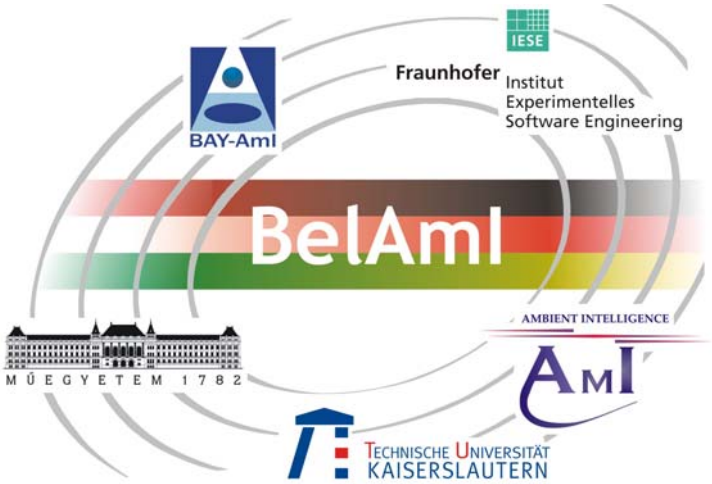
Fraunhofer Institut Experimentelles Software Engineering

Documenting Design Decisions: A Framework and its Evaluation in the Ambient Intelligence Domain



Authors:
Davide Falessi
Martin Becker

BelAmI Report No. 005.06/E
IESE-Report No. 050.06/E

Date: 2.3.06
Version: 1.0
Status: Final
Classification: Internal

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft.
The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competitive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach (Executive Director)
Prof. Dr. Peter Liggesmeyer (Director)
Sauerwiesen 6
67661 Kaiserslautern

Abstract

Design decisions crucially influence the success of every software development project. Consequently, a considerable amount of time in the development activities is usually spent to discuss design decisions and their possible outcomes. While the results of the design decisions are often documented quite well, the situation is usually different for the decision making processes themselves. Usually, little or no effort is spent to track the decision process and its findings, although this could be relevant and helpful information for future decisions in related projects.

This report deals with the question, how to document design decisions in software development projects in general and within the demonstrator development in the BelAmi project [1] in special. Usually design decisions have to consider the interests of various stakeholders. This holds especially in the Ambient Intelligence domain. Consequently, a design documentation activity focusing on the avoidance of misunderstandings and shortcomings in the stakeholders' intercommunication is, in our context, mandatory.

The aim of the presented work in this report is to provide a framework for documenting design decisions in a sound way and to analyze recurring patterns in decision making, i.e. importance, location, and conflicts of quality attributes.

Keywords: BelAmI, design rationale, case study, design, software development.

Table of Contents

1 Introduction

1.1 Context and Objectives

Design decisions crucially influence the success of every software development project. Consequently, a considerable amount of time in the development activities is usually spent to discuss design decisions and their possible outcomes. While the results of the design decisions are often documented quite well, the situation is usually different for the decision making processes themselves. Usually, little or no effort is spent to track the decision process and its findings, although this could be relevant and helpful information for future decisions in related projects.

This report deals with the question, how to document design decisions in software development projects in general and within the demonstrator development in the BelAmi project [1] in special. Usually design decisions have to consider the interests of various stakeholders. This holds especially in the Ambient Intelligence domain. Consequently, a design documentation activity focusing on the avoidance of misunderstandings and shortcomings in the stakeholders' intercommunication is, in our context, mandatory.

The aim of the presented work in this report is to provide a framework for documenting design decisions in a sound way and to analyze recurring patterns in decision making, i.e. importance, location, and conflicts of quality attributes.

1.2 Outline

This report is structured as follows: Chapter 1 introduces into the report.

To set the scene, chapter 2 presents the background, related concepts and existing work in the area of design decisions. Chapter 2.1 introduces the concept of design decision in general. Chapter 2.1.1 describes the problem of design decisions in the software context. Chapter 2.1.2 discusses how software design decisions can be analyzed by means of design spaces. Chapter 2.1.3 presents major techniques applied to help software developers (architects, designers, stakeholder, etc.) while making decisions. Chapter 2.2 introduces the concept of design rationale and possible usage scenarios of it. Chapter 2.3 elaborates on the context of the presented work: the domain of Ambient Intelligence Systems and the BelAmI project [1].

Chapter 3 motivates the need for capturing design decision rationale and identifies reasonable approaches to this end. Chapter 3.2 describes which and why patterns in decision making are important to be figured out.

Chapter 4 describes the approach developed in the BelAmI context.

Chapter 5 describes how this approach has been applied in a development project. This comprises the tools, method, and metrics used to document the design decisions.

Chapter 6 presents the results and comments of a conducted case study. In particular chapter 6.1 shows data achieved by applying the proposed method. Chapter 6.2 comments patterns in the made decisions.

Chapter 7 outlines ideas for future work. In particular chapter 7.1 presents an idea of empirically improving the capturing of design decision rationale. Chapter 7.2 presents an idea of empirically improving the CBAM [13] method to make decisions related to non-functional requirements.

2 Related Concepts and Existing Work

2.1 Related Concepts

2.1.1 Design Decision

As different notions are associated with design decision, it is appropriate to clarify the definitions applied in this document. A *design decision* is “a selection of an option among zero or more known and unknown options concerning the implementation, structure, interaction and usability of a software application” [20]. An architectural decision is a type of design decision, which can be defined as “activities related to deployment issues of a software application. Architectural design addresses the question: How are we going to implement this software application. It differs from implementation design in that the implementation constructs considered are of a much broader scope” [20]. A considerable amount of work has already been conducted concerning the documentation of architectural decisions. One the most important has been made by the SEI [2], which provides an outline for documenting architectures based on Kruchten’s 4+1 views [3]. In our best understanding, the difference between design decisions and architectural decisions is not so obvious and usually the terms are used as synonyms. In the following, we do not distinguish between these two types of decisions.

2.1.2 Design Space Representation

The concept of multi-dimensional design spaces [30] provides a useful means to describe and classify system architectures [4]. In this kind of representation, each dimension of a design space describes variation in one system characteristic or design choice. Categories or values correspond to design alternatives. A specific system design can be represented by a point in the design space that unambiguously identifies the dimensional values that correspond to its characteristics and structure. Dimensions of a design space can be of type continuous or discrete. However, they have to be ordered using metrics. One of the main problems in developing design spaces is to find the right granularity of classification. One method to increase the granularity of a dimension is to partition its values (continuous or discrete) into a smaller amount of discrete values (e.g., "low," "medium," "high"). This approach is considered useful, when the description provides too much information to be used effectively.

2.1.3 The Problem of Decision Making

As described previously, every alternative of a decision can be analyzed independently from the application domain by observing how it performs on several attributes (e.g. performance, security, reusability). In literature, the adjective “dominated” is used to indicate that an alternative is worse (dominated) wrt. all values of the considered attributes compared to at least another alternative. At the contrary, an alternative is considered to be of the type “pareto-optimal”, when there is no alternative, which has a better value in respect to any considered attribute. A trade-off can be defined as “a balancing of two opposing situations or qualities, both of which are desired” [26]. In other words, a trade-off is made during the selection of an alternative from the spectrum of the pareto-optimal ones.

Several approaches exist to figure out the spectrum of pareto-optimal alternatives for a certain decision systematically. The Multiple Criteria Analysis (MCA) [5] consists of a family of methods, which allow the evaluation of several alternatives based on a limited number of criteria (attributes) achieving a unique global judgment (score). The Multiple Criteria Decision Making (MCDM) can be defined as a decision making process, which suggests the best decision by ranking the spectrum of available alternatives based on multiple objectives (attributes) to achieve. Sometimes, comparing several objectives can become quite complicated, if they are too heterogeneous to be measured using different metrics (e.g. euro and seconds) of the same type (in the previous example quantitative), or also using different metrics (e.g. seconds and high reusability) of different types (quantitative and qualitative). The MCDM problems can be divided in two areas:

- Multi Attribute Decision Making (MADM) [6] provides the selection among a numerable amount of alternatives, where the related values for the considered objectives are discrete. In other words, you know what you want, and you know the spectrum of available alternatives and their relative scores (for every objective).
- Multi Objective Decision Making (MODM) [7] provides the design of the best alternative among an infinite number of them, which are implicitly expressed (e.g. by boundaries). One of the most famous techniques to address a specific type of this kind of problem, is the linear programming method [27].

Software development is traditionally driven by functional requirements. However, non-functional requirements (NFR), such as accuracy, security, and performance, are often as crucial to the success of the system as the functional requirements. In attempting to address one type of requirements systematically, (e.g. security), it is very hard to be systematic in meeting all the other requirements at the same time (e.g. performance) due to inherent trade-offs. The importance of NFR is widely recognized. Several approaches to engineer the required qualities into software in a systematic way already

exist. However, these approaches can be considered as incomplete, in the sense that they do not cover all qualities. In the following part, we briefly describe standard approaches to select among software design alternatives for achieving the desired NFR.

1. NFR-framework [8]: L. Chung et al. use “soft goals” as the basis of their usage of quality attributes. A soft goal is a goal with no clear-cut definition and/or criteria as to whether it is satisfied or not. They characterize quality attribute goals as soft goals and develop a process that involves viewing a design as a point on a quality attribute space. The following figure shows an example of the proposed formalism to address the quality attributes modifiability and performance using the techniques ”shared data” and “abstract data types”.

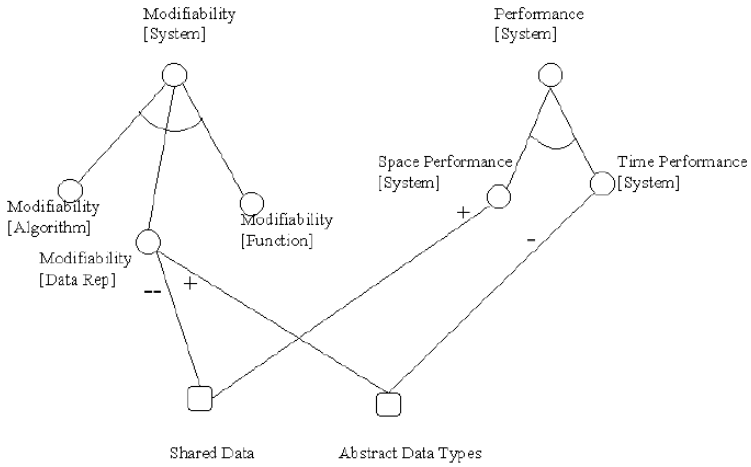


Figure 1: An example of achieving non-functional requirements via the visualization of alternative score in the NFR-tool.

2. House of Quality: Yoji Akao defined Quality Function Deployment as "a method for developing a design quality aimed at satisfying the consumer and then translating the consumer's demands into design targets and major quality assurance points to be used throughout the production phase" [31]. The most familiar form of QFD is the House of Quality that captures the customer's requirement (how), the corresponding technical requirements (what), interrelationships, correlations, priorities and benchmarks. The following figure shows the House of Quality matrix.

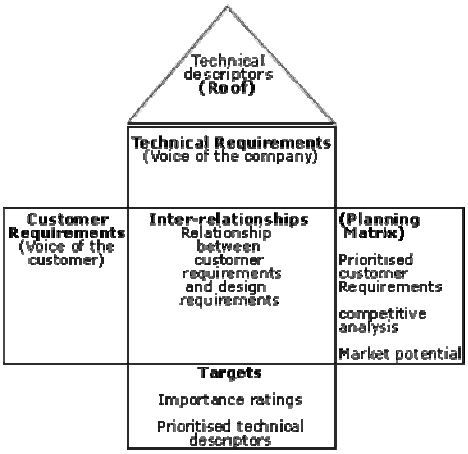


Figure 2: The house of quality matrix.

The following figure shows an example of use of House of Quality.

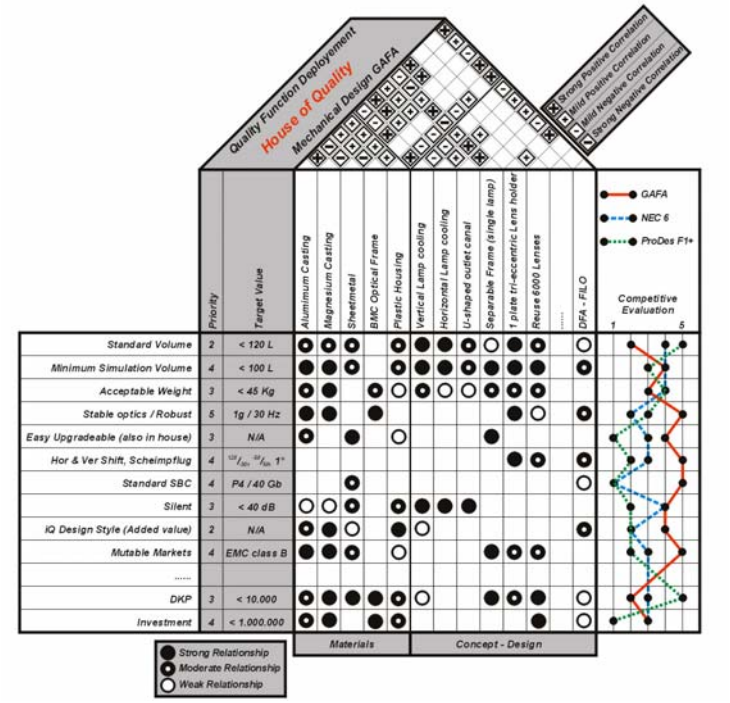


Figure 3: An example of use of the house of quality matrix.

3. Attribute Driven Design [9]: L. Bass et al. at the SEI developed with the Attribute Driven Design (ADD) method an approach to define a software architecture by basing the design process on the quality attributes that the software has to fulfil. Their approach is a recursive decomposition process where, at each stage in the decomposition, attribute primitives are chosen to satisfy a set of quality scenarios and then functionality is allo-

cated to instantiate the component and connector types provided by the primitives. The main characteristic of this approach with respect to the common ones is that it is focuses almost completely on NFR instead of functional requirements.

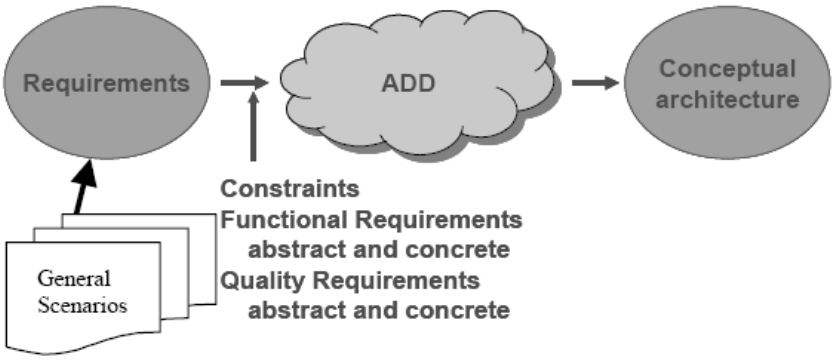


Figure 4: ADD in life cycle.

4. CBAM: the Cost Benefit Analysis Method [13] has been developed by the Software Engineering Institute and the Carnegie Mellon University. They suppose that quality attributes are, in most cases, dictated by architectural design decisions and that the benefits of a software system are assessable only relative to the business goals to be achieved with a system. Consequently, software architecture is the crucial artefact to study trade-offs and perform cost-benefit analyses in the design activities. Their model provides an architecture-centric approach to economic modelling of software design decision making, in which costs and benefits are traded off within the different system quality attributes. The CBAM consists of six main steps: 1. choosing scenarios and architectural strategies, 2. assessing QA benefits, 3. quantifying the architectural strategies' benefits, 4. quantifying the architectural strategies' costs and schedule implications, 5. calculate desirability, 6. make decisions. In few words, every quality attributes is scored with a number based on its importance, then, every available alternative is scored with four (the number of QA) number between 0 and 1 based on how it fulfils quality attributes. As a final step, the desirability of an alternative is computed by summing the multiplication of the importance of the QA with the level of fulfilment of the current alternative. The benefits of an alternative are computed as its desirability divided by the relative cost.

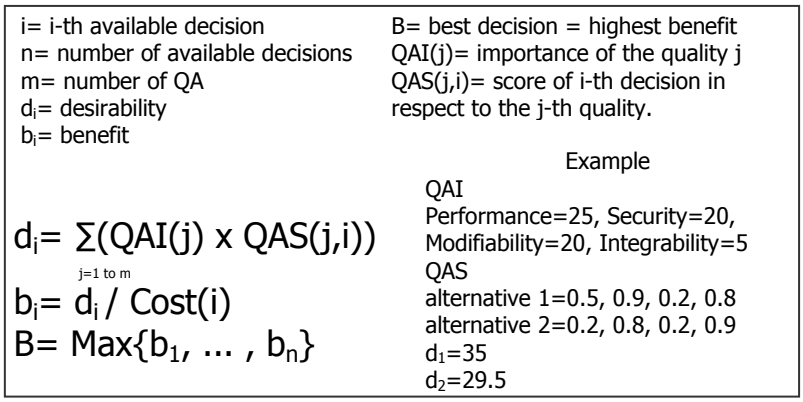


Figure 5: The Cost Benefits Analysis Method.

A general approach to solve trade-offs in achieving NFR is to prioritize them. As a matter of fact, it is very common that objectives in the requirements of a software system are contrary to each other. In other words, the fulfilment of the entire spectrum of software requirements is very difficult and usually too expensive to achieve.

1. The simplest way to decide which requirements are better to address, is to assign for each requirement a value of importance. This value can be qualitative (low, medium, high, must, should, may) or quantitative (weight) [12]. Afterwards, the objectives are achieved from the most important ones to the less ones until the available effort finishes.
2. Barry Boehm et al. developed the Win-Win Negotiation Model [11]. It provides a systematic way for identifying and resolving requirements conflicts by eliciting and negotiating artefacts such as win conditions, issues, options, and agreements. This model tries to "Make everyone a winner", and to generate a stakeholder win-win situation incrementally through the Spiral Model. The Win-Win negotiation tool is a Unix workstation-based groupware support system that allows stakeholders to enter win conditions, explore their interactions, and negotiate mutual agreements on the specifics of the new project being contracted. The model and support system also feature a central role for quantitative trade-off analysis tools such as COCOMO. Such a tool provides the extraordinary functionality of handling requirements by dealing not only with the importance of the requirements itself, but also on the characteristics of the technique to realize it, and its relations with other techniques to realize the other requirements. Requirements can be preferred using the following order: positively influenced (achieving one implies simplify the achievement of the other), independent (achieving one implies a null influence to the achievement of the other) and in conflict (achieving one implies complicates the achievement of the other). In this approach a requirement is refined and classified as a win-win case, when several stakeholders are satisfied by achieving them. The following figure shows the Win-Win spiral model main phases.

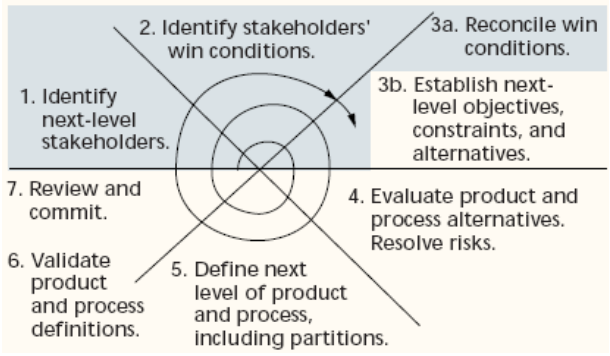


Figure 6: Phases in Win-Win spiral models. The blue area represents front-end activities, where objectives, constraints, and alternatives are derived from.

3. Previously, we highlighted that the relations among methods to achieve NFR play an important role in the decision making activity. Therefore it is reasonable to mention also the work made by H. Eguiluz and M. Barbacci, [10] here, which describes interactions among techniques addressing quality attributes quite well. Similar to the Attribute Drive Design method [9], where each quality attribute has its own set of techniques to address it, Eguiluz and Barbacci believe in the existence of a set of methods, to directly promote a certain quality attribute (e.g. cryptography → security, process replication → performance, testing → dependability, separation of concerns → modifiability). In their work, the techniques promoting each one have been selected and categorized in relation to quality attributes as promotion, detection, or correction. For each category, matrices are presented that provide a detailed description of why a particular interaction is positive, negative, or neutral, or cannot be determined without assessing a concrete system. A positive relation means that the two techniques are compatible to each other so that in using both of them the related quality attributes are addressed easily and with success. At the contrary, a negative relation means that the use of a technique negatively impacts the use of the other so that it is very hard to use the techniques together. The following figure shows the interactions among techniques in promoting quality attributes.

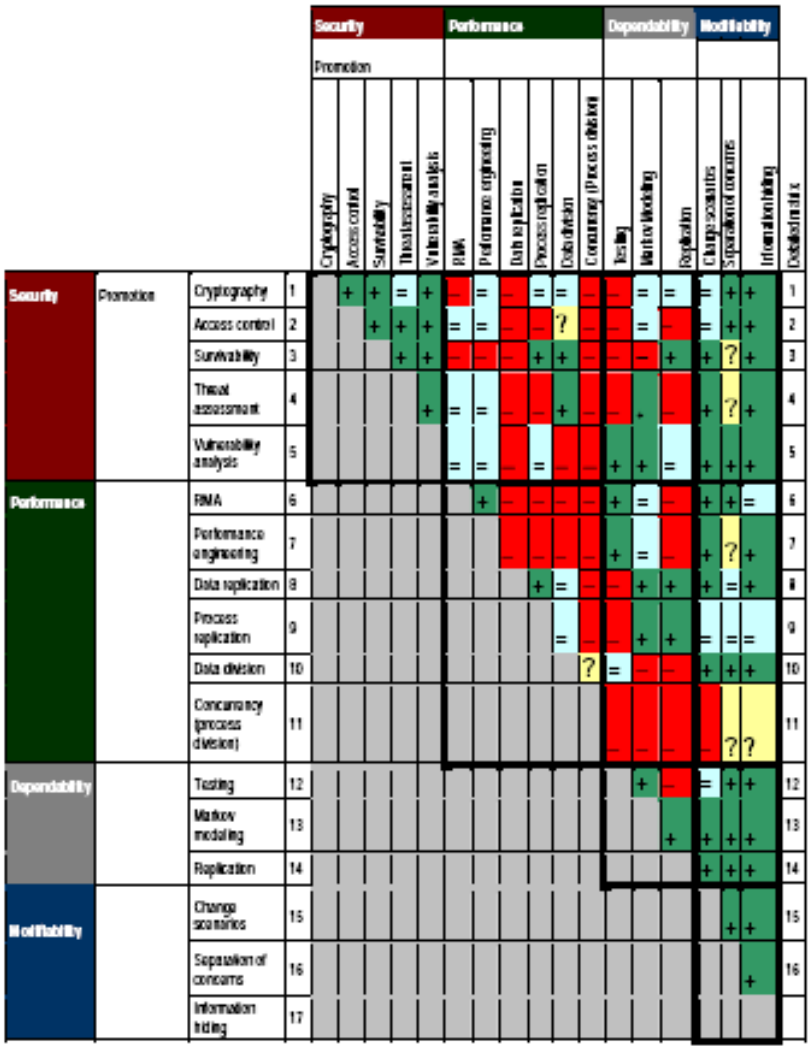


Figure 7: Interactions among techniques in promoting quality attributes according to [10]

2.2 Design Decision Rationale

2.2.1 Introduction

A design decision can be documented in many different ways. Design documentation ranges from formal design specifications, often following a rigorous standard imposed by an outside agency, to informal notes contained in the notebooks of the individual designers. There are many definitions of design rationale but the most appropriate in our context is provided by 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 the argumentation that led to the decision” [17]. Rationales can be classified into the following not mutually exclusive *types*:

- **Argumentation based**: the design rationale is primarily used to represent the arguments that define a design [14].
- **History based**: the rationale consists of the design history, the sequence of events that occurred while performing the design [14].
- **Device based**: a model of the device itself is used to both obtain and to present rationale [15]. The explanations of the design would be produced by using the model to simulate the behaviour of the device.
- **Process based**: the rational capture is integrated into the design process itself, which guides the format of the rationale. In Ganeshan, et. al. [16], the design description is modified only by changes to and refinements of the design objectives, thus capturing the rationale as part of the design process.
- **Active document based**: the rationale is pre-generated and stored in the system. In these systems, the designer creates the design and the rational system generates the rationale for it based on the system's stored knowledge. For each decision made, the system compares the decision made by the user with the decision that it would have made, based on its knowledge.

Design rationale can be used for the following several *purposes*:

- **Design verification**: verification if design meets the requirements and the designer's intent. This verification can occur at any point in the design process.
- **Design evaluation**: similar to the previous one, but the rationale is used to evaluate designs (and partial designs) and design choices relative to one another.
- **Design maintenance**: determination of what choices were made when performing the design in order to locate sources of design problems or to indicate where changes need to be made in order to modify the design.
- **Design reuse**: determination of which portions of the design can be reused and in some cases, suggest where and how it should be modified to meet a new set of requirements. It is especially important to let the designer know why the decisions were made. In some cases, what may seem like an inefficient solution may actually be critical to the design as a whole.
- **Design teaching**: assistance in teaching new personnel about the design. Besides providing insight into how it works, the rationale shows why each design choice was made. This conveys more information than a static description.
- **Design communication**: improvement of design both after and during the design process. By capturing the design choices and the reasons behind them, this information can be made available to others affected to the design to both give them insight into the design and allow them the opportunity to provide their input into the process [18]. It also can provide an efficient way to detect conflicts in the work of multiple designers and

shorten the review cycle [19]. By capturing the reasons behind design decisions, design rationale can be used to answer questions from design reviewers who need to know why a particular choice was made or why an expected choice was not made.

- **Design assistance**: assistance during the design process. The ability to verify and evaluate design choices allows the designer to view the results of their design decisions. Documenting the argumentation can perform several functions: it clarifies the arguments by encouraging designers to document the information and it can be evaluated to ensure that all issues are resolved and that alternatives are selected that meet the requirements without violating any.
- **Design documentation**: design documentation by offering a picture of the history of the design and reasons for the design choices as well as a view of the final product. If the rationale is stored in a computer-readable form, it can be used as part of a custom documentation generating system by allowing documentation to be generated from different perspectives and, in some systems, allowing the user to ask questions about the design. Some systems use the rationale to generate documentation aimed at different groups of people. A customer, for example, would require a different level of detail than a designer.

2.2.2 Methods to Capture Design Rationale

There are two general approaches to capture design rationale in an argument based manner. Both of them are based upon the entity-relationship paradigm:

- The Issue Based Information Systems (IBIS) deals with issues, positions, and arguments for which the emphasis is on recording the argumentation process for a single design [24].
- The Questions, Options, and Criteria (QOC) notation [25], for which assessments are relationships between options, criteria and arguments and they are used to conduct debate about the status of the entities and relationships.

2.3 Ambient Intelligence Domain

The work presented in this report was mainly driven by demands that arose during development of Ambient Intelligence solutions in the BelAmI project [1]. Ambient Intelligence (short: AmI) can be considered as the enabling technology for an emerging generation of systems, which provide their services in a flexible, transparent, and anticipative manner requiring minimal skills for human-computer interaction. Ambient Intelligence emphasizes on greater user-friendliness, support for human interactions, more efficient services support, and user-empowerment. In this vision, people will be surrounded by intelligent and intuitive interfaces embedded in everyday objects

around us and an environment recognizing and responding to the presence of individuals in an invisible way by year 2010.

Ambient Intelligence (AmI) is characterized by three main aspects:

- Ubiquitous computing
- Ubiquitous communication
- Intelligent user interfaces

Ubiquitous Computing means integration of microprocessors into everyday objects like furniture, clothing, white goods, toys, even paint. Ubiquitous Communication enables these objects to communicate with each other and the user by means of ad-hoc and wireless networking. An Intelligent User Interface enables the inhabitants of the AmI environment to control and interact with the environment in a natural (voice, gestures) and personalized way (preferences, context).

The development of AmI systems is a challenging inter-disciplinary task that demands many different skills and technologies and hence requires the cooperation of scientists from many different backgrounds.

2.3.1 General Characteristics and Constraints of AmI Systems

With AmI systems, computing will reach a new level of complexity, which poses challenging research efforts for their robust operation. The developers of such systems have to cope with the following peculiarities [1]:

- **AmI systems are embedded**. This means that they are an integral part of surrounding applications which they control. People using the application are usually unaware of the existence of embedded systems, i.e. an AmI system cannot be accessed by humans in the traditional way by a key board, mouse and screen.
- **AmI systems are mobile**. AmI systems are usually part of a moving application, e.g. a person, a car, a bicycle or a mobile robot. This poses the general problem of continuous electric power supply if the application requires non-stopping operation. It also poses the problem of network connectivity. AmI systems can never rely on a stable network platform with guaranteed connectivity to other network nodes. Instead, AmI nodes will form so called ad-hoc-networks, which are dynamically formed by all AmI nodes having direct or indirect connectivity to each other.
- **AmI systems are distributed**. An AmI node receives its intelligence from its interaction with other AmI nodes in its direct neighborhood by exchanging useful information and services with them. As such, all AmI nodes that potentially cooperate to fulfil their mission constitute a distributed system interacting via ad-hoc-networks.

- **AmI systems heavily rely on environmental information**. Environmental information concerns facts and objects in the surrounding neighborhood of an AmI node such as buildings, cars, people, barriers, temperature, etc. The detection and interpretation of such environmental information requires appropriate sensor interfaces. Sensors may be grouped to form intelligent sensor networks.
- **AmI systems communicate spontaneously and without explicit order**. AmI systems are under continuous pressure to keep their internal information cache up-to-date and comprehensive in order to fulfill their mission. This requires using every opportunity to get in touch with other AmI nodes joining the same ad-hoc network and exchange useful information with them.
- **AmI systems have unconventional human-computer interfaces**. Being embedded systems, AmI systems interact with humans via the application they control. The application dictates the necessary way of interaction between the AmI system and the human and requires unconventional interfaces which are customized for the given application scenarios. For example, let us consider a smart blind stick. A smart blind stick communicates with AmI systems in its neighborhood in order to detect the next safe road crossing. As soon as this information is available, the smart blind stick sends a message to a voice generation component which sends the voice output to the earphones of the handicapped person. It is obvious that it would not make any sense in this application scenario to send this information as text to a screen. In general, AmI systems communicate with humans in the most natural way. This dictates voice and gestured input/output as the preferred media.
- **AmI systems have to be highly adaptive**. Because of missing stable connectivity to services and information sources in ad-hoc networks, AmI systems can never base their operation on the availability of complete and up-to-date information and services. This has the consequence that AmI systems have to organize their services in an adaptive way, i.e. the degree of service varies with the amount of information available and the reach-ability of external services.
- **AmI systems are of highly heterogeneous nature**. In AmI distributed systems, communication requirements can range from few bit/s to several Gbit/s, computational requirements from some kop/s to several Top/s and the power requirements from microwatt to watts. To face this problem AmI architectures are layered architectures i.e. they are based on different types of nodes with various communication bandwidth, signal processing capabilities, and power constraints, respectively. Moreover, AmI nodes from different vendors will frequently join the same ad-hoc network. These nodes are usually based on different processors, operating systems and application software approaches, i.e. we are faced with a highly heterogeneous nature of distributed AmI systems. This poses a challenging burden on the development of international standards for protocols and interchange data formats in order to retain node interoperability.

2.3.2 The Assisted Living Demonstrator

The BelAmI [1] project follows a demonstrator-driven approach: Research issues are derived during the development of demonstrators that shall illustrate solutions for real-world problems. Hence the research issues are driven by the demonstrators. One of these research issues was how to document design decisions, which is addressed in this report.

Among the developed demonstrators to be developed is an “Assisted Living” demonstrator (G7) that actively supports daily routines and ensures that a structured daily routine is adhered to while keeping in the background. The system shall use conspicuous patterns of behavior to infer that a critical situation exists within the home of the assisted person, and then either calls for help or intervenes on its own. To this end it relies on diverse sensors embedded into the living environment of the assisted person, e.g. in cups, furniture, entertainment devices, and the electrical installation.

3 Study Motivation and View

3.1 Why Capturing Design Decision Rationale

As already mentioned above, the Ambient Intelligence domain is a challenging field that requires novel, interdisciplinary approaches. New combinations of required qualities, e.g. flexibility and efficiency, interoperability and safety, call for a thorough consideration of design decisions and their impact. This rises specific issues especially with respect to design and decision making. Furthermore, in most cases additional stakeholders with specific knowledge and views, are involved in the design activities compared to other systems. Especially during the analysis and design phase, these stakeholders have to negotiate and trade off their objectives among each other. During the implementation phase the assembling becomes crucial and tricky, as the pieces of the whole system have to be interconnected with each other. Whereas the technological compatibility problem often can easily be managed in the analysis phase and properly be followed during the developing phase, it often reveals to be quite difficult, to manage quality attributes of systems, like performance, and resource constraints in some small subset of the whole system. Usually, the related decisions show widespread impacts throughout the whole systems and therefore have to be considered by all involved architects and designers. All of them have to negotiate their objectives interfering with the issues of others. Obviously, there is a strong need to intercommunicate with each other and to make design decisions explicit. Hence, we strongly believe that in the AmI context the explicit capturing of design decision rationale is mandatory, in order to support the communication among the diverse stakeholders as much as possible.

3.2 Discover AmI Design Decision Patterns

Analyzing the drivers of each decision, we tried to identify the following issues in decision making:

- **Goals**: which objectives are important? This information can be useful to check, if the decisions' goals are compatible with the ones expected by other stakeholders.
- **Origin of goals**: which objectives are important for whom and where. This information can be useful to check, if the decisions' goals of a specific part of the system are similar to the one expected by other stakeholders (for that part of the system).
- **Common tradeoffs**: Which combinations of goals are difficult to achieve in combination. This information helps to work out and demonstrate potential conflicts in requirements. Such analysis was inspired by a previous work [10] by the tradeoff initiative of SEI.

4 The Proposed Approach: DGA

Following the definition of design rationale provided by Lee in chapter 2.2, the type of information that we want to document for improving the communication among stakeholders are the reasons why a decision has been taken. To this end, we propose an approach called “Decision Goals and Alternatives” (DGA). Here the rationale behind a design decision is represented documenting the data used in the CBAM method (chapter 2.1.3). In other words, we want to document the importance of objectives, available alternatives, and how they fulfil these objectives. In DGA every design decision in a software context is influenced only by the following items:

- functional requirements,
- non-functional requirements (quality attributes and constraints),
- business goals,
- relations among decisions.

The previous list is supposed to be complete, i.e. there is no additional item, not present in the list, which has an influence on a design decision. In the remaining part of the present work we will refer to this list as LoG (list of goals).

In DGA, the concept of decision is divided into decision type and decision alternative concepts. The decision type represents the problem to be solved (e.g. the selection of the programming language) and a decision alternative represents one possible outcome of the decision (e.g. Java). The distinction between the two concepts was driven by the insight that the level of importance of goals and constraints is an attribute of the former, whereas the level of fulfilment of that goals is an attribute of the latter.

Although several relations among decisions seem to be more an attribute of a decision type rather then of a decision alternative (e.g. the relation among the decision on the selection of an operative system and the decision of the selection of a programming language) this is not always true. Consequently, we document the item “relation among decisions” as an attribute of a decision alternative.

DGA approach comprises two phases: The first phase addresses the identification of objectives, constraints, and possible relations among decisions specific to the domain or project where the documentation has to be done. In other words, the previously proposed LoG cannot be used as it is, but needs to be refined to the specific usage context. However, its existence fosters this tailoring activity by providing a predefined high level framework. The

second phase of DGA encompasses a series of iterations. In each iteration one decision is documented by following these steps:

1. Describe the decision type by providing the following information:
   a. Identification number
   b. Textual description of the problem to be solved
   c. Identifier of the system's component, where the decision has the biggest impact
   d. Numerical score for each objective to describe its level of importance
   e. Textual description to describe the reason for the score 1.d
   f. Further textual description to document eventual important information not described before
2. Describe every decision alternatives by providing the following information (for each of them):
   a. Identification number
   b. Textual description of the approach to solve the problem
   c. Identification number of the related decision type (1.a)
   d. Numerical score for each objective to describe its level of fulfilment
   e. Textual description to describe the reason for the score 2.d
   f. Numerical score to describe the level of each existent relations among the current decision alternative and another
   g. Textual description to describe the reason for the score 2.f
   h. Further textual description to document eventual important information not described before

5 An Application of DGA

5.1 Equipment

This chapter describes how the DGA approach in chapter 4 was specialized in our context during the development of a demonstrator in the AmI domain (see Section 2.3.2).

From the characteristics of Ambient Intelligence described in chapter 2.3.1 we represent the spectrum of feasible functional requirements, of the previously proposed LoG, with the following list:

- Mobility support
- Distribution support
- Context-awaren
- Self-aware
- Natural interface
- Adaptability
- Heterogeneity support

Non-functional requirements, of the LoG, can be split in quality attributes of the system and resource constraints. In our context, in order to represent the spectrum of quality attribute objectives, we used the attributes and sub-attributes of the standard ISO 9126 [23]. From the characteristics of the AmI domain described in chapter 2.3.1 we represent the spectrum of potential resource constraints with the following list:

- Memory
- Energy
- Computational power
- Communication range
- Communication bandwidth

As the context of the presented work was an incremental development process, the business goals of the LoG are represented by the following list:

- Reduce the short-term effort (with respect to the next deadline), and
- Minimize the effort in the long-term (extensibility)

The number of relations among decisions is huge and of different types. We believe that the spectrum proposed by Kruchten [24] will fulfil our needs. Consequently, we represent the spectrum of potential relations, of the LoG with the following list:

- constrains
- forbids
- enables
- subsumes
- conflicts with
- overrides
- comprises (is made of)
- is bound to
- is an alternative to
- is related too
- traces to
- does not comply with

5.2 Method

The following steps have been refined for our context from the ones described in chapter 4 for documenting design decisions:

1. Describe the decision type by providing the following information:
    a. Identification number
    b. Textual description of the problem to be solved
    c. Textual identification description of the designer (who made the decision)
    d. Identification number of the system's component where the decision has the biggest impact
    e. Numerical score between 1(low) and 5(high) for each objective to describe its level of importance
    f. Textual description to describe the reason for the score 1.d
    g. Further textual description to document eventual important information not described before
2. Describe every decision alternatives by providing the following information (for each of them):
    a. Identification number
    b. Textual description of the approach to solve the problem
    c. Identification number of the related decision type (1.a)
    d. Numerical score 1(low) and 5(high) for each objective to describe its level of fulfilment
    e. Textual description to describe the reason for the score 2.d

f. Numerical score to describe the level of each existent relations among the current decision alternative and another
g. Textual description to describe the reason for the score 2.f
h. Further textual description to document eventual important information not described before

5.3 Metrics

This chapter describes the metrics used to analyze the patterns in decision making described in chapter 3.2.

5.3.1 Goals

In order to assess the level of importance of objectives we analyze the data described in chapter 5.2 in step 1.e.

5.3.2 Location of Goals

In order to study the level of importance of objectives related to the component, where this decision has the main impact, we analyze the data described in chapter 5.2 in step 2.d. in relation to the identification number described in step 1.d.

5.3.3 Goals correlation

In order to figure out, how much an objective is achieved in relation to the achievement of all the others objectives, a new metric called “distance” was defined as follows:

O_i= objective i

M= number of decision alternatives for a single decision

V_{ij}= value from 1(low) to 5(high) of alternative “i” with respect to the fulfilment of O_j

$D_{Oa\text{-}Ob}$= distance among objectives O_a and O_b = $\sum_{h:\ \text{from 1 to M}} |V_{ha}-V_{hb}| / M$

The range of values for a distance metric is a real number from 0 to 4. The metric represents how much the achievement of an objective is in contrast to the achievement of the other objectives. For example a high value implies a negative relation, scilicet it is difficult to achieve both of them. The previous defined “distance” metric is symmetric by construction. In our best under-

standing, the “amount” of conflict among two software requirements is symmetric (i.e. suppose that two conflicting requirements of a software system are the response time of n seconds and the budget of m euro, the supposed unfeasibility is of course a characteristic of the couple of requirements and consequently it is symmetric in respect to one of them).

6 Results and Comments

The documentation of design decision rationale was organized with an interviewer who helped designers to document their past choices with the framework proposed in chapter 4.

Data described in the followings subsections were derived from a structured interview with three decision makers of the G7 demonstrator (see Section 2.3.2). It deals with 12 decision types and 133 decisions alternatives. The average time needed to document each decision type was about fifteen minutes.

6.1 Data

Figure 8 shows the table used for documenting the information regarding made decisions described in steps 1.a to 1.d, and 2.a to 2.c (cf. chapter 5.2).

Figure 9 shows the table used for documenting the information regarding made decisions described in step 1.e of chapter 5.2. A structured interview was used to collect this data.

Figure 10 shows the table used for documenting the information regarding decisions making described in step 2.e (cf. chapter 5.2).

Designer's Name	Decision ID	Affected Component	Decision Description
Ewoud	A	Middleware	Communication mechanism between iCup and Mona
	1		Pull
	2		Push
	3		Event channel
Ewoud	B	Middleware	Architecture granularity
	4		Common layer
	5		More structured layer
Thomas	C	ICup	Strategy for sending data
	6		Non collecting data but sends it when events happen
	7		Collecting data and then sending it
Thomas	D	ICup	Separation of concerns in the code: movement sensor data vs. send activity
	8		Not separed movement sensor data from its sending activity
	9		Separed movement sensor data from its sending activity
Thomas	E	ICup	Separation of concerns in the code: drink sensor data vs. send activity
	10		Not separed drink sensor data from its sending activity
	11		separed drink sensor data from its sending activity
Thomas	F	ICup	Separation of concerns in the code: different types of sensor functionality
	12		Not separed sensor functionality
	13		Separed sensor functionality
Michalis	G	Middleware	Where to place the look-up operation
	14		On the component
	15		On the middleware
Michalis	H	Middleware	How to realize the publish subscribe mechanism
	16		Directly
	17		With a mediator
Ewoud	I	Cooler	How to sincronize the content of the refrigerator
	18		Aml cooler provide an interface
	19		Aml will sends events regarding the content of the refrigerator
Ewoud	J	Mona	Reasoning implementation
	20		Directly code the x statements
	21		Configurate the reasoning which creates the x statements
Ewoud	K	Mona	Type of database for the facts
	22		ms-acces
	23		mysql
Ewoud	L	Mona	Method to aggregate the information (facts)
	24		Each time
	25		Tracing

Figure 8: Design decision data comprising Designer's name (first column), Decisions ID (letter in the second column, Decision alternatives ID(numbers in the second column), Affected Component (third column) , Decision Description (fields in the fourth column corresponding to a row with a number in the third column), Decision Alternative Description (fields in the fourth column corresponding to a row with a letter in the third column).

TYPE	SUB-TYPE	DECISION Type ID											
		A	B	C	D	E	F	G	H	I	J	K	L
Non Functional Requirements	**Functionality**	1	1	1	1	1	1	1	1	1	1	1	1
	Suitability	1	1	1	1	1	1	1	1	3	1	1	1
	Accuracy	1	1	1	1	1	1	1	1	1	1	1	1
	Interoperability	5	5	1	1	1	1	1	1	1	1	1	1
	Security	1	4	1	1	1	1	1	1	1	1	1	1
	Compliance	1	1	1	1	1	1	1	1	1	1	2	1
	Reliability	5	1	2	1	1	1	1	3	1	1	1	1
	Maturity	1	1	1	1	1	1	2	1	1	1	1	1
	Fault tolerance	1	5	1	1	1	1	1	1	1	1	1	1
	Recoverability	1	1	1	1	1	1	1	1	1	1	1	1
	Compliance	1	1	1	1	1	1	1	1	1	1	1	1
	Usability	1	1	1	1	1	1	4	4	1	1	1	1
	Understand.	1	1	1	5	5	5	4	4	1	1	1	1
	Learnability	1	1	1	5	5	5	4	4	1	1	1	1
	Operability	1	1	1	5	5	5	4	4	1	1	1	1
	Attractiveness	1	1	1	1	1	1	4	4	1	1	1	1
	Compliance	1	1	1	1	1	1	2	1	1	1	2	1
	Efficiency	1	5	5	1	1	1	1	1	1	1	1	1
	Time Behavior	4	4	1	1	1	1	1	1	1	1	1	5
	Resource util.	1	5	5	1	1	1	2	1	2	1	1	4
	Compliance	5	1	1	1	1	1	1	1	1	1	1	1
	Maintainability	5	5	1	5	5	5	1	4	1	1	1	1
	Analyzability	1	1	1	5	5	5	1	1	1	1	1	1
	Changeability	5	5	5	5	5	5	5	5	4	4	1	1
	Stability	5	5	1	4	4	4	1	4	1	1	1	1
	Testability	1	1	1	5	5	5	1	1	1	1	1	1
	Compliance	3	1	1	1	1	1	1	1	1	1	1	1
	Portability	3	3	1	1	1	1	1	1	1	1	3	1
	Adaptability	1	1	1	1	1	1	1	1	1	5	1	1
	Instability	1	1	1	1	1	1	1	1	1	1	1	1
	Co-existence	1	1	1	1	1	1	1	1	1	1	1	1
	Replaceability	1	1	1	5	5	5	1	1	1	4	1	1
	Compliance	1	1	1	1	1	1	1	1	1	1	1	1
Functional Requirem	Embedded	3	1	3	1	1	1	1	1	1	1	1	1
	Mobile	5	5	5	1	1	1	5	5	3	1	1	1
	Distributed	5	5	4	1	1	1	5	5	3	1	3	1
	Context-aware	1	5	3	1	1	1	3	5	1	1	1	1
	Self-aware	1	5	1	1	1	1	5	5	1	1	1	1
	Natural int.	1	1	1	1	1	1	1	1	1	1	1	1
	Adaptive	1	5	1	1	1	1	5	5	3	4	1	1
	Heterogeneous	5	5	1	1	1	1	5	5	3	1	1	1
Costraints	Memory	5	5	3	1	1	1	1	1	1	2	1	1
	Energy	2	5	5	1	1	1	1	1	1	1	1	1
	Comput. power	1	5	3	1	1	1	1	1	1	1	1	5
	range comm.	1	1	1	1	1	1	1	1	1	1	1	1
	bandwidth	1	1	1	1	1	1	1	1	1	1	1	1
Business	Cost / effort short	5	5	5	5	5	5	5	5	5	5	5	5
goals	Cost / effort long	4	4	4	4	4	4	4	4	4	4	4	4

Figure 9: Design decision data regarding the importance of every goal for a specific decision type.

TYPE	SUB-TYPE	DECISION TYPE ID																							
		a			b		c		d		e		f		g		h		i		j		k		l	
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Non Functional Requirements	**Functionality**																									
	Suitability																		2	4						
	Accuracy																									
	Interoperability	1	1	5																						
	Security																									
	Compliance																						2	5		
	Reliability						2	4									2	5								
	Maturity														5	2										
	Fault tolerance																									
	Recoverability																									
	Compliance																									
	Usability														2	5	5	3								
	Understand.								1	4	1	4	1	4	2	5	5	3								
	Learnability														2	5	5	3								
	Operability														2	5	5	3								
	Attractiveness														2	5	5	3								
	Compliance														5	1							2	5		
	Efficiency																									
	Time Behavior																								1	5
	Resource util.				1	5													5	1					1	5
	Compliance																									
	Maintainability	5	5	5					2	4	2	4	2	4			5	3								
	Analyzability								2	4	2	4	2	4												
	Changeability	1	1	5	4	4			2	4	2	4	2	4			3	5	4	2	1	5				
	Stability	5	5	1					2	4	2	4	2	4			3	5								
	Testability								2	4	2	4	2	4	4	3										
	Compliance																									
	Portability								2	4	2	4	2	4									1	5		
	Adaptability																				1	5				
	Instability																									
	Co-existence																									
	Replaceability								1	4	1	4	1	4							2	4				
	Compliance																									
Functional Requirements	Embedded																									
	Mobile	5	5	5											1	5	3	5	4	4						
	Distributed	5	5	5											5	5	3	5	4	4			2	5		
	Context-aware																3	5								
	Self-aware														3	5	2	5								
	Natural int.																									
	Adaptive														2	5	2	5	5	2	2	5				
	Heterogeneous	1	3	5	1	5									3	3	3	5	5	5						
Costraints	Memory	5	5	5	1	5	4	1													5	4				
	Energy	3	5	5	1	5	1	4																		
	Comput. power	5	5	3	1	5	4	2																	1	5
	range comm.																									
	bandwidth																									
Business	Cost / effort short	2	2	5	5	1	4	4	4	3	4	3	4	3	5	3	4	2	5	2	5	3	5	1	5	2
goals	Cost / effort long	4	4	2	1	5	4	4	1	5	1	5	1	5	4	5	4	4	5	2	2	5	2	4	4	4
Made				x		x		x		x		x		x		x		x		x	x		x		x	

Figure 10: Design decision data regarding how a specific decision alternative fulfils a specific objective.

6.2 AmI Design Decision Patterns

6.2.1 Goals

The following figure shows the importance related to every possible objective computed in average of every documented decision type.

Rank	Objectives	Importance
1	Cost / effort short	5,00
2	Changeability	4,17
3	Cost / effort long	4,00
4	Maintainability	2,92
5	Distributed	2,92
6	Mobile	2,83
7	Stability	2,67
8	Understand.	2,50
9	Operability	2,50
10	Learnability	2,50
11	Heterogeneous	2,50
12	Adaptive	2,42
13	Replaceability	2,25
14	Resource util.	2,08
15	Testability	2,00
16	Self-aware	2,00
17	Context-aware	2,00
18	Analyzability	2,00
19	Memory	1,92
20	Time Behavior	1,83
21	Comput. power	1,83
22	Energy	1,75
23	Interoperability	1,67
24	Efficiency	1,67
25	Reliability	1,58
26	Usability	1,50
27	Portability	1,50
28	Attractiveness	1,50
29	Fault tolerance	1,33
30	Embedded	1,33
31	Efficiency Compliance	1,33
32	Adaptability	1,33
33	Security	1,25
34	Suitability	1,17
35	Maintainability Compliance	1,17
36	Usability Compliance	1,17
37	Maturity	1,08
38	Functionality Compliance	1,08
39	Functionality	1,00
40	Recoverability	1,00
41	range comm.	1,00
42	Natural int.	1,00
43	Instability	1,00
44	Reliability Compliance	1,00
45	Portability Compliance	1,00
46	Co-existence	1,00
47	bandwidth	1,00
48	Accuracy	1,00

Figure 11: Objectives ranked based on their average importance.

6.2.2 Location of Goals

The following figure shows the importance related to every possible objective computed in average of every documented decision type related to the component, where each decision has the main impact.

Objectives / Component		Middleware	ICup	Cooler	Mona
Non Functional Requirements	**Functionality**	1,55	1,00	1,40	1,07
	Suitability	1,00	1,00	3,00	1,00
	Accuracy	1,00	1,00	1,00	1,00
	Interoperability	3,00	1,00	1,00	1,00
	Security	1,75	1,00	1,00	1,00
	Compliance	1,00	1,00	1,00	1,33
	Reliability	2,81	1,25	1,00	1,00
	Maturity	1,25	1,00	1,00	1,00
	Fault tolerance	2,00	1,00	1,00	1,00
	Recoverability	1,00	1,00	1,00	1,00
	Compliance	1,00	1,00	1,00	1,00
	Usability	2,50	2,80	1,20	1,66
	Understand.	2,50	4,00	1,22	1,68
	Learnability	2,50	4,00	1,18	1,79
	Operability	2,50	4,00	1,32	1,83
	Attractiveness	2,50	1,00	1,27	1,69
	Compliance	1,25	1,00	1,00	1,33
	Efficiency	2,67	2,00	1,33	1,78
	Time Behavior	2,50	1,00	1,00	2,33
	Resource util.	2,25	2,00	2,00	2,00
	Compliance	2,00	1,00	1,00	1,00
	Maintainability	3,95	4,20	1,60	1,20
	Analyzability	1,00	4,00	1,00	1,00
	Changeability	5,00	5,00	4,00	2,00
	Stability	3,75	3,25	1,00	1,00
	Testability	1,00	4,00	1,00	1,00
	Compliance	1,50	1,00	1,00	1,00
	Portability	2,00	1,60	1,00	2,13
	Adaptability	1,00	1,00	1,00	2,33
	Instability	1,00	1,00	1,00	1,00
	Co-existence	1,00	1,00	1,00	1,00
	Replaceability	1,00	4,00	1,00	2,00
	Compliance	1,00	1,00	1,00	1,00
Functional Requirements	Embedded	1,50	1,50	1,00	1,00
	Mobile	5,00	2,00	3,00	1,00
	Distributed	5,00	1,75	3,00	1,67
	Context-aware	3,50	1,50	1,00	1,00
	Self-aware	4,00	1,00	1,00	1,00
	Natural int.	1,00	1,00	1,00	1,00
	Adaptive	4,00	1,00	3,00	2,00
	Heterogeneous	5,00	1,00	3,00	1,00
Costraints	Memory	3,00	1,50	1,00	1,33
	Energy	2,25	2,00	1,00	1,00
	Comput. power	2,00	1,50	1,00	2,33
	range comm.	1,00	1,00	1,00	1,00
	bandwidth	1,00	1,00	1,00	1,00
Business goals	Cost / effort short	5,00	5,00	5,00	5,00
	Cost / effort long	4,00	4,00	4,00	4,00

Figure 12: Objectives ranked based on their average importance.

Analyzing this data, it is easy to understand that objectives like heterogeneity, mobility and distribution be of high importance on the middleware component with respect to other objectives like analyzability and replaceability, which have the main importance on the ICup component.

6.2.3 Common trade-offs

The following figure shows the value of the metric distance discussed in chapter 5.3.3 applied to every pair of objectives.

		Non Functional Requirements																																Functional Requirements								Costraints					Business Goals	
		Functionality	Suitability	Accuracy	Interoperability	Security	Compliance	**Reliability**	Maturity	Fault tolerance	Recoverability	Compliance	**Usability**	Understand.	Learnability	Operability	Attractiveness	Compliance	**Efficiency**	Time Behavior	Resource util.	Compliance	**Maintainability**	Analyzability	Changeability	Stability	Testability	Compliance	**Portability**	Adaptability	Instability	Co-existence	Replaceability	Compliance	Embedded	Mobile	Distributed	Context-aware	Self-aware	Natural int.	Adaptive	Heterogeneous	Memory	Energy	Comput. power	range comm.	bandwidth	Cost / effort short	Cost / effort long
Non Functional Requirements	**Functionality**		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
	Suitability	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1,0	?	?	?	2,0	?	?	?	?	?	?	?	?	?	?	3,0	3,0	?	?	?	1,5	2,0	?	?	?	?	?	1,5	1,5
	Accuracy	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
	Interoperability	?	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1,3	?	4,0	0,0	?	?	?	?	?	?	?	?	?	1,3	1,3	?	?	?	?	3,3	1,3	2,0	0,7	?	?	3,3	1,0
	Security	?	?	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
	Compliance	?	?	?	?	?		?	?	?	?	?	?	?	?	?	?	4,0	?	?	?	?	?	?	?	?	?	?	3,5	?	?	?	?	?	?	?	4,0	?	?	?	?	?	?	?	?	?	?	0,5	3,5
	Reliability	?	?	?	?	?	?		?	?	?	?	1,5	1,5	1,5	1,5	1,5	?	?	?	?	?	1,5	?	3,5	3,5	?	?	?	?	?	?	?	?	?	3,5	3,5	3,5	4,0	?	4,0	3,5	1,5	3,5	2,0	?	?	2,3	2,8
	Maturity	?	?	?	?	?	?	?		?	?	?	1,0	1,0	1,0	1,0	1,0	3,5	?	?	?	?	?	?	?	?	3,0	?	?	?	?	?	?	?	?	0,5	2,5	?	1,5	?	1,0	2,5	?	?	?	?	?	3,5	2,0
	Fault tolerance	?	?	?	?	?	?	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
	Recoverability	?	?	?	?	?	?	?	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
	Compliance	?	?	?	?	?	?	?	?	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
	Usability	?	?	?	?	?	?	1,5	1,0	?	?	?		4,0	4,0	4,0	4,0	0,5	?	?	?	?	4,0	?	2,0	2,0	2,0	?	?	?	?	?	?	?	?	2,8	2,3	2,0	2,5	?	2,8	2,3	?	?	?	?	?	2,3	3,0
	Understand.	?	?	?	?	?	?	1,5	1,0	?	?	?	4,0		4,0	4,0	4,0	0,5	?	?	?	?	3,6	3,5	3,1	3,1	3,1	?	3,5	?	?	?	4,0	?	?	2,8	2,3	2,0	2,5	?	2,8	2,3	?	?	?	?	?	2,1	3,3
	Learnability	?	?	?	?	?	?	1,5	1,0	?	?	?	4,0	4,0		4,0	4,0	0,5	?	?	?	?	4,0	?	2,0	2,0	2,0	?	?	?	?	?	?	?	?	2,8	2,3	2,0	2,5	?	2,8	2,3	?	?	?	?	?	2,3	3,0
	Operability	?	?	?	?	?	?	1,5	1,0	?	?	?	4,0	4,0	4,0		4,0	0,5	?	?	?	?	4,0	?	2,0	2,0	2,0	?	?	?	?	?	?	?	?	2,8	2,3	2,0	2,5	?	2,8	2,3	?	?	?	?	?	2,3	3,0
	Attractiveness	?	?	?	?	?	?	1,5	1,0	?	?	?	4,0	4,0	4,0	4,0		0,5	?	?	?	?	4,0	?	2,0	2,0	2,0	?	?	?	?	?	?	?	?	2,8	2,3	2,0	2,5	?	2,8	2,3	?	?	?	?	?	2,3	3,0
	Compliance	?	?	?	?	?	4,0	?	3,5	?	?	?	0,5	0,5	0,5	0,5	0,5		?	?	?	?	?	?	?	?	2,5	?	3,5	?	?	?	?	?	?	0,0	3,0	?	1,0	?	0,5	2,0	?	?	?	?	?	1,8	2,5
	Efficiency	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
	Time Behavior	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?		4,0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	4,0	?	?	0,5	2,0
	Resource util.	?	1,0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	4,0		?	?	?	2,5	?	?	?	?	?	?	?	?	?	?	2,0	2,0	?	?	?	3,5	3,0	4,0	4,0	4,0	?	?	1,3	3,2
	Compliance	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
	Maintainability	?	?	?	1,3	?	?	1,5	?	?	?	?	4,0	3,6	4,0	4,0	4,0	?	?	?	?	?		4,0	2,9	3,3	4,0	?	4,0	?	?	?	3,5	?	?	3,2	3,2	2,0	1,5	?	1,5	2,0	4,0	3,3	3,3	?	?	2,5	2,8
	Analyzability	?	?	?	?	?	?	?	?	?	?	?	?	3,5	?	?	?	?	?	?	?	?	4,0		4,0	4,0	4,0	?	4,0	?	?	?	3,5	?	?	?	?	?	?	?	?	?	?	?	?	?	?	2,5	3,0
	Changeability	?	2,0	?	4,0	?	?	3,5	?	?	?	?	2,0	3,1	2,0	2,0	2,0	?	?	?	2,5	?	2,9	4,0		2,9	4,0	?	4,0	4,0	?	?	3,4	?	?	2,6	2,6	4,0	3,5	?	3,5	2,9	1,6	2,0	1,2	?	?	2,5	2,6
	Stability	?	?	?	0,0	?	?	3,5	?	?	?	?	2,0	3,1	2,0	2,0	2,0	?	?	?	?	?	3,3	4,0	2,9		4,0	?	4,0	?	?	?	3,5	?	?	3,2	3,2	4,0	3,5	?	3,5	2,0	2,7	2,0	3,3	?	?	1,9	3,0
	Testability	?	?	?	?	?	?	?	3,0	?	?	?	2,0	3,1	2,0	2,0	2,0	2,5	?	?	?	?	4,0	4,0	4,0	4,0		?	4,0	?	?	?	3,5	?	?	1,5	2,5	?	2,5	?	2,0	3,5	?	?	?	?	?	2,8	3,0
	Compliance	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
	Portability	?	?	?	?	?	3,5	?	?	?	?	?	?	3,5	?	?	?	3,5	?	?	?	?	4,0	4,0	4,0	4,0	4,0	?		?	?	?	3,5	?	?	?	3,5	?	?	?	?	?	?	?	?	?	?	1,9	3,0
	Adaptability	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	4,0	?	?	?	?		?	?	3,0	?	?	?	?	?	?	?	3,5	?	1,5	?	?	?	?	1,0	3,5
	Instability	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
	Co-existence	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
	Replaceability	?	?	?	?	?	?	?	?	?	?	?	?	4,0	?	?	?	?	?	?	?	?	3,5	3,5	3,4	3,5	3,5	?	3,5	3,0	?	?		?	?	?	?	?	?	?	3,5	?	2,5	?	?	?	?	2,0	3,5
	Compliance	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Functional Requirements	Embedded	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?
	Mobile	?	3,0	?	1,3	?	?	3,5	0,5	?	?	?	2,8	2,8	2,8	2,8	2,8	0,0	?	?	2,0	?	3,2	?	2,6	3,2	1,5	?	?	?	?	?	?	?	?		3,6	4,0	3,3	?	3,2	2,7	4,0	3,3	3,3	?	?	1,9	2,6
	Distributed	?	3,0	?	1,3	?	4,0	3,5	2,5	?	?	?	2,3	2,3	2,3	2,3	2,3	3,0	?	?	2,0	?	3,2	?	2,6	3,2	2,5	?	3,5	?	?	?	?	?	?	3,6		4,0	3,3	?	2,8	2,7	4,0	3,3	3,3	?	?	2,0	2,9
	Context-aware	?	?	?	?	?	?	3,5	?	?	?	?	2,0	2,0	2,0	2,0	2,0	?	?	?	?	?	2,0	?	4,0	4,0	?	?	?	?	?	?	?	?	?	4,0	4,0		3,5	?	3,5	4,0	?	?	?	?	?	2,0	3,0
	Self-aware	?	?	?	?	?	?	4,0	1,5	?	?	?	2,5	2,5	2,5	2,5	2,5	1,0	?	?	?	?	1,5	?	3,5	3,5	2,5	?	?	?	?	?	?	?	?	3,3	3,3	3,5		?	3,8	3,3	?	?	?	?	?	1,8	3,0
	Natural int.	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?		?	?	?	?	?	?	?	?	?
	Adaptive	?	1,5	?	?	?	?	4,0	1,0	?	?	?	2,8	2,8	2,8	2,8	2,8	0,5	?	?	3,5	?	1,5	?	3,5	3,5	2,0	?	?	3,5	?	?	3,5	?	?	3,2	2,8	3,5	3,8	?		2,8	2,0	?	?	?	?	2,1	3,4
	Heterogeneous	?	2,0	?	3,3	?	?	3,5	2,5	?	?	?	2,3	2,3	2,3	2,3	2,3	2,0	?	?	3,0	?	2,0	?	2,9	2,0	3,5	?	?	?	?	?	?	?	?	2,7	2,7	4,0	3,3	?	2,8		2,8	3,2	2,4	?	?	2,3	2,6
Costraints	Memory	?	?	?	1,3	?	?	1,5	?	?	?	?	?	?	?	?	?	?	?	?	4,0	?	4,0	?	1,6	2,7	?	?	?	1,5	?	?	2,5	?	?	4,0	4,0	?	?	?	2,0	2,8		2,9	3,6	?	?	2,0	2,7
	Energy	?	?	?	2,0	?	?	3,5	?	?	?	?	?	?	?	?	?	?	?	?	4,0	?	3,3	?	2,0	2,0	?	?	?	?	?	?	?	?	?	3,3	3,3	?	?	?	?	3,2	2,9		2,7	?	?	1,9	2,9
	Comput. power	?	?	?	0,7	?	?	2,0	?	?	?	?	?	?	?	?	?	?	?	4,0	4,0	?	3,3	?	1,2	3,3	?	?	?	?	?	?	?	?	?	3,3	3,3	?	?	?	?	2,4	3,6	2,7		?	?	1,2	3,0
	range comm.	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?		?	?	?
	bandwidth	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?		?	?
Business goals	Cost / effort short	?	1,5	?	3,3	?	0,5	2,3	3,5	?	?	?	2,3	2,1	2,3	2,3	2,3	1,8	?	0,5	1,3	?	2,5	2,5	2,5	1,9	2,8	?	1,9	1,0	?	?	2,0	?	?	1,9	2,0	2,0	1,8	?	2,1	2,3	2,0	1,9	1,2	?	?		2,0
	Cost / effort long	?	1,5	?	1,0	?	3,5	2,8	2,0	?	?	?	3,0	3,3	3,0	3,0	3,0	2,5	?	2,0	3,2	?	2,8	3,0	2,6	3,0	3,0	?	3,0	3,5	?	?	3,5	?	?	2,6	2,9	3,0	3,0	?	3,4	2,6	2,7	2,9	3,0	?	?	2,0	

Figure 13: Correlations among objectives.

The previous table contains uncalculated relations among objectives (valued with the letter "?"), because not every goal was scored in relation to every decision alternative. In order to improve the visibility of the previous figures, data are converted using the following symbols:

- "+" for a value from 0 to 0.99,
- "-" for a value from 1 to 1.99,
- "- -" for a value from 2 to 2.99,
- "- - -"for a value from 3 to 4,
- " " for "?".

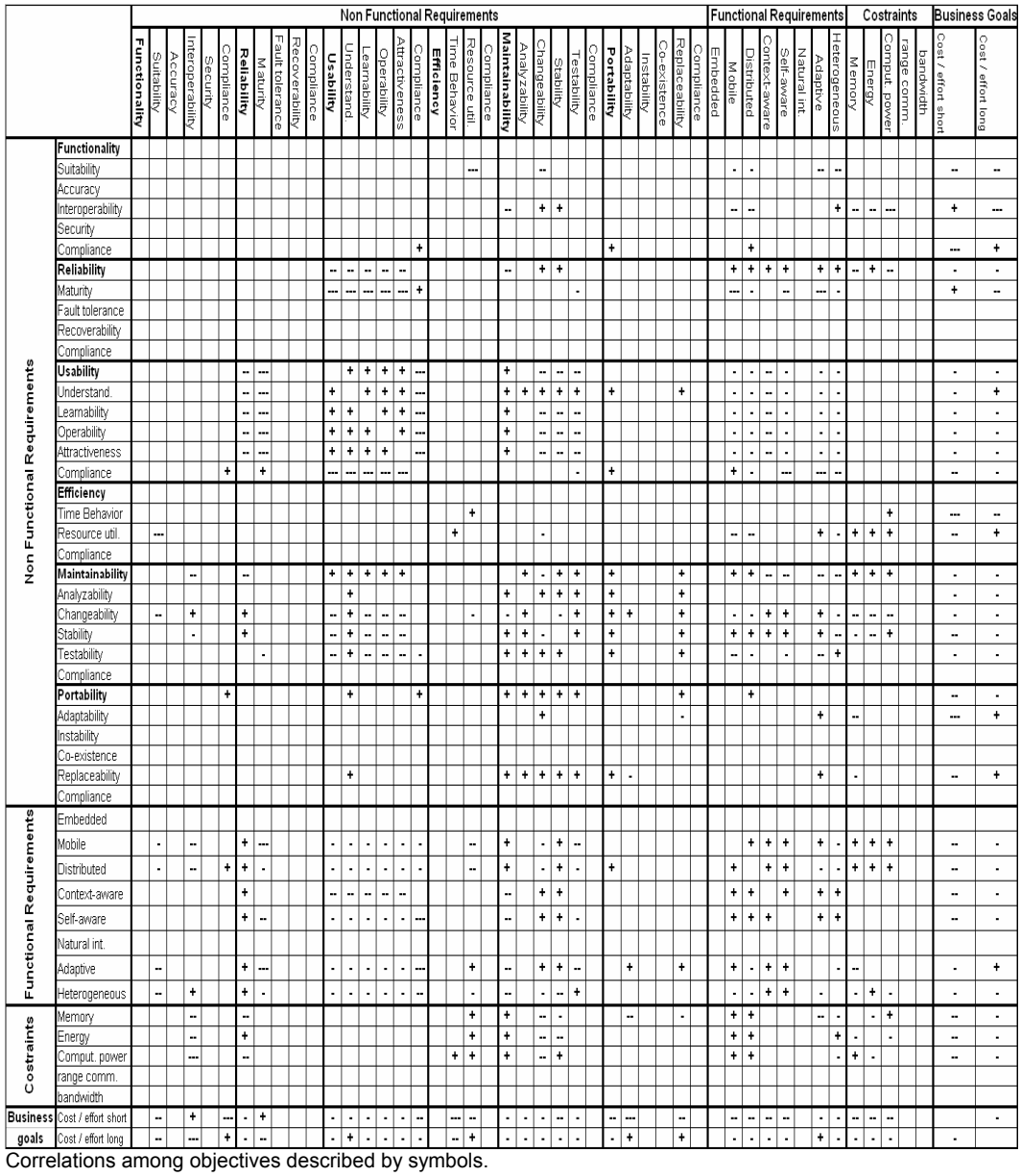


Figure 14: Correlations among objectives described by symbols.

7 Conclusion and Future Work

7.1 Conclusion

In this report we have motivated and presented the DGA approach for documenting design decision rationale and to analyze recurring patterns in decision making (importance, location and conflicts of quality attribute). The DGA approach revealed to be suitable for the proposed context,

We strongly believe that the use of such a technique in software development would provide several benefits as already discussed in papers dealing with design rationale.

7.2 Future Work

7.2.1 Which, When and How to Document Design Decision Rationale?

The work by Karsenty[29] discusses documenting design rationale for its pros and cons. Furthermore, another work of him [28] demonstrates the usefulness of documenting design rationale for design reuse. However there are no empirical studies trying to investigate the effects of variables “which”, “when”, and ”how” of design decision documentation on the variable “usefulness”.

7.2.2 Supporting Software Decision Making

The CBAM method discussed in chapter 2.1.3 contains several approximations. However, in our best understanding, there is currently no other method to support software designers in the decision making activity significantly different from CBAM . Furthermore, there are no empirical studies trying to investigate the effect of increasing the complexity (needed effort) of the method on its accuracy.

A decision making support tool would be helpful for software designers in dealing with the great number of traces among requirements (objectives) and design decisions [32].

Fewer methods exist that systematically bridge the gap between software requirements and software architecture [33]. We consider a survey on such topic as the new step of our research.

References

[1] Bilateral German-Hungarian Collaboration Project on Ambient Intelligence Systems, Project Outline V2.2 April 2005.

[2] P. Clements et al., Documenting Software Architectures: Views and Beyond, Pearson Education, 2003.

[3] P. Kruchten, “The 4+1 View Model of Architecture,” IEEE Software, vol. 12, no. 6, 1995, pp. 42–50.

[4] Jeff Tyree, Art Akerman. "Architecture Decisions: Demystifying Architecture," IEEE Software, vol. 22, no. 2, pp. 19-27, March/April, 2005.

[5] Bana e Costa C. A. (ed.), 1990. Readings in Multiple Criteria Decision Aid. Springer-verlag, Berlin.

[6] K. Paul Yoon, Ching-Lai Hwang, Multiple Attribute Decision Making : An Introduction (Quantitative Applications in the Social Sciences), 1995, ISBN: 0803954867.

[7] Multi-Objective Decision Making: Based on the Proceedings of a Conference on Multi-Objective Decision Making (Conference Series (Institute of Mathematics and Its Applications).), 1983, ISBN: 0122670809.

[8] Chung, L., Nixon, B., Yu, E. and Mylopoulos, J. Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Boston, Ma.

[9] Len Bass, Mark Klein, Felix Bachmann, "Quality Attribute Design Primitives and the Attribute Driven Design Method." 4th International Workshop on Product Family Engineering. Bilbao, Spain, 3-5 October 2001.

[10] Hernan R. Eguiluz and Mario R. Barbacci “Interactions among techniques addressing quality attributes”, CMU/SEI-2003-TR-003

[11] Barry Boehm, Alexander Egyed, Julie Kwan, Dan Port, Archita Shah, Ray Madachy. "Using the WinWin Spiral Model: A Case Study," Computer, vol. 31, no. 7, pp. 33-44, July, 1998.

[12] Volere Requirements Specification Template, http://www.systemsguild.com/GuildSite/Robs/Template.html

[13] R. Kazman, P. In, H-M Chen, From Requirements Negotiation to Software Architecture Decisions, Information and Software Technology, 2005, to appear.

[14] Garcia, A., Howard, H., Stefik, M. (1993), Active Design Documents: A New Approach for Supporting Documentation in Preliminary Routine Design, Tech. Report 82, Stanford Univ. Center for Integrated Facility Engineering, Stanford, Calif.

[15] Gruber, T. (1990), Model-based Explanation of Design Rationale, in Proceedings of the AAAI-90 Explanation Workshop, Boston, July 30, 1990.

[16] Ganeshan R., Garrett J., Finger, S. (1994), A framework for representing design intent, Design Studies Journal, V15 No. 1, January, pp. 59-84.

[17] Lee, J. (1997), Design Rationale Systems: Understanding the Issues, IEEE Expert, Vol. 12, No. 3, pp. 78-85.

[18] Fischer, G., Lemke, A., McCall, R., Morch, A. (1995), Making Argumentation Serve Design, in Design Rationale Concepts, Techniques, and Use, T. Moran and J. Carroll, eds., Lawrence Erlbaum Associates, pp. 267-294

[19] Pena-Mora, F., Sriram, D., Logcher, R. (1995), Design Rationale for Computer-Supported Conflict Mitigation, ASCE Journal of Computing in Civil Engineering, pp. 57-72.

[20] C. Zannier, F. Maurer: A Qualitative Empirical Evaluation of Design Decisions, Workshop on Human & Social Factors of Software Engineering; ACM Digital Libary, ACM Press 2005

[21] Philippe Kruchten, "An Ontology of Architectural Design Decisions," in: Jan Bosch (ed.), Proc. of the 2nd Workshop on Software Variability Management, Groningen, NL, Dec. 3-4, 2004

[23] ISO/IEC9126 Int'l Organization of Standardization and Int'l Electrotechnical Commission, Information Technology Software Product Evaluation Quality Characteristics and Guidelines for Their Use, ISO/IEC 9216: 1991(E), 1991.

[24] Conklin, J.E. and Begeman, M.L. gIBIS: A Hypertext Tool for Exploratory Policy Discussion, ACM Transactions on Office Information Systems, 6,303-331, 1988.

[25] MacLean, A., Young, R.M., Bellotti, V. and Moran T.P., Questions, options and criteria: Elements of design space analysis, Human-Computer Interaction, 6(3&4), 201-250, 1991.

[26] Cambridge Dictionary of American English™, http://dictionary.cambridge.org/define.asp?dict=A&key=tradeoff*1+0

[27] Saul I. Gass, Linear Programming: Methods and Applications, 2003 (fifth edition), ISBN: 048643284X.

[28] L. Karsenty, “An Empirical Evaluation of Design Rationale Documents”, Proc. of the Conf. on Human Factors in Computing Systems, ACM Press, NY, 1996, pp. 150-156.

[29] Shum S. & Hammond N., Argumentation-based design rationale: what use at what cost? International Journal of Human-Computer Studies, 40, 603-652, 1994.

[30] Thomas G. Lane. Studying software architecture through design spaces and rules. Technical Report CMU/SEI-90-TR18 ESD-90-TR-219, Carnegie Mellon University, November 1990

[31] Akao, Y., Quality Function Deployment, Productivity Press, Cambridge, MA (1990).

[33] James D. Kiper, Martin S. Feather, “Requirements, Architectures and Risks”, STRAW03.

[34] Software Requirements and Architectures: The CBSP Approach. 5th IEEE International Symposium on Requirements Engineering (RE’01), Toronto, August 2001.

Document Information

Title:	Documenting Design Decisions: A Framework and its Evaluation in the Ambient Intelligence Domain
Date:	2.3.06
Report:	IESE-050.06/E
Status:	Final
Distribution:	Internal