Value-Based Design Decision Rationale Documentation: Principles and Empirical Feasibility Study

Davide Falessi
University of Roma "Tor Vergata", DISP
Rome - Italy
falessi@ing.uniroma2.it

Giovanni Cantone
University of Roma "Tor Vergata", DISP
Rome - Italy
cantone@uniroma2.it

Philippe Kruchten
University of British Columbia ECE
Vancouver - Canada
pbk@ece.ubc.ca

Abstract

The explicit documentation of the rationale of design decisions is a practice generally encouraged, but rarely implemented in industry because of a variety of inhibitors. Methods proposed in the past for Design Decisions Rationale Documentation (DDRD) aimed to maximize benefits for the DDRD consumer by imposing on the producer of DDRD the burden to document all the potentially useful information. We propose here a compromise which consists in tailoring DDRD, based on its intended use or purpose. In our view, the adoption of a tailored DDRD, consisting only of the required set of information, would mitigate the effects of DDRD inhibitors. The aim of this paper is twofold: i) to discuss the application of Value-Based Software Engineering principles to DDRD, ii) to describe a controlled experiment to empirically analyze the feasibility of the proposed method. Results show that the level of utility related to the same category of DDRD information significantly changes depending on its purpose; such result is novel and it demonstrates the feasibility of the proposed value-based DDRD.

1. Introduction

The capture and reuse of the knowledge gathered while making design decisions is actually recognized to be one of the most promising step for advancing the software architecture state of the art by preventing its high costs of change, and the design erosion.

However, while we can find the documentation of the resulting design, it is usually not the case for the design rationale, i.e., the reasoning that brought the designer to a given choice. Authors have long touted the merits of capturing not only the design but also the rationale; however, the practice of Design Decision Rationale Documentation (DDRD) is not yet widely spread, due to some inhibitors, such as the required additional documentation effort. DDRD methods proposed in the past tend to maximize benefits for the DDRD consumer by imposing on the DDRD producer to document all the possible information. Because a tailored DDRD should focus just on the most valuable information, we suggest that the use of such a tailored DDRD would mitigate the effects of inhibitors and emphasize on the effects of motivators.

We propose a solution consisting in tailoring DDRD, based on its purpose (i.e., what we intend to do with the documentation). The objective of the present paper is twofold: (i) to discuss the application of Value-Based Software Engineering principles to DDRD, (ii) to describe a controlled experiment which empirically demonstrates that DDRD can be tailored based on its purpose (i.e., different DDRD "use cases" require different information).

The paper is structured as follows: Section 2 gives some background for the present work. Section 3 describes the proposed Value-Based DDRD, while Section 4 describes its empirical feasibility study. The paper concludes with Section 5 including an outlook to future work.

2. Study motivation

2.1 Value-Based Software Engineering

Nowadays, "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" [1]. Consequently, "a resulting value-based software engineering (VBSE) 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" [1]. In the present work we apply VBSE principles on DDRD; we propose a Value-Based approach to DDRD (VB DDRD), which focuses on documenting only the set of required information based on its purpose.

2.2 Design Decision Rationale Documentation

2.2.1 Introduction There are many definitions of design rationale. According to Jintae Lee, "design rationales include not only the reasons behind a design decision but also the justification for it, the other alternatives considered, the tradeoffs evaluated, and

0-7695-3092-3/08 $25.00 © 2008 IEEE
DOI 10.1109/WICSA.2008.8

the argumentation that led to the decision" [2]. Nowadays, the use of DDRD seems to be promising to support software architecture design and, more important, its maintenance. But in reality, during the architecture development process, many important decisions are not documented explicitly, together with their rationale, but are immediately embedded in the models the architects build [3]; consequently, some useful knowledge attached to the decision and the decision process are lost forever [4]. This cause the design erosion and a high costs of software architecture change [5].

Three important studies have empirically investigated the benefits of using DDRD. Karsenty [6] proposed an empirical evaluation concerning the utility of DDRD in the maintenance of a nine-month old software project. Bratthall, Johansson, and Regnell [7] presented a controlled experiment to evaluate the importance of DDRD when predicting the impact of changes on software architecture evolution; their results show that DDRD clearly improves effectiveness and efficiency. Falessi, Cantone, and Becker [8] presented results from a controlled experiment with students, in individual and team-based decision-making, when decision-makers are allowed to use or not the DDRD, in the presence of requirement changes.

2.2.2 Inhibitors. Although several studies empirically demonstrated that the use of DDRD brings numerous benefits [6] [7] [8], such type of documentation is not widely adopted in practice; and we may wonder why. We focus on the following DDRD inhibitors:

Bad timing and delayed benefit. The period in which design decisions are taken is usually critical for the success of the software project. People involved in the development process, while taking design decision, are already busy in trying to do, as better as possible, the more recognized tasks and to meet the related deadline. In such circumstances, the task of enacting DDRD is considered less important than the other ones and consequently it is put at the end of the queue; and eventually it is never enacted. Our experience shows that when trying to suggest documenting design decisions rationale in the appropriate time, the people answer is likely to be "We already have a lot of problems!"

Information unpredictability. The DDRD consumer and producer are often different persons. People who are in charge to evolve a project are often not the original designers, who in the mean time moved to better, greener pastures. Hence, the DDRD producer cannot predict which information the consumers will need in the future. As a result, the producer documents all the information that could be useful. Finally, the DDRD becomes too much expensive to write, maintain, and read.

Overhead. Several DDRD techniques already exist; however they are usually focused on maximizing the consumer benefits rather than minimizing the producer effort. Consequently, people involved in the documentation and maintenance activities are supposed to spend a huge amount of effort.

Unclear benefit. Decision-makers do not know for which purpose it is useful to read the rationale of which decision.

Lack of motivation. Caused by absence of direct benefits or no personal interest. People in charge of documenting and maintaining DDRD artifacts (the decision-makers) are not the ones that directly benefit from DDRD; hence, they are not that motivated. Lee [2] already stressed the importance of the existing problem that DDRD producer and consumer differ. Moreover, experts may be not interested in making their valuable knowledge explicit as they may consider it as their personal property. In other words, some experts could see no clear advantage in doing DDRD.

Lack of maturity. Only few tools are currently available to support DDRD and therefore the field is to consider as rather immature.

Potential inconsistencies. DDRD implicitly represents the results of the design. If DDRD and the design documents are not well updated, potential inconsistencies in case of decision changes might occur.

2.3 Related Works

We could not find any previous study which applied value-based software engineering principles (or similar) to DDRD, but some works seem related. Lee [2] already stressed that "what you represent depends on what you want to do with it" and he found that different systems, aimed to support different activities, focused on different category of DDRD information. However, to the best of our knowledge, the present study is the only empirical investigation, involving subjects' opinion, aimed to evaluate the level of importance of different DDRD information for enacting different activities. Shum and Hammond [9] pointed out that without a good Return On Investment (ROI), the system in charge to manage DDRD would not be used or finally it would be counterproductive. Heindl and Biffl [10] reported a case study on a type of value-based requirements tracing, which aims to systematically support project managers in tailoring the traces, and is based on the following parameters: stakeholder value, requirements risk/volatility, and tracing costs. The main results of that case study are: (a) value-based requirements tracing takes around 35%

of the effort required by full tracing; (b) more risky requirements need more detailed tracing.

The relation between DDRD and its use cases have been empirically investigated in two main works by Van der Ven *et al.* [11] and Tang *et al.* [12]. Van der Ven's *et al.* [11] present a use case model that arose from industrial needs, and is meant to explore how DDRD can be satisfied through the effective usage of architectural decisions by the relevant stakeholders. Similarly to the present work, they clearly describe that different type of stakeholders enact specific DDRD use cases. Tang *et al.* [12] report on a survey trying to evaluate the importance of DDRD as perceived by industrial architects (or designers). The goal of their study was similar to ours, as they investigated which type of DDRD information is in general more used by practitioners. We try to go a step further by investigating the level of importance, related to each category of DDRD information, for enacting specific DDRD use cases.

Tyree and Akerman [3] proposed a framework to document design decision rationale for demystifying system architectures. In the present study we used such a DDRD template, as an example of instance of DDRD; we are investigating the level of utility of each category of such a DDRD template to enact different DDRD UC.

3. A value-based approach to design decision rationale documentation

3.1 Rationale

Past DDRD methods aimed to maximize the DDRD consumer's return by forcing the DDRD producer to document all the potentially useful information. Such methods have been employed independently from the system requirements and business context where they should provide benefits. As a matter of fact, we still have to evaluate them as value-neutral methods (see Section 2.1). However, because such a value-neutral approach strengthens DDRD inhibitors (see Section 0), the question is: What approach should we follow for mitigating the impact of those inhibitors? Our conjecture is that the answer is the injection of Value-Based software engineering principles on DDRD methods, which is what we aim to investigate and describe in the remaining of this paper.

Remarkably, DDRD is strongly related with making knowledge explicit and thus making the design process and the design results re-usable. Experiences with software reuse in the late nineties [13] showed that only strategic, pre-planned reuse that considers the features' value really pays off and allows reuse in the large. Transferring this experience to the DDRD context eventually results in the conclusion that DDRD has to be preplanned in order to be successful in the end. A clear strategy, which considers the potential benefits that can be drawn from DDRD at which costs and risks, is required. As the benefits, costs and risks differ from case to case, the strategy has to be defined anew for different development contexts. In the following section, we propose a Value-Based DDRD (VB DDRD). In particular, we propose to take crucial aspects of every software engineering practice into account: *Where* (project context), *Who* (beneficiary stakeholders), *When* (DDRD type of use), *Why* (Software or business metrics) and *How* (DDRD required information). The rationale behind this is to take advantage of an *a priori* understanding of who will profit later on, from what set of information, in which amount, in order to cope with the additional effort that has to be spent for producing and maintaining the DDRD. This allows a more Value-Based DDRD than the other DDRD processes already proposed in the literature.

In our best understanding, there has been no study investigating the need for different information for the different parts neither of the design nor for different type of decisions. Consequently our approach focus on tailoring the documentation based on its purpose rather than on the type of decisions because actually there is no suitable categorization.

3.2 DDRD Use-cases

Let us consider various usage scenarios for DDRD. A DDRD Use-Case (DDRD UC) occurs when one or more actors use DDRD in order to achieve a certain advantage while enacting a particular activity in a specific project context. A DDRD UC is characterized by the following attributes:

- ID: the identifier (primary key) of the scenario; it allows to establish associations among different tables, as shown in the remaining.
- Actor(s): an abstraction on the kind of people involved in the DDRD UC. The Producer is involved in providing and updating the DDRD. The Consumer takes advantages by the existence of DDRD.
- Context: System or industry characteristics where the usage scenario happens. This comprises information regarding the environment and the underlying driver.
- Activity: The activity that the actors enact, in which DDRD is used.

Advantages: Product or process metrics that determine the benefit of the DDRD usage.

In the remaining of the paper we will call DDRD UC(s) the different purposes of the DDRD as.

Table 1 describes a set of five DDRD UC(s) that we will use subsequently in this paper. This list is certainly incomplete but illustrative and valid. To summarize, the use cases of DDRD we consider are:

1. Identification of wrong knowledge on the solution space.
2. Identification of wrong knowledge on the problem space.
3. Design verification.
4. Detection of conflicts.
5. Impact analysis of changes.

In order to explain the meaning and practical usage of such DDRD UC(s), let us consider the usage scenario No. 5 in Table 1, which is a DDRD UC concerning the management of requirement changes. Nowadays, changes in the requirements or business goals are becoming more frequent than in the past. Although much effort has already been spent on software requirements engineering, we are still unable to make that discipline deterministic and fix the requirements in just one shot of the development process. Consequently, fixing software requirements is still a hard job even if customers are maturing and becoming increasingly able to define stable needs. In case of a requirements change, it is crucial for both managers and architects to understand which decisions (and which system artifacts) are still valid and which other ones have to be re-done (i.e., re-designed and/or re-implemented). Traceability among requirements, design decisions, and software artifacts allows managers to easily recognize the type of action required for each software artifact and, consequently, to re-schedule the project activities.

Table 1: DDRD use-case description.

DDRD UC						
ID	Actor		Context		Activity	Advantages
	Producer	Consumer	Environment	Driver		
1	Designers / Architects	Designers / Architects	Several designers with similar competence but different levels of expertise	Large system	Detection of wrong knowledge on decisions	System quality
2	Designers / Architects	Requirement Analysts	Ambiguous or conflicting or inter-related requirements	System complexity	Detection of wrong requirement understanding	Effort
3	Designers / Architects	Reviewers	Common context	Common context	Design checking (verification and evaluation)	Effort
4	Designers / Architects	Mainteners	Maintenance of a built system	System evolution	Detection of conflicts among new requirements and old decisions	Effort
5	Designers / Architects	Designers / Architects / Managers	Common context	Common context	Impact Evaluation	Effort

3.3 Key Idea

Let us call "required information" a kind of information without which, independently from the effort of the readers, the meaning of something cannot be understood; "useful information", a kind of information that helps to a small or large extent the readers to understand the meaning of something; "optional information", a kind of information which helps readers, but it is not required, in understanding something.

The key idea is that all the information included in a DDRD might be useful but sometimes some information is merely optional. We expect that the amount of importance related to the information included in the DDRD depends on the DDRD UC. In other words, we expect that different DDRD UC(s) require different categories of DDRD information. In such case, the DDRD can be tailored based on the DDRD UC(s) to enact. In our view, the adoption of a tailored DDRD, consisting only of the required set of information, would mitigate the effects of DDRD inhibitors, as described in the remaining of this section.

3.4 Process

describes the flow of activities (rectangles) and the flow of information produced/consumed (parallelograms) in our VB DDRD process. In order to select a scenario – among those that are still left for analysis, if any – from a predefined DDRD-UC database (see Table 1), the first activity in Figure 1 ("Scenario selection") is executed. Subsequently, information concerning the selected scenario is analyzed, which is:

- Context: to figure out the probability that this scenario will occur in the business and system context.
- Metrics: to realize the benefits that such a usage scenario provides.
- Required DDRD information: to estimate the cost of (amount of effort to spend on) documenting and maintaining DDR.

At this point, the adoption of DDRD in the current scenario(s) is economically evaluated for ROI. Since the benefit is achieved only in case the related scenario occurs, we weight the ROI described in [14] by the occurrence probability of the specific scenario (i.e., ROI(s)=OccurrenceProbability(s) * Benefit(s) / Cost(s).

Note that this approach is orthogonal on customizing the DDRD based on the importance of the decision.

3.5 Expected Advantages

Three main elements characterize our proposed value-based rationale documentation process:

- A clear definition of the advantages, the related overall value, and the roles of stakeholders involved in a particular DDRD UC execution
- A tailored DDRD involving only the required information
- The assumption that different DDRD UC(s) require different DDRD information.

shows which of those components we expect to mitigate which DDRD inhibitors. In particular:

Overhead: A tailored DDRD implies less information to document and maintain; hence, a

diminished effort has the effect of mitigating the overhead.

Potential inconsistencies: The tailored DDRD implies less information and hence less documentation. Less documentation implies both less required effort for DDRD maintenance and less probability of inconsistencies occurrence.

Bad timing & delayed benefit: The possibility to spend less time to produce the DDRD highly increases the possibility that people, who are busy to meet their projects deadlines, find enough time to develop such DDRD.

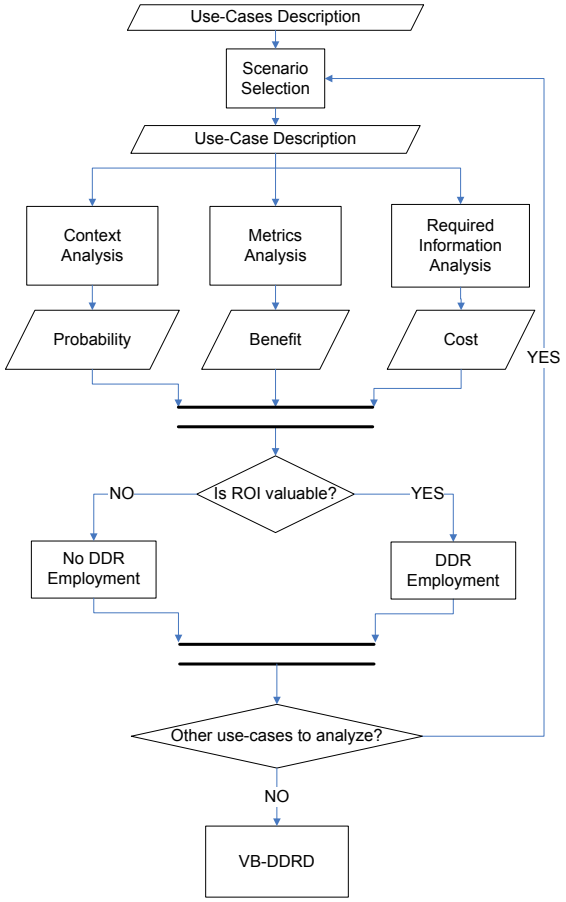

Figure 1: Activity and information flows for the proposed Value-Based Rational Documentation process.

Lack of motivation: The clear definition of who will profit from who allows the existence of a role (performed by real person or virtually) in charge of controlling that the specific producers provide, and the relate consumers use, the expected DDRD.

Unclear benefit: The clear definition of which advantages are achieved by enacting which scenario (DDRD UC) mitigates misunderstandings about pros and cons.

Information unpredictability: despite the DDRD producer cannot perfectly estimate which information the consumer will require; in this work we provide some data to do that (see).

Maturity: the relationship between DDRD information and DDRD UC provides a new and promising tactics to increase maturity of DDRD.

It is rather evident that such a process to document design decision rationale will address and mitigate to some respects most of the DDRD inhibitors. However, we are still left to validate its key assumption that different DDRD use-cases require different DDRD information.

4. EMPIRICAL FEASIBILITY STUDY

4.1 Experiment Process Description

4.1.1 Research objectives. As discussed just above, we do not imagine our VB DDRD to be a panacea, but we expect to obtain both advantages and disadvantages in adopting it. In the absence of valid formal models for our VB DDRD, we are unable to compute pros and cons of our approach. Our decision is hence to proceed empirically, in particular to conduct controlled experiments, with the goal of investigating the feasibility of our approach rather than comparing it with other ones. We preferred a feasibility study to a comparison because our method provides what other methods do not provide: a realistic means for mitigating all the known DDRD inhibitors. Additionally, it seemed that evaluating our process advantages (i.e., the amount of mitigation on DDRD inhibitors) is not an interesting direction, for the following reasons:

- **Evident advantages**: it seems obvious, for instance, that writing a subset of information (tailored DDRD) requires less effort than writing the information in full (a complete DDRD).
- **Difficult empirical validations**: it is hard, for instance, to measure how the personal interests can be mitigated by the presence of a manager who checks the DDRD development.

The goal of our study, in the sense given by Basili [14], is *to analyze* the DDRD *for the purpose of* evaluation *with respect to* the perceived utility *from the point of view of the* researcher *in the context of* post-graduate Master students of software engineering. The aims of this empirical study is both explorative and confirmative: i) to discover the perceived levels of importance related to each category of DDRD information for enacting different DDRD use cases, ii) to confirm the hypothesis that the level of importance related to the same category of DDRD information is different for different DDRD UC(s). The latter is the key principle of our proposed VB DDRD, without which our VB DDRD would not work.

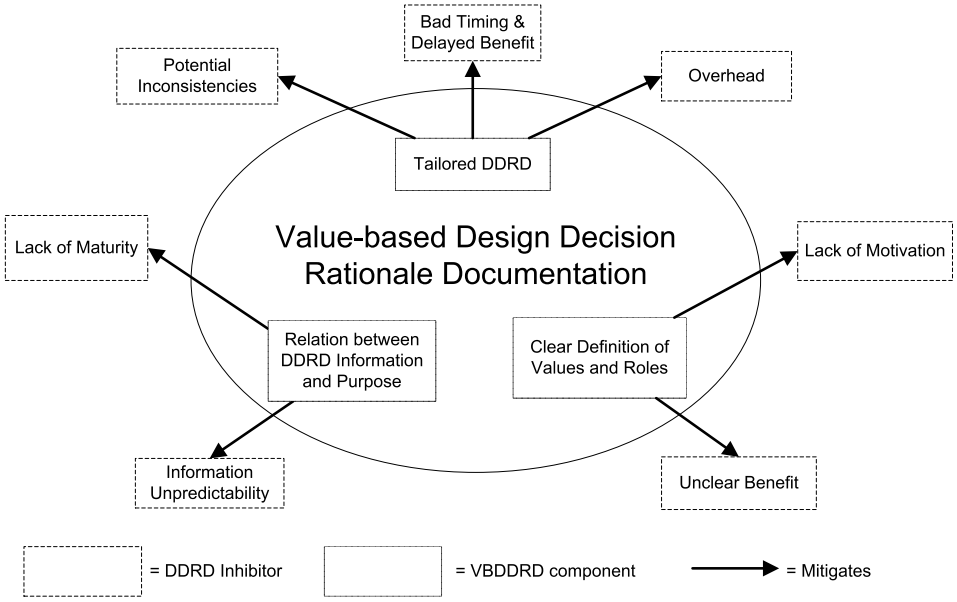

Figure 2: Expected effects on DDRD inhibitors

4.1.2 Hypotheses. In order to investigate the assumption that the levels of importance related to categories of DDRD information are different for different DDRD UC(s), we derive the following null hypotheses (respectively alternative hypotheses) for the present study. When enacting DDRD UC(s) with ID(s) *i* and *j*, there is no significant difference ($H_{0ij\text{-}}$) (resp. there is significant difference, $H_{1ij\text{-}}$) between the levels of perceived importance related to the category of DDRD information *x* ($H_{\_x}$). Notice how *i* and *j* can be any DDRD UC(s); in the present study we used the DDRD UC(s) described in Table 1.

We highlight that the assumption that the levels of importance related to the same category of DDRD information are different for different DDRD UC(s) is valid only if the above null hypothesis is rejected for one or more combinations of *i* and *j*. In particular, each specific combination of *i* and *j* in which the null hypothesis is rejected, identifies a specific pattern in the perceived importance level of a category of DDRD information.

4.1.3 Variables. The DDRD UC is the experiment factor. The experiment takes into account five DDRD UC(s) (see Table 1); they represent five levels of this independent variable (i.e. treatments). As dependent variables, we used the utility related to each specific category of DDRD information as perceived by subjects for enacting a specific DDRD UC. Participating subjects expressed quantitative measures of the utility of a specific category of DDRD by a 3-point ordinal scale (useless, optional, or required). We used an array of thirteen categories of DDRD as proposed by Tyree and Akerman [3]. We controlled at a constant level the remaining independent variables such as experience of the participating subjects, experiment materials, environment, and complexity of the experiment objects. We also blocked a further independent variable, the type of documentation, because we were not interested in investigating that dimension, as explained in the remaining sections.

4.1.4 Subjects. Fifty students at the last year of their Master Degree in something similar to computer engineering, at the University of Rome "Tor Vergata", participated in our work as experiment subjects. While most of our subjects had already had some experiences in software companies, only few of them can be considered as software professionals. Hence, in order to gain on external validity, we modeled five different roles or types of stakeholders: authentication, human interface, operative system, communication protocol, and data storage. Then, subjects expressed their preference for each role, according to their previous experience and level of confidence with the responsibilities of a role; it was done well in advance of the last training session. Afterwards, we assigned roles to subjects by trying to maximize the coverage of their expressed preferences (i.e., experience).

4.1.5 Design. In order to analyze the relationship, if any, between the utility of DDRD information categories and the DDRD purposes, we selected five DDRD use-cases to investigate (see Table 1) and five decisions for each role. We adopted the DDRD UC(s) reported in Table 1 among the ones available from the literature [15] with the aim to minimize the validity threats, according to our context (e.g., available time,

subjects experience in executing the use cases, experimenters ability in providing fine replica objects).

Each experiment decision consists in a DDRD-documented decision already made in the past by a decision-maker who virtually left, and a new set of requirements. Since a decision (and its DDRD) is compatible to all five DDRD use cases, we utilized all the twenty-five decisions with each DDRD UC.

We utilized two types of documentation (i.e. tailored or complete) because the level of perceived utility, related to a category of DDRD information (i.e,. expected utility), may change based on the presence or absence of enough documentation for such category. Because we are not interested to investigate the impact of such a factor, we blocked the experiment with respect to that variable [16].

In order to limit the occurrences and mitigate the influence of several threats to validity, we balanced the experiment design in this way:

- Each decision has been adopted the same number of times.
- Each treatment (on each decision) has been applied the same number of times.
- Each subject applied all DDRD UC (on different decisions).
- All treatments (i.e. DDRD UC(s)) have been applied the same number of time (i.e. 10) in all the available orders (i.e. as first, second, third, fourth, and fifth).
- Each subject applied only one time each of the five DDRD UC(s); each subject encountered just one time all the five decisions belonging to his specific competence.

4.1.6 Experimental Material and Tasks. In order to replicate the context of the real world, we used a synthetic software project, which is quite similar to another experimental object we had already been using successfully [8]. The experiment project was concerned with a public transportation system characterized by ambient intelligent issues (e.g., resource constraints, heterogeneous sensors, etc.) [17]. It provided the possibility both to derive five decisions regarding five different competences (i.e., roles), and to provide valid experiment objects (i.e., good replica), as for a previous experiment we conducted [8].

The concepts related to software architecture were not covered by a single role but all the roles concerns decisions related to it. As a matter of fact, the adopted decisions are inter related to each other, for examples:

- The decision related to 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 from the type of DMBS, the communication protocol, and the architectural pattern (e.g., MVC).

Concerning the requirements change, we adopted causes like: 1) Variations in the industrial strategic partnerships; 2) Changes of customer requests resulting from his experience in using the previous version of the product; 3) Technology advances.

Each subject received the materials containing (1) the experiment rules, (2) the system main characteristics, (3) five different decisions (with related DDRD and new requirements), (4) a description of which DDRD UC enacts on which decision, and in which order, (5) the form to fill data in. In particular for each of the five DDRD UC(s) to enact, subjects had to exec the following steps: 1) to understand the current DDRD UC to enact, 2) to enter the form with the current time (initial time), 3) to read the DDRD related to a specific decision, 4) to enact the DDRD UC (write the requested answer), 5) to write the final time on the form, 6) to describe the level of perceived utility for each category of DDRD information.

4.1.7 Execution Preparation. We've gained experience over several years in conducting controlled experiments similar to the one in this paper, which took into consideration: type of objects (DDRD, ambient intelligence domain, paper-based), subjects (class size, experience, expectation), and context (laboratory). Such an experience helped us in: (1) designing and implementing the experiment objects, (2) settings the experiment lab, (3) motivating the students, (4) training the participating subjects. In particular, regarding the training phase (a) we choose a time duration of five hours (three sessions), (b) we avoided to use terms which we had experienced to be source of misunderstanding, (c) we corrected the students' wrong assumptions and expectations regarding the experiment, and (d) we carefully distilled information regarding the experiment, by clearly explaining almost all the experiment related attributes less the experimenters' expectations.

4.2 Experiment Result Description

4.2.1 Data Analysis

4.2.1.1 Descriptive Statistics. In general, the more a category of DDRD information is perceived as "Required", the more it is valuable for DDRD consumers. Table 2 shows the mean and the variance of the subjects who perceived a specific category of DDRD information as required while enacting the DDRD UC(s). Hence, Table 2 helps us to understand which parts of the documentation are interesting for the readers in case they have to enact all the DDRD UC(s). This result is similar to [12].

Table 2: Percentage of subjects that felt as "required" a specific category of DDRD information

DDRD Information	Mean (%)	Variance
Issue	91	25
Decision	79	110
Status	25	318
Assumptions	49	117
Constraints	54	54
Positions	72	43
Argument	67	45
Implications	21	28
Related decisions	28	104
Related requirements	74	150
Related artifacts	14	8
Related principles	21	60
Notes	5	9

Figure 2 shows the percentage of subjects who, when they finished executing a specific DDRD UC, marked a specific category of DDRD information as "Required". Figure 2 supports our understanding of the relationship, if any, between the importance of a specific category of DDRD information and specific DDRD UC(s). In other words, DDRD producers can use Figure 2 as a reference to choose whether to neglect or to include any category of DDRD information, in the DDRD to be provided, according to the DDRD UC(s) to enact. In our best knowledge, the results presented in are completely novel; not only in relation to the DDRD but also to any other type of documentation (i.e., we did not found any study investigating the amount of perceived utility related to a documentation category, for specific purposes). Finally, seems to confirm that the level of perceived utility does change both in relation to the category of DDRD information, and the DDRD UC enacted.

4.2.1.2 Hypothesis Testing For each category of DDRD information, in order to test the null hypothesis (the level of the perceived utility insignificantly depends on the DDRD UC), we used the Kruskal-Wall test on all the five DDRD UC(s). In case it was possible to reject (p-value < 0.05) that null hypothesis for a category of DDRD information, then we used the Mann-Whitney for testing all the possible combinations of couples of DDRD UC(s) with respect to such a category of DDRD information. Table 3 describes the results of the statistical test on the level of perceived utility related to a specific category of DDRD for enacting a specific combination of DDRD UC(s). Hence, Table 3 describes the results of applying these tests; "Yes" means a statistical significant difference (p-value < 0.05) in the level of perceived utility for enacting specific DDRD UC(s) (all the adopted DDRD UC(s) in the second column, a combination of two DDRD UC(s) in the others) regarding a specific category of DDRD information (as identified by the first column).

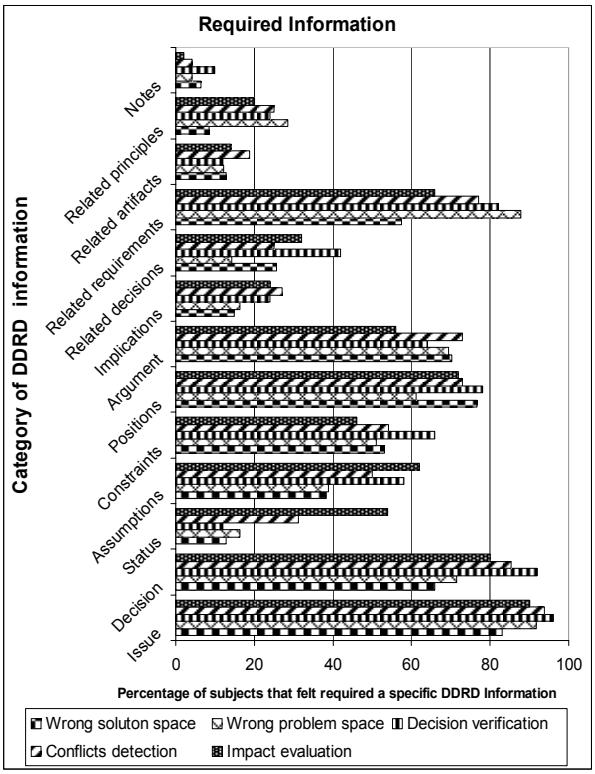

Figure 3: Percentage of subjects that felt as "required" a specific category of DDRD information for enacting a specific DDRD UC.

Table 3: Statistical significance (Yes)/insignificance (No) in the difference of the level of perceived utility related to a category of DDRD for enacting a specific combination of DDRD UC.

DDRD Information	All 5 DDRD UC	DDRD UC ID Combination									
		1-2	1-3	1-4	1-5	2-3	2-4	2-5	3-4	3-5	4-5
Issue	No	No	No	No	No	No	No	No	No	No	No
Decision	Yes	No	Yes	Yes	No	Yes	No	No	No	No	No
Status	Yes	No	No	No	Yes	No	No	Yes	No	Yes	Yes
Assumptions	Yes	No	Yes	No	Yes	Yes	No	Yes	No	No	No
Constraints	No	No	No	No	No	No	No	No	No	No	No
Positions	No	No	No	No	No	No	No	No	No	No	No
Argument	No	No	No	No	No	No	No	No	No	No	No
Implications	No	No	No	No	No	No	No	No	No	No	No
Related decisions	Yes	No	Yes	No	No	Yes	No	No	Yes	No	No
Related requirements	Yes	Yes	Yes	Yes	No	No	No	Yes	No	Yes	No
Related artifacts	No	No	No	No	No	No	No	No	No	No	No
Related principles	No	No	No	No	No	No	No	No	No	No	No
Notes	No	No	No	No	No	No	No	No	No	No	No

4.2.2 Result Interpretation We briefly discuss the results related to single category of DDRD Information. Note that measurements made in the experiment are repeatable in real-world projects; if you disagree with these results, you could repeat then the measurements in your own organization.

Issue. According to our expectations, results show that this category has been perceived extremely useful. Obviously, the description of the issue to be addressed by the decision is key whatever the DDRD UC might be. In particular, by observing Table 2 we noticed that such a category have been perceived quite always (i.e. 91% of the time) as "required". Its low variance (i.e. 25) reveals that its huge importance is shared among all the DDRD UC(s). Such a result of generalizability is confirmed by Table 3 because its importance does not significantly change in any DDRD UC(s).

Decision. This information has been felt quite useful by subject. In particular it has been required always more than the 60% of the times (see Table 3). However, its importance is relatively low in case it has used to check wrong solution/problem space. Such a result can be explained by taking the example in which the category related requirements describe the need for high security while the category arguments includes low security. This mismatch is already evident without the further information provided by the category decision.

Status. In theory the "status" of the decisions is really important in order to evaluate the impact of a change while it is not important for other considered DDRD UC(s), such as for example the identification of erroneous requirement (i.e. DDRD UC with ID 2). This is confirmed by and by observing the low importance in the average (i.e. 25%) and the huge variance among different DDRD UC(s) (i.e. 318).

Assumptions. By analyzing Table 2 we observe that only the half of the time such information has been perceived as required. In particular, by observing we noticed how its utility is lower in case it has used to check wrong solution/problem space. Again, this may be explained by the fact that for such DDRD UC(S) it is more important to know the related requirements and the arguments.

Constraints. Again, only the half of the time such information has been perceived as required. However, based on Table 3 such an information seems not to change its level of importance according to the different purposes of the documentation.

Positions. The description of the alternatives considered for making a particular decision has being perceived in general quite useful (i.e., 73% of times) for enacting all the DDRD UC(s) as confirmed by the absence of significant difference among any different DDRD UC(s).

Arguments. The information related to the reasons for making a particular decisions has being perceived in general quite useful (i.e., 73% of times) for enacting all the DDRD UC(s) as confirmed by the absence of significant difference among any different DDRD UC(s). In particular its utility is higher than expected by us in case it has used to estimate the impact of a new decision.

Related decisions. The information regarding related decisions has been perceived quite useless (i.e,. 28% of times). We expect that such an information will became the most important one when it will be well supported by standardized tools able to provide fast navigation among several DDRD(s) and to process their relationships.

Related requirements. The description of the project requirements related to the decision to re-design has been perceived quite useful (i.e., 74%). However its importance significantly change based on the purpose of the documentation. In particular, Table 2 validate our expectation that in order to understand if the decision maker had a good knowledge on the solution space it is not useful to know the related requirements but just the arguments that led to that decision.

Notes. For the sake of completeness, any documentation framework includes a field, like our Notes, to allow the users to insert information, if any felt as missing and due. Based on , our subjects considered almost useless the category "Notes" of information (i.e., only 5% of the times). Such a result suggests that the proposed DDRD framework [3] fits well the decisions and the DDRD UC(s) that have been adopted in this empirical study.

By analyzing Table 3 horizontally, we find that five DDRD information categories (Decision, Status, Assumption, Related decision, Related requirement) show significant statistical difference in the perceived utility level while enacting different DDRD UC(s). In other words, the utility of such categories of DDRD information depends on the DDRD UC to enact. By analyzing Table 3 in a vertical way we find that each possible combination of DDRD UC differs in the level of perceived utility for at least one category of DDRD information, despite the combination of DDRD UC with ID 2 and 4. In other words, according to our results, whether DDRD is tailored on the basis of the "expected" utility level, the DDRD UC(s) with ID 2 and 4 would have the same DDRD, while all the other DDRD UC(s) would have a different tailored DDRD (i.e. DDRD with different included and/or omitted categories of information). Note that Table 3 describes an insignificant difference among DDRD UC(s) with ID 2 and 4; this result is completely different from defining them as identical: as a matter of fact, for instance, the columns 1-4 and 2-4 are different.

To conclude, both descriptive and statistical results confirm our expectations that the level of utility related to the same category of DDRD information significantly changes according to the DDRD UC. Therefore, the DDRD consumer requires, or not, a

specific category of DDRD information according to the DDRD UC to enact. Consequently, results suggest that the DDRD producer can tailor the documentation by including only the information required for the DDRD UC(s) that are expected to be enacted.

According to the experiment results, our proposed VB-DDRD would provide an effort saving of the 50% in the average (among DDRD UC(s)) by supposing that i)all categories require the same effort and, ii) a category is included in the documentation only in case its level of perceived utility is higher than 50%.

5. CONCLUSION AND FUTURE WORK

Older Design Decisions Rationale Documentation (DDRD) methods aimed at maximizing the DDRD consumer benefits by forcing the DDRD producer to document all the potential useful information; they eventually ran into too many inhibitors to be used in practice. In this paper we propose a value-based approach for documenting the reasons behind design decision (VB DDRD), based on *a priori* understanding of who will benefit later on, from what set of information, and in which amount. Such VB DDRD offers means to mitigate all the known DDRD inhibitors and it is based on the hypothesis that the set of required DDRD information depends on the DDRD use case (DDRD UC) to enact. In order to validate such a hypothesis we ran an experiment in a controlled environment, employing fifty subjects, twenty-five decisions, five different DDRD UC(s), and 250 DDRD UC(s) executions. Each subjects practically used the documentation to enact all the five Use Case(s) by providing an answer and a level of utility for each category of DDRD. Both descriptive and statistical results confirm our expectancies that the level of utility, related to the same category of DDRD information, significantly changes according to the DDRD UC. Such result is novel and imply that the DDRD consumer requires, or not, a specific category of DDRD information according to the DDRD UC to enact. Consequently, results suggest that the DDRD producer can tailor the documentation by including only the information required for the DDRD UC(s) that are expected to be enacted (i.e. valuable). This result demonstrates the feasibility of our proposed VB DDRD.

6. References

[1] S. Biffl, A. Aurum, B. Bohem, H. Erdogmus, and P. Grünbacher, *Value-Based Software Engineering*: Springer, 2005.

[2] J. Lee, "Design Rationale Systems: Understanding the Issues," *IEEE Expert: Intelligent Systems and Their Applications*, vol. 12(3), pp. 78-85, 1997.

[3] J. Tyree and A. Akerman, "Architecture Decisions: Demystifying Architecture," *IEEE Software*, vol. 22(2), pp. 19-27, 2005.

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

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

[6] 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: ACM Press, 1996.

[7] L. Bratthall, E. Johansson, and B. Regnell, "Is a Design Rationale Vital when Predicting Change Impact? A Controlled Experiment on Software Architecture Evolution," presented at International conference on product focused software process improvement, Oulu , Finland, 2000

[8] D. Falessi, G. Cantone, and M. Becker, "Documenting Design Decision Rationale to Improve Individual and Team Design Decision Making: An Experimental Evaluation," presented at International Symposium on Empirical Software Engineering, Rio De Janeiro, Brazil, 2006

[9] B. Shum and N. Hammond, "Argumentation-Based Design Rationale: What Use at What Cost?," *International Journal of Human-Computer Studies*, vol. 40(4), pp. 603 - 652 1994.

[10] M. Heindl and S. Biffl, "A case study on value-based requirements tracing," in *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering*. Lisbon, Portugal: ACM Press, 2005.

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

[12] A. Tang, M. A. Babar, I. Gorton, and J. Han, "A Survey of Architecture Design Rationale," *Journal of Systems & Software*, vol. 79(12), pp. 1792-1804 2007.

[13] I. Jacobson, M. Griss, and P. Jonsson, *Software Reuse: Architecture, Process and Organization for Business Success*. Reading, MA: Addison-Wesley, 1997.

[14] V. Basili, G. Caldiera, and D. Rombach, "Goal/Question/Metric Paradigm," *Encyclopedia of Software Engineering*, vol. 1(John Wiley & Sons), pp. 528-532, 1994.

[15] P. Kruchten, P. Lago, H. van Vliet, and T. Wolf, "Building up and Exploiting Architectural Knowledge," 2005

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

[17] P. Remagnino, G. L. Foresti, and T. Ellis, *Ambient intelligence : a novel paradigm*. New York: Springer, 2005.