A Representation Model of Design Rationale for the Design of ERP Systems

Sandra Kawamoto
Jorge Rady de Almeida Junior

Escola Politécnica da Universidade de São Paulo, Depto. de Eng. de Computação e Sistemas Digitais
sandrak@totvs.com.br, jorge.almeida@poli.usp.br

Abstract: Special attention has been given to documentation and support activities in Software Engineering design, mainly when they are related to complex systems and distributed teams. Usually, information related to the final decisions of each phase is registered. However, the reasons of each decision and the alternatives that were considered and discarded are not documented. Capture and recovery of this type of information, in an efficient way, are the purposes of the Design Rationale study. Recording this information can facilitate maintenance, reuse and even the design phase, providing a better understanding of the system, by knowledge spread, communication and integration among the project team. The main concern is when and how to capture this information with low interference in designers' usual activities, so that benefits can overcome the costs involved in this task. The purpose of the present work is to show, with plausible reasons, the great usefulness of the application of the Design Rationale technique in ERP systems design, proposing a new representation model for recording design decisions in these systems.

Keywords: Design Rationale, Representation Models, Enterprise Resource Planning (ERP).

(Received April 07, 2009 / Accepted July 16, 2009)

1. Introduction

The main focus of this work is the proposal of a new representation model of Design Rationale for computational systems design. The proposal is applied to an ERP (Enterprise Resource Planning) development.

Dutoit et al. [01] state that new technologies (web oriented, components, patterns, etc.) and the process models (agile, risk-oriented, and model-oriented) reflect the challenges of software engineering nowadays: design of more complex software in distributed teams, at lower time and cost. Dutoit et al. [01] complements that the emphasis on technologies and process models masks the fact that software engineering is basically an activity based on people and that the success of a project or product depends on the decisions taken during the whole project.

Normally, the standard documentation of projects contains a description of the final project, i.e., the final decisions that were taken. Design Rationale encompasses not only the decisions, but also the reasons that supported each decision, including its reasons and alternatives considered [02]. Design Rationale provides aid for review, maintenance, documentation, evaluation and learning of the project.

Design Rationale is especially important for software projects. In general, the software undergoes several changes during its development cycle, not only to perform corrections as well to change or incorporate new requirements. The software maintenance phase is very expensive and it becomes more complex if the original team of architects is not available. Software is usually handled by different teams, and only part of them participates effectively throughout the project review process. Another particularity of a software project is the existence of multiple solutions for the same problem. Considering all these features, the Design Rationale has great potential to be a technique that can add value to the software development process.

Thus, the purpose is to propose a new model of representation of Design Rationale, to be applied to ERP projects. The intention of this new model is to supply some deficiencies of three Design Rationale models studied, providing a more powerful tool for documentation and analysis.

The proposal is based on the description and comparison of three models of Design Rationale: QOC (Questions, Options and Criteria), IBIS (Issue Based Information System) and DRL (Decision Representation Language). These are the main models considered in researches on the subject [01], [03], [04], [05].

The choice of Design Rationale as the basic concept of this paper is justified by the fact that its use can provide great benefits to the software development process [06], [07], [08]. However, in practice there was still not a significant adhesion to its use [09]. At the beginning of the research, many hoped that its practical application could, quickly, be widespread. There was not, however, the estimates of how difficult it would be to define approaches and systems that could be successfully used in real projects [01].

The choice here was over ERP systems because practically all the specific characteristics of this kind of application are favorable to the use of the Design Rationale.

According to Dutoit et al. [01], after more than 35 years of research in Design Rationale, some basic questions remain unsolved, including:

1. How to capture the justification of decisions in the project (rationales), that is, how to extract and store the information;
2. What the best way to represent the justification of project decisions is;
3. How to formalize these pieces of information, that is, how to transform the information in the desired representation form;
4. How the stored information may be used;
5. What the potential barriers to the capture, representation, formalization and use of Design Rationale are.

This paper contributes to answer question 2, because it analyzes the Design Rationale representation models, proposing a more appropriate model to ERP systems.

In section 2, the basic theory which involves the concept of Design Rationale is described, contemplating its definition and presentation of three Design Rationale representation models, including a comparison between them.

Section 3 contains the proposal of this paper, that is, an analysis of the use of the Design Rationale in developing environments of Enterprise Resource Planning (ERP), presenting a proposal for a more appropriate representation model for this kind of system. Finally, section 4 contains the final conclusions.

2. Design Rationale

Software-oriented artifact designs are still found in practice. Considering this paradigm, the emphasis is on the generation and tracking of intermediary project artifacts (requirements, specifications, prototypes and documentation) that lead to the final system. But the development process of these artifacts remains implicit and hidden in meetings, notebooks, e-mails or in the programmer's memory [06]. Consequently, this information will not be available and what is worse, their recovery may not be possible in the needed time.

In the new technologies (web orientation, components and application models) and in the newer process models (agile, the risk-oriented, model-oriented), there is a similar problem. These new technologies and process models pose a challenge to software engineering: building more complex software in distributed teams, at lower cost and time. However, the emphasis on technology and process models often leaves in the background the fact that software engineering is essentially an activity based on people and that the success of a project or product depends on the decisions taken during the engineering phase [01].

Generally, only information about the final decisions of each phase of the software development cycle is stored, because the analysts consider the task of documenting each of the alternatives investigated very costly (in terms of time). However, adequately storing and retrieving this information provide various advantages along the project: better design knowledge, easier maintenance and communication among the project team, better reuse possibility, easier integration of a new member and also a lower possibility to implement an option already discarded in the past.

At first, these advantages leave no doubt that it is worthwhile to incorporate the capture of this information to the software development process. The great question is how to capture and retrieve this information efficiently and with minimal interference to the usual process activities. Thus, this scenario seems to be very adequate to consider the use of Design Rationale, the basic idea of which is to capture and retrieve that kind of information.

2.1 Definition

There are several works related to project analysis and decision management that mention the term Design Rationale to describe the capture of the designers reasons concerning decisions taken during the project development. However, this activity is not limited to the project elaboration phase, but applied to all software development phases. Considering that many surveys about rationale had, in the past, their focus on project activities, the term Design Rationale is the one that predominated and it is also the most often used [10].

Design Rationale refers to the documentation of alternatives, choices and decisions made during the project development process, as well as the reasons to have taken a particular way. Several researchers have already presented their definitions. Below, are some of them.

According to [11], Design Rationales consist of explanations of the relationships between structure, behavior and functionality of artifacts. Some examples are how a structure implements a feature, or how a certain behavior is justified by a structure. Design Rationale also explains the decision-making process.

For Souza et al. [12], Design Rationale is a documentation representation containing the reasons and arguments for a specific artifact. It includes both the selected and the discarded alternatives, evaluated changes and the arguments which led to a decision.

Design Rationale is the information that explains why an artifact is structured the way it is and has a determined behavior [08].

According Hu et al. [13], Design Rationale is the explanation of why an artifact or some part of it has been designed in a particular way. It includes considerations, arguments, changes and decision-making of a project artifact. This information can be valuable and even critical to many people who deal with the artifact.

In Burge, Brown [14], Design Rationale is defined as the decisions taken during the design analysis phase and the reasons that lead to such decisions.
Each definition provides a proper focus to the subject, but, in general, they are similar definitions. Design Rationale is related to information on the reasons, considerations and the justifications for a decision, and also to the alternatives that were considered and, eventually, discarded. This paper uses the term Design Rationale with this meaning.

2.2 Design Rationale Representation Models

A wide variety of approaches of Design Rationale has been proposed. Most differ only in the nomenclature of the nodes and their relationships. According to a great part of the researchers, including Stumpf [05] and Shum [04], the focus is on three models: Issue Based Information System (IBIS), Questions, Options and Criteria (QOC) and Decision Representation Language (DRL). These models are described below.

2.2.1 Issue Based Information System (IBIS)

Historically, the movement of Design Rationale began with the Issue Based Information System (IBIS), described by Kunz and Rittel. It aimed at providing support to groups that were faced with complex problems [15]. The system guided the identification, structuring and decision of questions raised and provided appropriate information to the discussion, creating a decision plan. It was not software related, but it was a method of modeling the general argumentation.
A few years later, Rittel was already convinced that design problems were fundamentally different from the well-defined problems of science and called them wicked problems. These problems could not be solved by means of traditional analytical approaches [16]. Rittel [17] then proposed an argumentative approach to this kind of problems and used the IBIS model to implement this approach.
The IBIS is used to record ideas and relationships during project discussions while they are occurring. It presents a framework on how the issues are discussed. The focus is not on how the problem is resolved, how the alternatives are extracted and evaluated or how to get to a consensus. The IBIS puts more emphasis on the process representation by which decisions are taken. Therefore, the IBIS model aims at providing support to structure the discussion so that the information can be captured and structured, helping developers to solve their issues.
The IBIS has three kinks of nodes:

1. ISSUES: problems under discussion;
2. POSITIONS: possible solutions to problems;
3. ARGUMENTS: favorable or not opinions to the various solutions sought.

An ISSUE may ask, generalize or specialize another ISSUE, resulting in SUBISSUES, each one with its POSITIONS and ARGUMENTS [18]. ISSUES are created and argued by the fact that different views may be taken. ARGUMENTS are built in defense or against the various POSITIONS until the ISSUE is solved. This occurs when the opponents are convinced or when it is possible to implement a procedure for formal decision [15]. These relationships are illustrated in Figure 1.
The ISSUES are the organizational elements of the systems. Among their properties, the following can be mentioned [15]:

- ISSUES have the form of questions;
- The origin of ISSUES can be contradictory statements;
- ISSUES are specific to particular situations. POSITIONS are developed using private information on the problem environment;
- ISSUES are suggested, questioned, specialized, generalized or replaced.

Conklin and Begeman [19] adapted IBIS to be used in software engineering, by creating a hypertext system called graphical IBIS (gIBIS). Its main changes and extensions are:

- Creation of an additional type of node called OTHER, used as an escape mechanism for cases in which it was not possible to find a way to express a consideration in the IBIS model;
- Creation of an additional type of node called EXTERN, which contains non-IBIS material, as requirements documents, design sketches or codes;
- Inclusion of the relationships SPECIALIZES and GENERALIZES between two POSITION or ARGUMENT.

Figure 2 shows the types of nodes and relationships of the extended IBIS model used in the construction of gIBIS. The possible connections include:

- RESPONDS TO: a POSITION can answer to an ISSUE;
- SUPPORTS, OBJECTS TO: the ARGUMENTS can justify or oppose a POSITION;
- GENERALIZES, SPECIALIZES: ISSUES can generalize or specialize another ISSUE;
- QUESTIONS, IS SUGGGESTED BY: ISSUES can question or be questioned by other ISSUES, POSITIONS or ARGUMENTS;
- REPLACES: an ISSUE may be replaced by another ISSUE;
- OTHER: as an escape mechanism, the OTHER node can be connected with any other node by means of the connector OTHER.

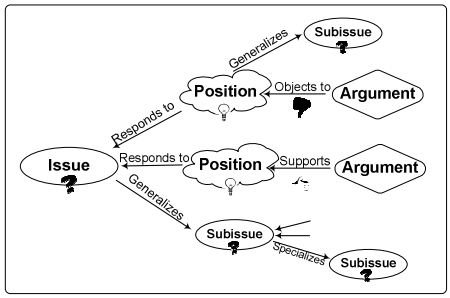

Figure 1 - IBIS: Nodes Representation based in Kunz and Rittel [15]

Conklin and Begeman [19] define gIBIS as a hypertext system that uses colors and a relational database server of high performance to facilitate the construction and research of IBIS networks.

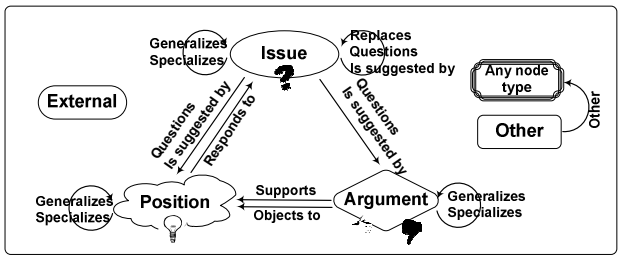

Figure 2 - gIBIS: IBIS Extension

2.2.2 Questions, Options and Criteria (QOC)

Developed by MacLean et al. [22], this model emphasizes the exploitation of the project alternatives and the choices made. It is an approach that uses semiformal notation based on reasoning to systematically represent visions of the project space. The project space is a set of relationships or conceptual dimensions used to compare projects and alternative solutions [04].

The model presents a framework based on the process of how the alternatives are generated and evaluated. This organization allows the explicit representation of a structured space of project alternatives and the considerations that led to its choice. In the design options space answers to the questions are possible [23].

The model includes three types of nodes: QUESTIONS, OPTIONS and CRITERIA. The QUESTIONS are the key problems to be solved, the OPTIONS are the alternatives raised to solve the problems and the CRITERIA justify the existing options. In addition to these elements, there are also the assessments, which are the relationships between OPTIONS and CRITERIA. In its simplest form, the assessments may have two kinds of relationships: POSITIVE ASSESSMENT or NEGATIVE ASSESSMENT. Figure 3 shows the general structure of the QOC model.

In summary, the model aims to identify key problems (QUESTIONS), raise and justify (via CRITERIA) the project alternatives (OPTIONS).

In the QOC, the criteria are stored, allowing the explicit capture of project restrictions and facilitating the problem resolution, which does not occur in the IBIS.

According to Hu et al. [13], QOC is simple to learn and to use, and there is a growing number of research projects using it. Another advantage highlighted by the authors is that it is relatively easy to create a QOC for reverse engineering of the system, preserving it for future use.

2.2.3 Decision Representation Language (DRL)

Lee and Lai [24] proposed a notation that is an extension of the QOC and gIBIS models. As its name says, Decision Representation Language (DRL) has been developed as a language to represent decision-making. One of the main concerns was the increase in expressiveness and functionality. To cover these features, there was an increase in complexity [20].

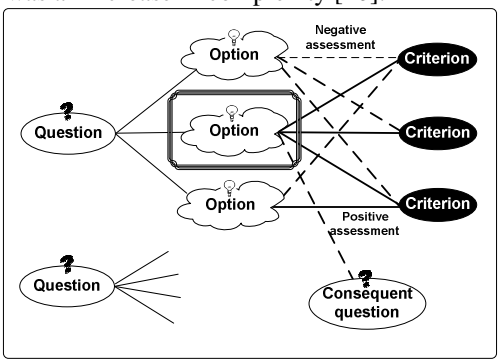

Figure 3 - QOC Model, based on the description of MacLean [22]

DRL is a more complete argumentation model, involving a larger number of nodes and relationships than the other two models. Jarczyk, Loffler and Shipman [23] relate the main types of nodes (ALTERNATIVE, GOAL, CLAIM, QUESTION and PROCEDURE), of a hierarchy with 26 node types and more than 20 relationships.

For Louridas and Loucopoulos [20], there are 5 main entities: DECISION PROBLEM, GOAL, ALTERNATIVE, CLAIM and QUESTION. Lee [25] includes the nodes PROCEDURE and GROUP. The functions of these entities are:

- DECISION PROBLEMS and GOALS: a DECISION PROBLEM is a controversial subject in the project and a GOAL is a set of requirements (criteria) to fix it;
- ALTERNATIVES represent possible solutions to DECISION PROBLEMS;

- CLAIMS are used to argument. Considering that all relationships are subclasses of the CLAIM object, an argumentation can be based on other one;
- QUESTIONS are used to lead the discussions during the project.
- PROCEDURES are steps that should be taken to obtain answers to a question. They represent auxiliary aspects in the decision-making process.

The main relationships between the nodes of the DRL model are [25]:

- A DECISION PROBLEM "is a subdecision of" another DECISION PROBLEM;
- A GOAL "is a subgoal of" a DECISION PROBLEM;
- A CLAIM "supports" or "denies" an ALTERNATIVE or another CLAIM;
- A CLAIM "answers" a QUESTION;
- A QUESTION "queries" or "influences" a CLAIM;
- A PROCEDURE "is an answering procedure for" a QUESTION;
- A PROCEDURE "is a subprocedure of" another PROCEDURE;

Figure 4 illustrates the DRL model, showing the possible relationships between the nodes.

This approach was used in the development of a knowledge-based tool called SIBYL to represent justifications for project decisions [26]. First, instances of a DECISION PROBLEM are created. Then the GOALS and ALTERNATIVES are added [27]. Through appropriate modules, the evaluation and reasoning process of this information are carried out.

The tool provides various services including the dependency management (tracking of decisions that have dependencies among themselves), precedence management (other decisions share the same goals), point of view management (arguments share the same assumptions) and plausibility management (the power to support the argumentation of an alternative).

Lee and Lai [24] argue that a representation must support a variety of project tasks, such as answering questions about the progress of the project, the alternatives generated, the assessments that led to the choice of certain alternatives and the possible transfer of knowledge to future projects or even other people that come to work in the proper project. DRL was developed to support all these issues. Its emphasis is to manage the qualitative elements of decision-making and their dependencies.

In the next item, a comparative analysis of these three models is performed.

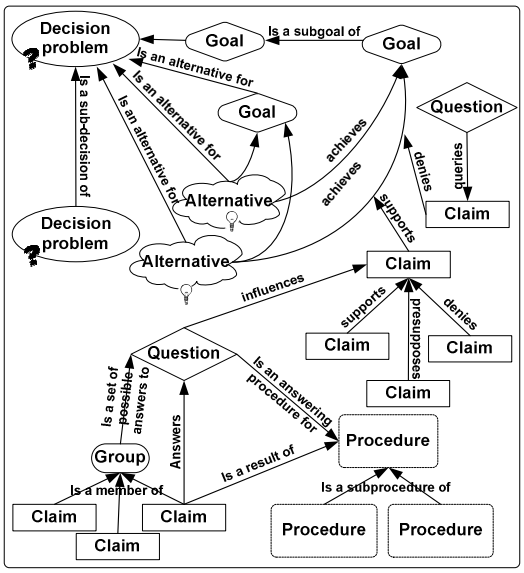

Figure 4 - DRL Model, based on Lee [24]

2.2.4 Comparison between Models

The three models are based on arguments and have semiformal notation. They differ in the extent of features that they seek to capture and on the way these models are used in a representation.

In general, the three models have the two nodes referent to the problems and alternatives of solutions. Each model uses its own nomenclature to make references to these nodes. IBIS and DRL have a node that defines the arguments for the choice or rejection of an alternative. QOC and DRL have a node that indicates the criteria for the solution of a problem. This feature is important for a better and more accurate decision-making [21], because these criteria indicate desired properties or requirements that must be met. Moreover, QOC does not explicitly present information relating to the arguments, which end up being stored implicitly in the node OPTION. DRL has a larger number of nodes and relationships types, providing greater flexibility, but making its understanding more complex.

Depending on the purpose of the model, it can be classified as descriptive or prescriptive. A model is said to be descriptive if the goal is only to describe the designers' reasoning processes. No attempt is made to change the designer's way of thinking or even the decisions taken. However, the information stored can improve procedures for the other phases of the software developing life cycle, such as the development, maintenance or reuse of project artifacts. Design Rationale can also be used to transfer knowledge to new team members.

The prescriptive approach, on the other hand, aims to improve the process during the project design, improving the way of thinking of the people involved. One goal is to correct perceived deficiencies in the reasoning of project issues, making it more correct,

consistent and complete. In this approach, records of Design Rationale can also be created to assist the processes outside the project design stage. It should also be noted that descriptive and prescriptive approaches are not always mutually exclusive. For example, some approaches have the primary intention of being descriptive, but also have some prescriptive goals [01].

Considering the basic objectives of the QOC and DRL models, they are classified as descriptive and IBIS is classified as prescriptive. QOC is fundamentally descriptive since its primary purpose is to create a designer's decisions representation that is sufficiently detailed to provide information to other phases of the development life cycle of an artifact.

For Dutoit et al. [01], two features distinguish QOC from IBIS. The first refers to the questions scope. While IBIS can be related to any project topic, the issues of the QOC model deal exclusively with features of the artifact being designed. Therefore, the issues of the QOC model always have possible answers (OPTIONS), which describe the properties of the artifact being designed. The second factor that distinguishes the two models is that the QOC uses judgments to indicate whether the alternatives meet each criterion set. Although it is possible to represent these judgments in the IBIS model, by means of arguments, IBIS has not an explicit representation of the criteria as model elements. Nguyen, Swatman and Shanks [18] complement this, emphasizing that each alternative is evaluated individually by its own arguments. There is not a common set of arguments to all alternatives.

Another limitation of the IBIS model is the generation of a complex network when there are a large number of nodes, hampering the search for data [21].

The DRL model is much more complete, involving a larger number of nodes and relationships than the other two models. One of the goals was to increase the expressiveness and functionality. Lee and Lai [02] argue that the DRL is more expressive than the other models because it serves a broader range of issues. On the other hand, to accommodate these features, there was an increase in complexity [20].

This comparative analysis between the models provided the basis for the proposal presented in the next section. Considering the theoretical analysis, DRL seems to be very complex to be used in ERP systems. The best solution is likely to be based on the QOC or IBIS model.

3. Proposal of a Design Rationale Representation Model for Project of ERP Systems

This chapter contains the main proposal of this work. Initially, an analysis of the use of Design Rationale in ERP systems is conducted. Then, the search made is described, presenting a graphical representation of the Design Rationale for each case. Finally, a proposal for a Design Rationale representation model is presented for ERP systems projects.

3.1Design Rationale and ERP

This item contains an applicability analysis of the Design Rationale for ERP systems projects. The purpose is to confirm that the Design Rationale is useful for these types of software systems.

If the information about the justification of project decisions is available, it is possible to use it in various ways aiming at supporting the project activities. Gruber and Russel [28] list some of these activities:

- Checking work: as the information captured explains certain parts of the project, it is possible to carry out checks in order to detect inconsistencies;
- Clarifications: Design Rationale provides important information about the project. It is very useful especially in complex projects;
- Information sharing: a history of the project can avoid trying to use alternatives already exploited. This is especially important in projects in which people that have not participated in the initial phases of the project should define corrections or add new features to the system;
- Institutional Memory: in large organizations, many designers are working concurrently in several projects, over long periods of time. They need to share common artifact models, communicate with each other and with other departments. Capturing knowledge through Design Rationale, in an explicit format, it is important to accumulate and share knowledge within the organization.

  Looking up a project for ERP system, it is observed that it has several features that propitiate an adequate use of Design Rationale.

The ERP systems are developed over a long period, and generally by large groups of people. The software usually suffers systemic and technological changes starting from the current version, i.e., newer system versions are built on the basis of the previous ones. Considering that such systems have a considerable legacy (many code lines written), it is difficult to use a new technology that does not take into account the existing system. Consequently, although a version is considered a new project, in fact, it is dependent on all previous versions.

Therefore, the design of a release, in a more global view, covers the design of all previous versions. A new feature or changing something that already exists may need information from the initial system design, possibly carried out several years ago.

The life cycle of this kind of software differs from the others, because changes are constantly made in the system (for corrections, legal changes, new features, etc.). Usually there are various updates between one version and another. Moreover, a feature of this kind of

system is the need of customization to meet the specific needs of each client. This activity is considered another project, made by another team of the company or even by a team of another company.

It is very common that consulting firms carry out the ERP implementation and customization. Information on the ERP projects decisions can be valuable to this phase, avoiding inconsistencies, unnecessary work and attempts to test alternatives already discarded. The most recent surveys suggest that the lack of knowledge of the ERP system has been a matter of great importance to many organizations [29].

It is not unusual for the designer of a particular routine not to be present when it is necessary to make any changes in its functionality. This is due to two main factors: long software life cycle and high turnover of people in this area.

For all these features, the project of an ERP system is a very suitable case to the use of Design Rationale. It is a complex project, which undergoes a lot of changes, has a long life cycle and involves a large number of people. Consequently, it will be very useful if it is possible to store information in order to carry out project verifications or to provide clarifications to the people involved..

The documentation of project decisions generates a history, facilitating the sharing and accumulation of information. One benefit is to avoid an attempt to use alternatives already discarded in the past. This information will be important in all the life cycle phases of the ERP.

Additionally, an ERP system has an implementation phase, in which customizations are made for each company. This is the most critical phase, which often generates dissatisfaction of both the implementing team and of the people from the client company. Recent researches revealed a significant reduction in the level of implementation satisfaction of ERP systems in the period from 1998 to 2000 [30]. It is believed that much of the problem is caused by lack of information and communication. Design Rationale can contribute to solve such problems.

It is very difficult to anticipate all project decisions that will be questioned [11]. Therefore, the use of Design Rationale should have the ability to answer a lot of questions about the project. Another tactic is to focus on the information on a specific part of the project. This strategy can be more effective in cases in which there is a history of previous projects and it is possible to predict which parts of the project are the most likely to have their information questioned.

The latter option seems to be very appropriate for ERP projects. By means of the history, it is usually possible to determine which modules suffer more updates, which functions have more problems, which are the critical processes, etc. Then the focus can be put on specific parts of the project.

3.2 Real Cases Analysis

In this item, some actual cases of decisions on the design of an ERP system are presented. The first attempts to collect data were made by means of e-mails to the developing coordinators of ERP systems of three large companies. From a total of thirty three people, only one responded to the request. Realizing that the problem was the understanding of what was being requested, the data collection began to be made by means of informal interviews.

In general, the interviewees showed some difficulty in remembering examples. Even when a case was remembered, not all alternatives or reasons for the decision were remembered. Nearly half of those interviewed requested assistance from another person in the team to remember some detail. In total, thirteen people were interviewed. In a poll conducted by Tang et al. [31], 74% of interviewees said that they forgot the reasons related to the project decisions.

After the interviews, twenty-two cases were collected. Four cases were selected to be detailed in this paper. The others have a very similar representation and, practically, do not add any extra information to the proposed model. Then these four cases of decision of ERP system are presented.

3.2.1 Case 1: Credit Analysis

The Commercial Automation module manages from the operational control related to the cash register and attendance, to the financial management of inventories and purchases.

In this module, there is a credit analysis process that is conducted at the moment of paying for the goods purchased, if the operation is performed by check. It consists of the status verification of the customer credit in the market.

During the project analysis, two alternatives were considered to the implementation of this process:

1. Query via Web Service: each terminal that needs information sends a process calling for a query. This process may take some time. A new query request can only be sent when the first one returns. This solution is simple to implement and provides security, preventing a purchase from being made without any credit verification. Moreover, a customer may happen to switch to a payment method that does not require credit verification (cash, for example). In this solution, after the credit analysis query is requested, it is necessary to wait for the answer to finalize the purchase. Moreover, other people may be waiting to make a credit analysis and will have to wait for the credit analysis query, the result of which will not even be used.

2. Local Semaphore: rather than simply sending the requests, the system implements a local query queue. If a request for credit analysis is cancelled, the system checks whether the request is in the queue. If not, nothing is done, indicating that the query is being processed. If yes, the query is removed and other queries take place. In this case, it is necessary to implement this queue logic, but it is a more flexible solution, with better performance.

Figure 5 shows a graphical representation of the problem using the IBIS model. While it is possible to represent all information in the picture, it is unclear which is the chosen alternative and for what reason, since the two alternatives have the same number of positive and negative arguments.

Normally, an argument is more important than another. Considering such a fact, an alternative can support fewer arguments that another one, but even so, it may be the one chosen. This can happen because some arguments can be more important than others.

In this case, the option chosen was the use of a local semaphore, optimizing the process.

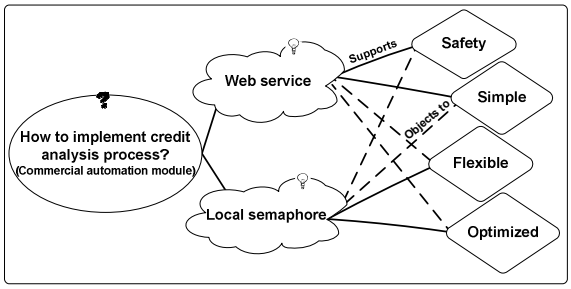

Figure 5 - Design Rationale Representation for Case 1

3.2.2 Case 2: Human Resources – Note Table

In the Human Resources module, there is a routine that makes the notes of entry and exit of each employee. Basically, a comparison between an official table with the work schedules pattern is made and the notes of entries and exits of each employee. Based on this analysis, it is possible to calculate discounts and extra hours at the end of the month.

Considering a large data volume, this routine must be accomplished with very good performance. The main point is the definition of the processing method of these tables:

1. Physical Table: the data are in a table in the database. Despite the direct processing being slower than having the data in memory, it is possible to take advantage of several features of the database management system (integrity, triggers, indexes, etc.).
2. Data in memory: the data are read from the database and are in memory during analysis. This option requires extra implementations, such as data loading, data validation and sorting. The advantage is the access speed to data after they are ready to be processed.

Figure 6 shows the representation of the Design Rationale for this case. The main question creates another issue, which is how to get the best performance.

Initially, the alternative "data in memory" has been chosen. This case does not add any new element unless the specialization of the question lies in another issue - “How to get better performance?”

The interesting point in this case is that the interviewees reported that today the choice is no longer the best. The increase in functions to handle exceptions made the choice of data in memory slower. Thus, the alternative “physical table" became the best solution.

3.2.3 Case 3: Business Process Management (BPM)

The goal of Business Process Management (BPM) is providing greater visibility and transparency of procedures and consequent possibility of monitoring and auditing. One of the features of this tool is to provide flexibility for changes and enhancements according to demand.

The Business Process Management (BPM) allows processes modeling, the rules parameterization of business associated with the flow, definition and restriction of access for users to consider the skills and activities to be undertaken and the control over process application features (time and process cost). Moreover, it allows the development of computer applications, called the fourth software layer, integrated or not legacy systems, such as: proprietary systems, ERP, CRM, telephony, and other technologies.

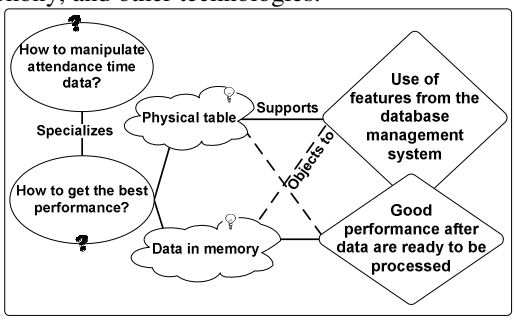

Figure 6 - Design Rationale Representation for Case 2

Considering the use of BPM with ERP, an external software tool can be used, or the functionality can be integrated to the system. One of the ERPs analyzed decided for its integration to ERP standard.

During the analysis phase of this module, one of the main decisions was to choose the data entry method in the system. Basically there were two options:

1. Pre-defined model: the model containing all the system features is defined by programming, and cannot be changed during implementation. So it is easier to implement, but less flexible, requiring intervention of the IT area if it is necessary to make

some changes. In this model, the system will ask the information to the user as he goes through the screens.

2. Dynamic Model: the definition of the data entry model is made by the company manager during the system implementation. After the completion of this activity, the environment is ready for the information input. The idea is that this manager understands the business of the company and he does not necessarily need to understand computing. Consequently, the model is dynamically defined and can be modified without the intervention of the IT staff. The implementation of this solution is more complex and requires a validation of the user defined model. However, this is the solution that allows the creation of a system with features that follow the Business Process Management (BPM) concepts.

The representation of this case in the Design Rationale in the IBIS model is shown in Figure 7. In the figure, it is not possible to define which the best alternative is, since each supports two arguments and is opposed to the other two.

The solution of this issue was made by the dynamic model implementation.

3.3 Model Definition

This item presents the proposal for a model to represent the Design Rationale considering ERP systems. Hu et al [13] claim that a good representation model is essential to an effective recovery.

The proposed model is intended to capture the information of Design Rationale in the project analysis phase of the development cycle of an ERP system. Information recovery can be made throughout the software life cycle, including the project analysis phase, the implementation phase of the system and the development of other product versions.

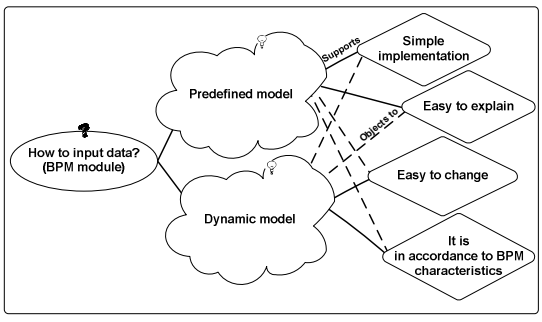

Figure 7 – Design Rationale Representation for Case 3

The goal is to have a model that could represent the main elements found in the decisions of ERP systems, which were, at the same time, simple and intuitive. Moreover, the objective was proposing a model that could be helpful during the project analysis, helping the discussions and choices and also contribute to the other system life cycle phases, providing a more complete documentation. To meet this criterion, the graphical representation should explain the choice of a particular alternative.

The DRL was discarded because its emphasis is on the decision-making management and its dependencies. Its complexity hinders its application in larger projects and with large number of people such as ERP systems. The high turnover is another obstacle to adopt this model. The training time necessary to understand the new technology can make the adoption of this model infeasible. Then, IBIS and QOC seemed to be the most appropriate models for this case.

In Case 1, it was verified that the IBIS was the model that best represented the example characteristics. The main reasons for choosing this model are that the issues may refer to any project question and not just the artifact being designed. The graphic representation was validated, after presenting it to the proper people interviewed.

For the majority of cases, it was found that only the IBIS model was not sufficient to present a self-explanatory model, especially with respect to the chosen alternative. Taking into account only the number of arguments that are favorable to an alternative, there is a risk of not choosing the best one. The solution was to set weights for each argument. These weights define the importance degree of the arguments, considering the problem. The range of valid values goes from zero to ten, and the higher the number, the greater its importance.

Thus, the model could represent a very common feature in the cases reported: each argument has a different importance for the issue solution. Figure 8 shows the IBIS model with weights in the arguments.

Finally, after a review of cases such as that presented in section 3.2.4 (BPM), one limitation of the IBIS model described in the literature was confirmed: the lack of explicit representation of the criteria as model elements [18]. The inclusion of a specific element to represent a criterion (requirement or restriction) eliminated this limitation. This information, which is present in the QOC model, turns the model more complete. Figure 9 illustrates the final proposal of the Design Rationale model for ERP systems. This model was established through the combination of features of the IBIS and QOC models and the inclusion of a quantification attribute about the importance of the arguments.

It was possible to adequately represent all the decision cases of ERP systems considered in the research by means of this model. The graphic representation of this new model for the four cases presented in Section 3.2 is depicted below.

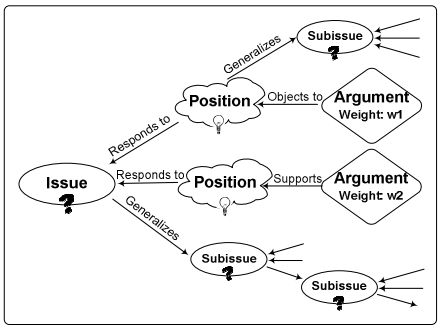

Figure 8 - Design Rationale Model for ERP Systems: Inclusion of the attribute "Weight"

3.4 Model Application to the Real Cases

This section contains the application of the proposed model in the previous real cases presented in section 3.2.

3.4.1 Case 1: Credit Analysis

Figure 10 shows a graphical representation of the problem using the new model. To represent the difference in importance between the arguments, weights are used. Thus, it can be seen why the alternative of a local semaphore was chosen.

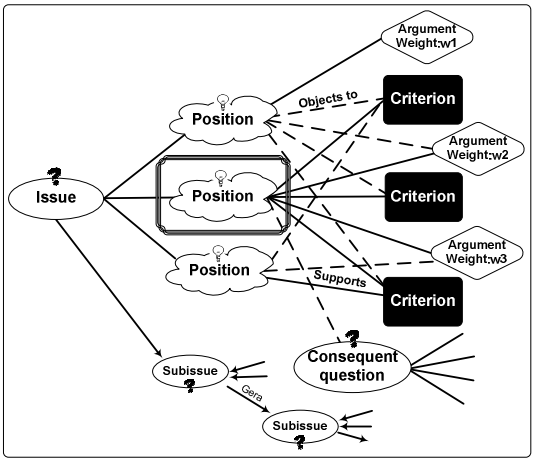

Figure 9 - Proposal of a Design Rationale Model for ERP Systems

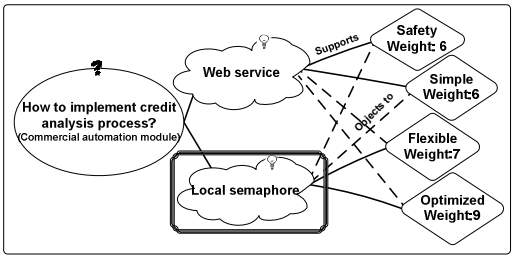

Figure 10 – Design Rationale Representation for Case 1

3.4.2 Case 2: Human Resources – Note Table

Figure 11 shows the Design Rationale representation for the case of the note table. Considering that the second argument is the one that provides the best answer to the question (Weight: 8), the alternative "data in memory" has been chosen.

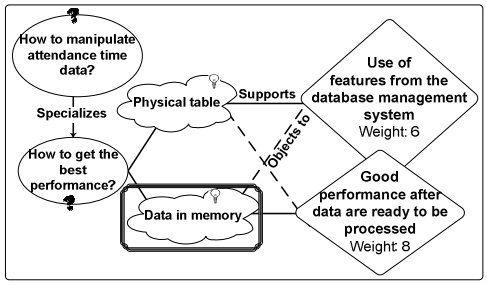

Figure 11 - Design Rationale Representation for Case 2

As mentioned earlier, the increase in functions to handle exceptions just made the choice of data in memory slower. To reflect this change in the graphical representation, it is only necessary to change the "weight" of the corresponding argument, as illustrated in Figure 12. Thus, the alternative "Physical Table" seems to be the best solution.

This case reinforces the usefulness of having the property "weight" in the arguments indicating its importance in the solution of the issues. It is essential to understanding changes in the chosen alternatives.

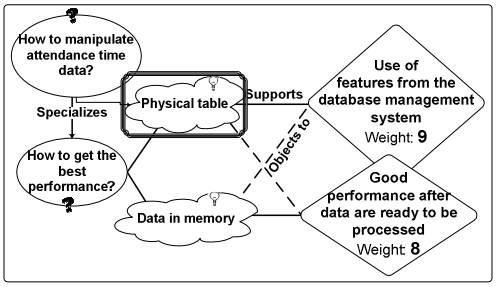

Figure 12 - Design Rationale Representation for Case 2 – Current Situation

3.4.3 Case 3: Business Process Management (BPM)

The Design Rationale representation of the reported case in the new model is represented in Figure 13. The figure can be more easily interpreted with the representation of the node Criterion. The node "Agree with the BPM features" is a system requirement and, therefore, the solution must support this item. Therefore, the "dynamic model" is the solution to the problem.

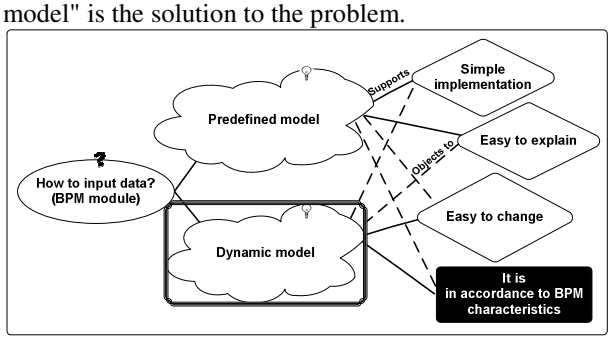

Figure 13 - Design Rationale Representation for Case 3

3. Conclusions

For many Design Rationale researchers, the value of capturing information on the project decisions meant that this technique would be quickly disseminated in the companies. Furthermore, the academic research on the subject continued growing.

While there are several surveys on Design Rationale and on Enterprise Resource Planning (ERP), separately, no study was found on the combination of the two issues. Thus, the study on the Design Rationale application in ERP Systems demonstrated the enormous value of its use. The project justifications can be documented in a complete and accurate way, particularly helping the designers of this kind of system.

Another important factor is the observation of some deficiencies found in two modeling forms of Design Rationale, which allowed the proposition of an adaptation to the way of modeling, which showed to be quite appropriate to record the reasons for designing ERP systems. The proposed model for ERP systems, albeit simple, serves all of the cases collected.

After the elaboration of the Design Rationale proposed model, it was possible to see that the greatest difficulty is not in the definition of processes or models. The technical challenges are small when compared with the human challenges.

It should be noted that the proposed model aimed to capture information from Design Rationale in the preparation phase of the project development cycle of an ERP system. Information recovery can be made throughout the system life cycle, including the project elaboration phase, in addition to the deployment phase, as well as the development of other product versions.

The proposed model is capable of representing models that correspond to the main elements found in the decisions of ERP systems, assisting in the project major decisions, providing an even more complete documentation.

References

[01] DUTOIT, A. et al. **Rationale Management in Software Engineering**, Editors, Springer, 2006.

[02] LEE, J.; LAI, K. **What's in Design Rationale?** in Design Rationale - Concepts, Techniques, and Use, T. Moran and J. Carroll, Eds. New Jersey: Lawrence Erlbaum, p. 21-51, 1996.

[03] ROSSI, M. et al. **Method Rationale in Method Engineering**. Proceedings of the HICSS-33, Maui, HI, IEEE Computer Society, 2000.

[04] SHUM, S. B. **Cognitive Dimensions of Design Rationale**. In D. Diaper, & N. Hammond (Eds.), People and Computers VI, Proceedings of HCI'91, Cambridge University Press, 1991.

[05] STUMPF, S. C. **Argumentation-based Design Rationale -** the Sharpest Tools in the Box. Research Note RN/98/103, Computer Science Department, University College London, University of London, U.K., Available in: http://www.cs.ucl.ac.uk/staff/S.Stumpf/Reports/IN9801.html Access in: 10/02/2006.

[06] SHUM, S. B.; HAMMOND, N. **Argumentation-Based Design Rationale: What Use at What Cost?** International Journal of Human-Computer Studies, p. 603-652, 1994.

[07] FISCHER, G. et. al. **Making argumentation serve design**. In T. Moran and J. Carroll, editors, Design Rationale Concepts, Techniques, and Use, p. 267-294, 1995.

[08] CONKLIN, J.; BEGEMAN, M. L. **gIBIS: A Hypertext Tool for Exploratory Policy Discussion**. Proceedings of the 1988 Conference on Computer Supported Cooperative Work (CSCW-88), ACM, Portland, Oregon, p. 140-152, 1988.

[09] BOSCH, J. **Software Architecture: The Next Step**, Software Architecture: First European Workshop, EWSA 2004, St Andrews, UK., p 194-199, 2004.

[10] SAUER, T. **Using Design Rationales for Agile Documentation**. Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises 2003 (WETICE'03), 2003.

[11] GRUBER, T. R.; RUSSEL, D. M. **Design Knowledge and Design Rationale: A framework for representation, capture, and use**. Technical Report KSL 90-45, Knowledge Systems Laboratory, Stanford, California, 40p, 1991.

[12] SOUZA, C. R. B. et al. **A Model Tool for Semi-automatic Recording of Design Rationale in Software Diagrams**. In Proceedings of the String Processing and Information Retrieval Symposium, p. 306–313, 1998.

[13] HU, X. et al. **A Survey on Design Rationale**: Representation, Capture and Retrieval. Engineering with Computers: An Int'l Journal for Simulation-Based Engineering, v. 16, p. 209–235, 2000.

[14] BURGE, J. E.; BROWN, D. C. **Reasoning with Design Rationale.** In Proceedings of the Artificial Intelligence Design Conference, 2000.

[15] KUNZ, W.; RITTEL, W. J. **Issues as Elements of Information Systems**, Working Paper No 131, University of California, Berkeley, 1970, reprinted in 1979. Disponível em: http://www-iurd.ced. berkeley. edu/pub/WP-131.pdf. Access in: 06/12/2005.

[16] RITTEL, H.; WEBBER, M. **Dilemmas in a General Theory of Planning**, Policy Science, Vol. 4, p. 155-169, 1973.

[17] RITTEL, H. **On the Planning Crisis: Systems Analysis of the 'First and Second Generations'**, in Proceedings of the Systems Analysis Seminar, European Association of National Productivity Centers, 1972.

[18] NGUYEN, L.; SWATMAN, P. A.; SHANKS, G. **Supplementing Process-Oriented with Structure-Oriented Design Explanation within Formal Object-oriented Method**. Software Engineering Conference, 1998. Proceedings. 1998 Australian, p.118-132, 1998.

[19] CONKLIN, J.; BURGESS-YAKEMOVIC, K. **A Process-Oriented Approach to Design Rationale**, in Design Rationale Concepts, Techniques, and Use, T. Moran and J. Carroll, (editors), Lawrence Erlbaum Associates, Mahwah, NJ, p. 293-428, 1995.

[20] LOURIDAS, P.; LOUCOPOULOS, P. **A Generic Model for Reflective Design**, ACM Transactions on Software Engineering and Methodology (TOSEM), v.9 n.2, p. 199-237, 2000.

[21] FRANCISCO, S. D. **DocRationale: uma ferramenta para suporte a Design Rationale de artefatos de software**. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo, São Carlos - SP, 2004.

[22] MACLEAN, A. et al. **Questions, options and criteria: Elements of design space analysis**. Human-Computer Interaction, p. 201-250, 1991.

[23] JARCZYK, A.; LOFFLER, P.; SHIPMAN, F. **Design Rationale for Software Engineering:** A Survey: In Proceedings of 25th Annual Hawaii International Conference on System Sciences, p. 8-10, 1992.

[24] LEE, J.; LAI, K. **A Comparative Analysis of Design Rationale Representations**, Human-Computer Interaction Special Issue on Design Rationale, p. 251-280, 1991.

[25] LEE, J. **SIBYL: A qualitative design management system**. In P.H. Winston and S. Shellard, eds., Artificial Intelligence at MIT: Expanding Frontiers, Cambridge MA: MIT Press, p. 104-133, 1990.

[26] LEE, J. **SIBYL: A Tool for Managing Group Decision Rationale**. In Proceedings of the CSCW'90 Conference, ACM Press, New York, p. 79-92, 1990.

[27] MONK, S.; SOMMERVILLE, I.; PENDARIES, J. M.; DURIN, B. **Supporting Design Rationale for System Evaluation**. In: Proceedings of the Fifth European Software Engineering Conference, Barcelona, Espanha, p. 307–323, 1995.

[28] GRUBER, T. R. **Model-based Explanation of Design Rationale**, in Proceedings of the AAAI-90 Explanation Workshop, Boston, July 30, 1990.

[29] DAVENPORT, T. H. Mission Critical – Realizing the Promise of Enterprise Systems, Boston, MA: Harvard Business School Press, 2000.

[30] SKOK, W.; LEGGE, M. Evaluating enterprise resource planning (ERP) systems using an interpretive approach, Proceedings of the 2001 ACM SIGCPR conference on Computer personnel research, San Diego, California, USA, p.189-197, 2001.

[31] TANG, M. et al. A Survey of the Use and Documentation of Architecture Design Rationale, 5th Working IEEE/IFIP Conference on Software Architecture (WICSA 2005), p. 89-98, 2005.