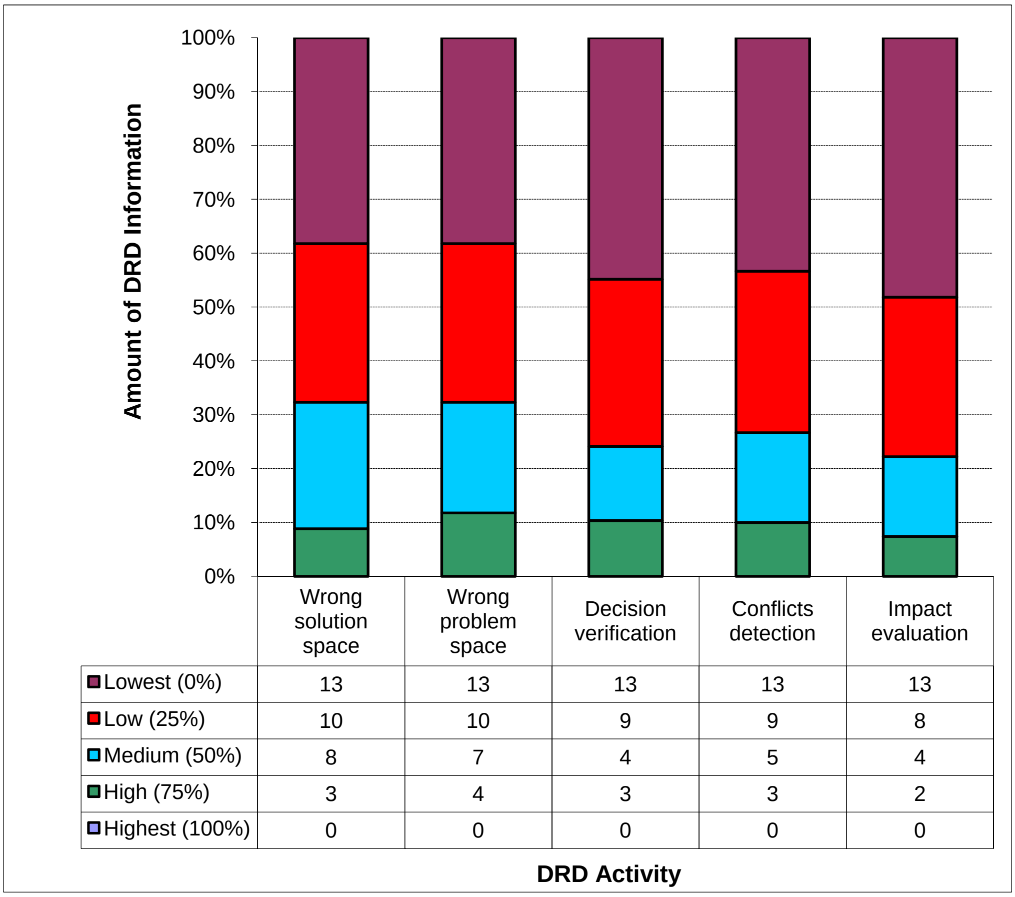
System complexity: The level of difficulty of the experiment task was realistic because, as in the real world, several opposite and interrelated objectives characterized these complex decisions [50].

5.3.2 Conclusion Validity

Conclusion validity concerns the reliability of the observed relations among the experimental variables [45,51].

Random heterogeneity: subjects were rather homogeneous since they took the same university program. Since the experiment design is balanced in different aspects, eventual differences should be balanced across treatments, and hence did not influence the results.

Fishing: We chose neither the 13 DRDs nor the five activities; therefore the analysis of variance among the combinations of categories and activities was not a fishing expedition.



	Wrong solution space	Wrong problem space	Decision verification	Conflicts detection	Impact evaluation
🟪 Lowest (0%)	13	13	13	13	13
🟥 Low (25%)	10	10	9	9	8
🟦 Medium (50%)	8	7	4	5	4
🟩 High (75%)	3	4	3	3	2
🟪 Highest (100%)	0	0	0	0	0

Figure 7: Amount of information in a tailored DRD according to different thresholds on Required.

5.3.3 Internal Validity

Internal validity is the degree to which conclusions can be drawn regarding the causal effect of the independent variables on the dependent variables [45,51]. In our view, the level of internal validity related to our experiment should be considered high since we balanced the order of activity executions and we randomized the application of treatments.

5.3.4 Construct Validity

Construct validity is the degree to which the independent variables and dependent variables accurately measure the concepts they purport to measure [45,51]. This was addressed in multiple ways.

Mono-operation bias: this was the threat with highest priority. To prevent experimental results from being too specific to the experimental objects (decisions), we adopted a set of twenty-five different objects. Hence, since all the subjects performed all (five) activities, for each dependent variable we have a distribution composed of two hundred fifty data points.

Restricted generalizability across constructs: regarding the proposed value-based DRD, the present study analyzes its efficiency (i.e., documentation effort reduction) without considering the effectiveness aspect. In other words, it may be unclear whether a reduced documentation negatively affects the output quality of the activity it supports. In general, the quality of a design decision (i.e., the activity output) is difficult to measure [47]. In this study, we indirectly and implicitly estimate effectiveness as the probability that a subject requires information that is not included in the customized documentation.

Hypotheses guessing and experimenter expectancies: During the training sessions, we did not convey our expectations and hypotheses to participants.

Evaluation apprehension: We informed the subjects that they would not be personally evaluated based on their answers.

Low motivation: Subjects were not pressured to participate in the experiment. We clearly explained that their course grade would not be related to their presence or performance during the experiment. This is an approach we have successfully adopted over several years in other experiments [47,52,53]. We observed a high level of commitment and concentration, as visible on the video we posted at: http://eseg.uniroma2.it/DDRD-Experiment-December2006.zip.

5.3.5 External Validity

External validity is the degree to which research results can be generalized to other settings than the experiment one. Since the use of students as subjects may be interpreted as an important threat to external validity, we carefully described the experimental process by following standard guidelines [54] in order to enable replications in different, possibly more realistic, contexts. The following list discusses how we addressed various external validity threats, with an emphasis on the experimental subjects and objects.

Experimental Subjects: The use of students as experimental subjects has been largely described, for example in [55]. A number of studies show that differences between students and practitioners may not be relevant. This is the case, for example, of studies in the context of requirements selection [56] and assessment of lead-time impact [57]. There is no evidence supporting the hypothesis that students' performance differs from that of practitioners when performing these activities.

About half of the subjects had a significant professional experience. In addition, most computer science and engineering courses include practical exercises or projects; therefore, computer-science master students should be considered not so far from practitioners as claimed in [58].

In addition, since an information item represents a piece of knowledge, it is obvious that people with less experience or knowledge tend to require more information items than more experienced people. Therefore, more experienced readers require less documentation, which results in greater effort savings for documenting only the required information. Consequently, assuming students have less experience than practitioners, the effort savings in adopting a value-based DRD should logically be expected to be higher in industrial contexts than in our experimental setting.

Experimental objects: Regarding the representativeness of the experimental objects, we adopted a realistic system successfully used in another experimental study [35]. Moreover, the adopted decisions were complex since they were characterized by several, opposite and interrelated objectives, as is the case in reality. We decided to develop a large number of decisions (i.e., twenty-five, five per role) in order to study the impact of activities in a somewhat independent manner from the specifics of the decisions to be made. In other words, we defined a large number of varied decisions to achieve better external validity.

6. Conclusion and Future Work

Documenting software design rationale is a generally encouraged practice: recording a designer's line of reasoning enables it to be revisited in order to assess it, to approve it, or more simply to learn from it, regarding either the system being designed or the decision process itself. But the methods and tools proposed to document the design rationale have not been very successful; they all tend to try to maximize the benefits for the consumer of the rationale, at the expense of the producer—that is the designer—and they are therefore too onerous for systematic industrial use. The underlying intuition of our work is that design rationale documentation can be used in many activities (e.g., what if analysis, avoiding design erosion) but not all of them are enacted in every context. Therefore, documenting all the information that supports all the activities becomes somehow ineffective from a cost-benefit point of view; this eventually inhibits practitioners to document the rationale of their design decisions. The key idea is that design rationale documentation should be introduced (for adoption) only to support activities that are particularly hard to perform with the usual procedures and documentation. A value-based design rationale documentation contains only the information that will be required, with a significant probability, to support its readers in a given activity. Therefore, the aim of the paper was twofold: 1) validating the feasibility of the customization strategy, 2) assessing its effectiveness by computing the expected effort savings.

It is obvious that a complete documentation provides more benefits but it entails more costs than any customized documentation. Hence, in this paper we reported a study that investigates this customization activity by analyzing whether the decision to document or exclude an information item should rely on its category and the activity it aims to support. An efficient customization is essential because it provides a realistic means for mitigating the known inhibitors of documenting design rationale.

Results from a controlled experiment and its replica show that:

- The value of an information item is significantly affected by its category; moreover, different categories have different values for different activities. Therefore, it is possible to prioritize the information to document according to the category and the activities that design rationale documentation targets.
- On average, across activities, a value-based design rationale documentation contains 46% of the information in the full documentation. The maximum effort reduction provided by a value-based design rationale documentation is obtained for the "decision verification" activity: only four out of 13 information categories have been shown to be required more than 50% of the time.
- The exact replica provided similar results to the original controlled experiment; this suggests that all the important variables have been taken into account.

In conclusion, our results demonstrate the feasibility (hypothesis test) and the efficiency (effort reduction) of the proposed value-based design rationale documentation. The reduced amount of information to document (54%) is expected to mitigate the effects of inhibitors that are currently preventing practitioners from documenting the rationale of their decisions. A value-based design rationale documentation enables its practical adoption by focusing and investing documentation effort where it is most needed.

Because documentation deserves to be treated as an economic investment, we are currently developing a design rationale documentation supporting tool that includes an economic model. According to the level of support provided in the past by a specific information category for enacting a specific activity, the economic model would suggest to the designer which information to document, when it is expected to be needed and valuable in the future.

7. Acknowledgement

The work of Lionel Briand and Davide Falessi was supported by a grant from Det Norske Veritas, Norway, in the context of the ModelME! project.

8. References

[1] J. Tyree, and A. Akerman, "Architecture Decisions: Demystifying Architecture," *IEEE Software*, vol. 22, no. 2, pp. 19-27, March/April, 2005.

[2] P. Kruchten, "An ontology of architectural design decisions in software intensive systems," in Proceedings of the 2nd Groningen Workshop on Software Variability, 2004.

[3] D. Falessi, G. Cantone, R. Kazman, and P. Kruchten, "Decision-making Techniques for Software Architecture Design: A Comparative Survey " *ACM Computing Surveys*, vol. (in press), 2010

[4] J. Bosch, "Software Architecture: the Next Step," in First European Workshop on Software Architecture (EWSA 2004), St Andrews, Scotland, 2004, pp. 194-199.

[5] J. Lee, "Design Rationale Systems: Understanding the Issues," *IEEE Expert: Intelligent Systems and Their Applications,* vol. 12, no. 3, pp. 78-85, 1997.

[6] I. Gorton, *Essential Software Architecture*: Springer-Verlag New York, Inc., 2006.

[7] M. Greenacre, *Correspondence Analysis in Practice*, New York: Chapman & Hall/CRC, 2007.

[8] R. C. de Boer, and R. Farenhorst, "In Search for Architectural Knowledge," in Third Workshop on SHAring and Reusing architectural Knowledge (SHARK 2008) colocated with 30th Int. Conf. on Software Engineering (ICSE 2008) Leipzig, Germany, 2008.

[9] J. Burge, and D. Brown, *Design Rationale Types and Tools,* AI in Design Group, Computer Science. Department, 1998.

[10] J. Conklin, and L. M. Begeman, "gIBIS: a hypertext tool for exploratory policy discussion," in Proceedings of the 1988 ACM conference on Computer-supported cooperative work 1988.

[11] A. MacLean, R. M. Young, V. M. E. Bellotti, and T. P. Moran, "Questions, options, and criteria: elements of design space analysis," *Design rationale: concepts, techniques, and use*, pp. 53-105: Lawrence Erlbaum Associates, Inc., 1996.

[12] A. Tang, M. Ali Babar, I. Gorton, and J. Han, "A Survey of Architecture Design Rationale," *Journal of Systems & Software,* vol. 79, no. 12, pp. 1792-1804 2007.

[13] R. Capilla, Nava, and J. C. Dueñas, "Modeling and Documenting the Evolution of Architectural Design Decisions," in Proceedings of the 2nd Workshop on Sharing and Reusing Architectural Knowledge, ICSE Workshops, IEEE DL, 2007.

[14] P. Kruchten, P. Lago, and H. van Vliet, "Building up and Reasoning about Architectural Knowledge," in 2nd International Conference on the Quality of Software Architectures, Vaesteras, Sweden, 2006.

[15] J. S. Van der Ven, A. Jansen, P. Avgeriou, and D. K. Hammer, "Using Architectural Decisions," in Proceedings of the 2nd International Conference on the Quality of Software Architectures, Västerås, Sweden 2006.

[16] R. Farenhorst, J. F. Hoorn, P. Lago, and H. v. Vliet, "What Architects Do and What They Need to Share Knowledge," *Technical Report IR-IMSE-003, VU University Amsterdam,* 2009.

[17] A. Jansen, J. Bosch, and P. Avgeriou, "Documenting after the fact: Recovering architectural design decisions," *J. Syst. Softw.,* vol. 81, no. 4, pp. 536-557, 2008.

[18] L. Lee, and P. Kruchten, "Capturing Software Architectural Design Decisions," in Canadian Conference on Electrical and Computer Engineering, 2007. CCECE 2007., 2007, pp. 686-689.

[19] S. Biffl, A. Aurum, B. Bohem, H. Erdogmus, and P. Grünbacher, *Value-Based Software Engineering*, Berlin: Springer, 2006.

[20] A. Egyed, P. Grunbacher, M. Heindl, and S. Biffl, "Value-Based Requirements Traceability: Lessons Learned," in 15th IEEE International Requirements Engineering Conference (RE '07), 2007, pp. 115-118.

[21] P. Arkley, S. Riddle, and T. Brookes, "Tailoring Traceability Information to Business Needs," in 14th IEEE International Requirements Engineering Conference (RE'06), 2006.

[22] M. Heindl, and S. Biffl, "A case study on value-based requirements tracing," in Proceedings of the 10th European Software Engineering Conference, Lisbon, Portugal, 2005.

[23] A. Egyed, S. Biffl, M. Heindl, and P. Grunbacher, "Determining the cost-quality trade-off for automated software traceability," in Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, Long Beach, CA, USA, 2005.

[24] J. Cleland-Huang, "Just Enough Requirements Traceability," in Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06) - Volume 01, 2006.

[25] J. Cleland-Huang, "Requirements Traceability - When and How does it Deliver more than it Costs?," in Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06), 2006.

[26] V. R. Basili, and H. D. Rombach, "Support for comprehensive reuse," *Software Engineering Journal,* vol. 6, no. 5, pp. 303-316, 1991.

[27] D. Falessi, G. Cantone, S. A. Sarcià, G. Calavaro, P. Subiaco, and C. D'Amore, "Peaceful Coexistence: Agile Developer Perspectives on Software Architecture," *IEEE Software,* vol. 27, no. 2, 2010.

[28] S. W. Ambler, and R. Jeffries, *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process* Wiley, 2002.

[29] S. W. Ambler, "The TAGRI (They Aren't Gonna Read It) Principle of Software Development," in available at: http://www.agilemodeling.com/essays/tagri.htm 2007.

[30] J. E. Burge, J. M. Carroll, R. McCall, and I. Mistrk, *Rationale-Based Software Engineering*: Springer Publishing Company, Incorporated, 2008.

[31] M. Fowler, "The New Methodology," *http://martinfowler.com/articles/newMethodology.html*, 2005.

[32] L. Karsenty, "An empirical evaluation of design rationale documents," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Common Ground, Vancouver, British Columbia, Canada, 1996.

[33] L. Bratthall, E. Johansson, and B. Regnell, "Is a Design Rationale Vital when Predicting Change Impact? A Controlled Experiment on Software Architecture Evolution," in International Conference on Product

Focused Software Process Improvement, Oulu , Finland, 2000.

[34] O. Zimmermann, T. Gschwind, J. Küster, F. Leymann, and N. Schuster, "Reusable Architectural Decision Models for Enterprise Application Development," in QoSA 2007, Medford, MA, USA, 2007.

[35] D. Falessi, G. Cantone, and M. Becker, "Documenting Design Decision Rationale to Improve Individual and Team Design Decision Making: An Experimental Evaluation," in Proceedings of the 5th ACM/IEEE International Symposium on Empirical Software Engineering, Rio de Janeiro, Brazil, 2006.

[36] B. Shum, and N. Hammond, "Argumentation-Based Design Rationale: What Use at What Cost?," *International Journal of Human-Computer Studies,* vol. 40, no. 4, pp. 603 - 652 1994.

[37] I. John, J. Knodel, T. Lehner, and D. Muthig, "A practical guide to product line scoping," in Software Product Line Conference, 2006 10th International, 2006, pp. 3-12.

[38] P. Clements, J. D. McGregor, and S. G. Cohen, *The Structured Intuitive Model for Product Line Economics (SIMPLE),* CMU/SEI-2005-TR-003, 2005.

[39] J. Favaro, K. Favaro, and P. Favaro, "Value based software reuse investment," *Ann. Softw. Eng.,* vol. 5, pp. 5-52, 1998.

[40] R. Capilla, F. Nava, and C. Carrillo, "Effort Estimation in Capturing Architectural Knowledge," in 23rd IEEE/ACM International Conference on Automated Software Engineering, 2008. (ASE08), 2008, pp. 208-217.

[41] F. Shull, J. Carver, S. Vegas, and N. Juristo, "The role of replications in Empirical Software Engineering," *Journal of Empirical Software Engineering,* vol. 13, no. 2, pp. 211-218, 2008.

[42] J. Nehmer, M. Becker, A. Karshmer, and R. Lamm, "Living assistance systems: an ambient intelligence approach," in Proceedings of the 28th international conference on Software engineering, Shanghai, China, 2006.

[43] P. Kruchten, P. Lago, H. van Vliet, and T. Wolf, "Building up and Exploiting Architectural Knowledge," in Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), 2005, pp. 291-292.

[44] D. Hosmer, and S. Lemeshow, *Applied Logistic Regression*: Wiley-Interscience Publication, 2000.

[45] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering: an Introduction*: Springer, 2000.

[46] W. Dzidek, E. Arisholm, and L. Briand, "A Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance," *IEEE Transactions on Software Engineering,* vol. 34, no. 3, pp. 407-432, 2008.

[47] D. Falessi, M. Ali Babar, G. Cantone, and P. Kruchten, "Applying Empirical Software Engineering to Software Architecture: Challenges and Lessons Learned," *Empirical Software Engineering,* vol. 15, no. 3, pp. 250-276, 2010.

[48] P. Kruchten, *The Rational Unified Process: An Introduction* 3rd ed.: Addison-Wesley Professional, 2003.

[49] A. Jansen, and J. Bosch, "Software Architecture as a Set of Architectural Design Decisions," in 5th Working IEEE/IFIP Conference on Software Architecture (WICSA 5), Pittsburgh, 2005.

[50] H. R. Eguiluz, and M. R. Barbacci, *Interactions Among Techniques Addressing Quality Attributes,* CMU/SEI 2003-TR-003, 2003.

[51] N. Juristo, and A. M. Moreno, *Basics of Software Engineering Experimentation* Springer, 2006.

[52] D. Falessi, and G. Cantone, "Exploring Feasibility of Software Defects Orthogonal Classification," *Software and Data Technologies*, J. Filipe, B. Shishkov and M. Helfert, eds., Berlin: Springer 2006.

[53] G. Cantone, L. Colasanti, Z. Abdulnabi, A. Lomartire, and G. Calavaro, "Evaluating Checklist-Based and Use-Case-Driven Reading Techniques as Applied to Software Analysis and Design UML Artifacts," *ESERNET*, 2003.

[54] A. Jedlitschka, M. Ciolkowski, and D. Pfahl, "Reporting Experiments in Software Engineering," *Guide to Advanced Empirical Software Engineering* F. Shull, J. Singer and D. I. Sjøberg, eds.: Springer, 2008.

[55] J. Carver, L. Jaccheri, S. Morasca, and F. Shull, "Issues in Using Students in Empirical Studies in Software Engineering Education," in Proceedings of the 9th International Symposium on Software Metrics, 2003.

[56] M. Svahnberg, A. Aurum, and C. Wohlin, "Using students as subjects - an empirical evaluation," in Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Kaiserslautern, Germany, 2008.

[57] M. Host, B. Regnell, and C. Wohlin, "Using Students as Subjects; A Comparative Study of Students and Professionals in Lead-Time Impact Assessment," *Empirical Software Engineering Journal,* vol. 5, no. 3, pp. 201-214, 2000.

[58] D. Sjoberg, E. Arisholm, and M. Jorgensen, "Conducting experiments on software evolution," in Proceedings of the 4th International Workshop on Principles of Software Evolution, Vienna, Austria, 2001.