A Case-based Approach for Reusing Decisions in the Software Development Process

Hércules Antonio do Prado[1,2], Edilson Ferneda[1], Aluizio Haendchen Filho[3] and Sandra Silva de Alvarenga[4]
[1]MGCTI, Catholic University of Brasilia, Brasília, Brazil
[2] Brazilian Agricultural Research Corporation, Brasília, Brazil
[3] UNIFEBE, Brusque, Brazil
[4]UniCEUB, Brasília, Brazil

Keywords: Software Engineering, Design Rationale, Case-Based Reasoning, Knowledge Management.

Abstract: This paper proposes a process for supporting reuse of decisions during the software development process, involving architectural, technological, or management issues, in order to help reducing time and costs in process. A survey with software engineering professionals was performed aiming at identifying a set of decision-making cases that could be applied to design the process. From the result of this survey, a process was implemented, including related software and procedures that eases the reuse of decisions made during software development projects. Design Rationale techniques were applied to structure the cases that were represented and recovered by means of a Case-Based Reasoning approach. The applicability of this approach was evaluated by means of a two-phase case study. The first one encompassed the construction of the case base using the cases identified previously and the second was focused in the application of the system and its evaluation by means of group dynamics. The focal group was chosen among a set of software engineering experts from companies and universities located in Brasília, the Brazilian capital. Satisfactory results were found with respect to the usefulness of the model to improve the performance of software development when past cases are available.

1 INTRODUCTION

The software development process involves the use of highly specialized and expensive technical knowledge. This knowledge can represent important asset if properly registered and made available for reuse. Usually, the development of an activity can benefit from the experience of past decisions, avoiding the re-work or even the adoption of solutions that are not the best for the same class of problems.

However, it is not usual to adopt a systematic approach for this purpose. Usually, these activities are carried out occasionally and without reference to the context of the problem. Indeed, documenting decision making throughout the software lifecycle can be costly or have limited success chances if adequate support is not available for the management of the involved knowledge.

Although the widely recognized importance of reuse in Software Engineering (SE) (Gopalakrishnan, 2015, Moaven et al., 2008), the available solutions do not retain the context of the design decisions or even the course taken by the software engineer in the formulation of solutions. Usually, the approach is to design components with broad spectrum of application that are difficult to apply to specific problems. The treatment of the knowledge involved in the construction of components for reuse is still a problem without a largely accepted solution.

Misleading and re-working should be mitigated to the maximum, and successful solutions should serve as the basis for new problems solving. An aggravating factor is that in general, the knowledge involved is dispersed, large and very dynamic (Parreiras e Bax, 2003).

During the software development process, many solutions can be devised for a given problem situation. Many are the arguments involved in the discussions for the definition of an alternative. What is sought is a narrative based on well-defined criteria

Prado, H., Ferneda, E., Filho, A. and Alvarenga, S.
A Case-based Approach for Reusing Decisions in the Software Development Process.
DOI: 10.5220/0006356606250631
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 1*, pages 625-631
ISBN: 978-989-758-247-9

as a rationale for the choice of a way forward. There are many risk situations for these knowledge assets. Explanations offered by users to understand a demand and the knowledge that is lost by employee turnover can be cited as examples. All this knowledge asset can be lost by failing in documenting the solutions to the problems. (Burge, Brown, 2000).

Despite the many facilities provided in the realm of SE to document information related to software projects, the non explicitation of tacit knowledge involved in decision-making weakens the overall process. In addition, companies are increasingly concerned with reaching higher levels of capacity maturity models, spending much more time with documentation, metrics, and indicators than simply writing and debugging codes (Burge, 2008).

This paper proposes an environment for representation, storage and retrieval of the knowledge involved in the software development process, in order to: (i) avoid the repetition of past errors and (ii) evolve products based on previous successful decisions.

2 RELATED WORKS

Considering the literature regarding to concrete technological solutions for decision reuse in SE, Conklin and Begeman (1988) presented gIBIS (Graphical Issue-Based Information System), proposal to capture design deliberations in their early stages. The system uses IBIS notation for argumentation, focusing on the collaborative working context. Afterwards, a graphical representation was incorporated into the IBIS vocabulary by means of a directional graph that shows the contents nodes. It allows a hierarchical view of the information, being able to represent problems, answers and arguments.

Rus, Lindvall, and Sinha (2001) take as start point the fact that the software development process involves several profiles of professionals that need to interact and decide on a variety of options. In this process, timely and quality solutions must be sought, while keeping a good tradeoff between the cost-time to obtain a solution.

DocRationale (Francisco, 2004) is a tool for supporting the capture, representation, and retrieval of software artifacts based on Design Rationale (DR) (Shum, 1991). The idea is the collaboration among members of development teams, that register solutions for later consultation in different projects. DocRationale promotes the many forms of digital communication (audio, video and e-mail files, among others) in order to complement rationale documentation, all of which are captured hierarchically and chronologically ordered. In order to provide DR support for software artifacts, DocRationale is process-oriented, enabling the maintenance of phases, activities, and artifacts of a project.

InfoRat (Inference Over Rationale) also uses DR for knowledge representation and is able to make inferences on a particular design, to detect inconsistencies and estimate the impact of changes (Burge, Brown, 2000). The tool is designed to be used along with the Eclipse development environment. It provides an ontology criteria visualization for evaluation and selection of alternatives. The capture is performed manually and separately from the design process, adding a high cost to the software process.

Based on the Decision Rationale Language (DRL), which allows the representation of rationale decisions, SYBIL (Lee, 1990) is a system that aims to assist users in the management and representation of the qualitative aspects involved in the decision-making process. The decision-making tasks are supported by graphs that allow the visualization of different alternatives involved and their respective evaluations. SYBIL eases the user interaction with the environment by means of a visual user interface. It enables the management of dependency, precedence, and evaluation, supporting decision-making based on information quality.

SEURAT (Software Engineering Using RATionale) (Burge, Brown, 2004) supports the use of DR in software maintenance. It provides an overview of the rationale and possible inferences pointing to unresolved or inconsistent issues resulting from software modifications. This system adopts RATSpeak, a DRL-based representation, which allows the generation of an arguments´ ontology, organized in a hierarchy of constraints that can be applied to a software. SEURAT is integrated with the Eclipse development environment.

Aiming to assist the teaching of fresh students in the software development field, Analogus (Santos Jr., 2009) was developed in a virtual environment that acts in the resolution of programming problems. This environment applies the case-based framework jColibri to suggest programming problems similar to the current one. In addition, to simulate a virtual teacher an intelligent dialogue agent was embodied in the solution. Departing from a set of information about a programming problem reported by a student, the system searches the case base for similar

problems already solved. This set of cases is made available to the intelligent agent that presents similar aspects of the cases to the student. The student solves the new problem by reusing the knowledge presented by the agent. A teacher checks the resolution proposed by the student and verifies the need for adjustments. A problem is considered solved and made available when all of the teacher's considerations are met by student.

ECoCADe (Evidence, Context, and Decision Support Cases) is a framework to support evidence-based decision making, mainly by assisting developers in modeling evidence and case representation. ModECoCa (Context Evidence Modeling and Case Representation) is an instantiation of ECoCADe. Its involves the Case-Based Reasoning (CBR) cycle with evidence-based practice procedures, considering the diversity of contexts (problem actor, evidence generation and decision making), supported by the decision-making model of Simon et al. (1987).

Kruchten et al. (2006) propose an ontology to organize different types of decisions, as well as an extensive list of attributes focused on the documentation and evolution of each decision. An information retrieval technique gathers the traces of the information collected using data mining.

3 FOUNDATIONS

Two basic techniques are involved in this proposal: DR and CBR. Toulmin (1958) can be considered the precursor of semiformal graphical representation schemes for visualization of arguments based on DR. An argument consists of a fact or observation, a logical step, and an assertion. The rationale is based on a reference or an explanation. The most relevant features of DR are: (i) ease of explicit documentation (Tyree, Akeman, 2005); (ii) retention of context so that the suitability of solutions can be verified (Rittel, 1973); (iii) ease solutions obtained from the discussion on alternatives in groups; (iv) avoidance of conflicts between restructuring of a solution with the rest of the project (Burge and Brown, 2000); (v) capture of knowledge related to intellectual capital, collaboration and knowledge sharing and (vi) visibility of all ideas that have helped to guide the project, facilitating continuity in the same chain of reasoning or the integration of new participants.

Rittel (1973) found in DR a response to the complexity involved in a project. For him, the real problems in design situations were not well described and associated with a set of possible solutions, precluding a clear view of the problem from the outset.

Souza et al. (1998) emphasize that DR helps: (i) solving similar problems; (ii) understanding the design of a product; (iii) maintenance of a product, avoiding the forgetfulness of the reason for the adoption of certain solutions; (iv) impact assessment of changes; (v) communication between teams; (vi) monitoring and finding errors; and (vii) reduction of arbitrariness in the decision-making process, since it is based on the justification of a choice.

The literature considers DR as an interesting alternative to represent problematic situations and associated solutions, along with the respective justifications. Despite the relative consensus on this advantage, there have been few advances in the adoption of tools based on DR. It follows three perspectives: argumentation, communication, and documentation (Shipman, McCall, 1997). Argumentation and documentation focus on project decisions and the reasons behind them. The first one structures how the decision maker addressed the problem and the second one provides knowledge about the project to external people. The communication perspective is an attempt to preserve the information interchange among the team (Burge, Brown, 2000).

Although Shipman and McCall (1997) have considered communication as the strongest motivation for capturing a DR, they points the difficulty involved for indexing it. Francisco (2004) argues on the usefulness of argumentation recovery, but emphasizes the existence of problems related to the capture in this perspective.

The DR power is that it ensures the preservation of information related to important decision-making in a project. However, to enable this ability it is necessary to develop a representation that addresses: What is appropriate to represent? How could this representation be used? What alternatives are designed to solve an issue? Why a solution was adopted? What feedbacks were offered over the time that a particular solution was used?

In addition, it is important to ensure that the essential issues will be clear to others that will need to deal with them later. It is essential to promote the visibility of all ideas that have helped to guide the project up to the present, facilitating the continuity in the same chain of reasoning or even the integration of new participants.

However, Horner and Attwood (2006) pointed the following restrictions to DR: (i) limitation of human information processing, (ii) difficulties in

eliciting tacit knowledge; (iii) lack of incentives to capture; (iv) difficulty in perceiving probable benefits; (v) costs; (vi) risk of exposing an employee if his decision was not satisfactory; (vi) identifying what rationale should be stored and what methods are used for recovery; and (vii) applicability of the reasoning.

The basic idea of CBR is to keep the information related to the alternatives considered for solving a problem and their justifications. Amodt (1994) proposed the following steps for the knowledge representation:

- **Case representation:** structure of cases for representation, indexing and recovering;
- **Cases recovery:** it requires a clear definition of the criteria that best represent a problem and what kind of similarity will be considered. The data retrieval process should be able to discard the insignificant cases showing only those that offer an adequate proximity to the solution;
- **Cases reuse:** there are two possibilities in this step: a complete copy of the recovered solution or its adaptation to the new problem;
- **Cases review:** both the success and failure of an adopted solution must be stored in the case base; in cases of failure, it is necessary to identify the reasons for this result, contributing in the future for avoiding the occurrence of similar faults;
- **Cases retention:** learning process of the CBR, the appropriation of results from both reused cases and those that were not useful.

CBR was developed under the necessity to mitigate the dependency of specialist for problem resolution. To make CBR feasible, the problem knowledge must be represented by a contextualized case, registering an episode in which a problematic situation has been solved. A case is characterized by a problem situation associated with its respective solution. By means of similarities, old solutions can be adapted to solve new problems.

A case can be considered as a basic structure of knowledge encapsulation; an opportunity for learning by experience. For Watson and Marir (1994), a case is a contextualized piece of knowledge, representing an experience that contains past lessons and its context of use. A case is represented by: (i) a problem (case scenario); (ii) a solution (set of steps to solve the problem) and (iii) a result (domain behavior after solution application).

Shiu and Pal (2004) consider that CBR should be used in scenarios that include the following characteristics: (i) when it is impossible to fully understand the domain; (ii) there are exceptions in new situations; (iii) the problem occurs recurrently; (iv) it is advantageous to adapt a situation in order to solve another problem; and (v) previous situations provide significant inputs to new situations.

Watson (2003) states that, in general, the representation of a case can be seen as a set of pairs (attribute-value) indexed (information that helps to reduce the search space of a case) or not indexed, used for storing information about the case context of the case context.

4 PROPOSED APPROACH

We propose DecisionMaker, an environment that combines CBR and DR for reuse in SE. CBR were adopted since a DR can be considered a case. The work involves the proposition of procedures, roles and activities that help the SE process.

4.1 Description

When facing a problem, the developer starts the decision-making process supported by DecisionMaker. As a first step, it makes the representation of the case, which allows the tool to suggest similar cases. The developer can now decide on the reuse of a case (completely or even through an adaptation). Then, the tool allows the refinement of the designed solution, by means of review cycles, providing feedback and adjustments to meet the problem requirements. Finally, the case is evaluated and retained in the memory of cases of the system, being able to support the resolution of future problems (Fig. 1).

The tools functionalities were defined on the basis of four use cases:

- **Problem register:** consultation, inclusion,

Figure 1: Decision Making Process – DecisionMaker.

modification or even exclusion of a problem in the system.
- **Solution register:** consultation, inclusion, modification, exclusion or even evaluation of alternatives to a problem.
- **Search similar situations:** suggest a group of situations similar to the problem informed.
- **Solution reuse:** complete or partial use of a solution attributed to a previous problem.

For the similarity calculation among cases, the closest neighbor method was used by comparing the corresponding vectors. The cases are also characterized by the category and complexity of the problem and the technology involved, according to Tables 1, 2 and 3, defined from expert consultation.

After recording the problem, the system presents a list of cases considered similar to the one presented. The problem-solving features are triggered in order to retrieve records that answer the following questions: (i) what problems are similar to the one registered? (ii) what is the level of similarity for each attribute?

To help identifying these criteria, similarity functions were defined (Table 4). Only those cases that are considered *Very Similar* or even *Similar* are presented to the user, ordered according their degree.

Given the suggestions presented, the system user evaluates if some of them solve the presented problem. If so, the decision maker selects the one chosen for reuse in solving the problem. It is worth to remember that a reused solution does not need to solve a problem totally, as this can only serve as the basis for the definition of a new solution. Thus, cases presented as similar can assist the decision maker in solving the problem by merely serving as inspiration.

After choosing the solution, the decision maker performs the implementation. However, just as it does during software testing, the solution to the problem can suffer many adjustments until it reaches its correct format. In this activity, known as problem review, the decision maker changes problem information and alternatives as needed.

Next, the user evaluates the solution. For this, the decision maker assigns a score from 0 to 10 considering the degree of satisfaction experienced with the resolution of the problem. This evaluation will also serve to point out that the case has been completed and can be made available as a suggestion for new problems.

Table 1: Domain - Problem Category.

Category	Description
Programming	Problems related to coding.
Solution Architecture	Problems related to the architectural solution chosen to the project or even to the pattern adoption.
Realization	Problems related to the realization of systems analysis artifacts as sequence diagrams of sequence, activities or classes.
Management	Management problems including team atitude or strategy.
Business Strategy	Problems involving actions that an corporation needs to reach its desired position in market.
Activity Execution	Problems related to the effective use of a resource.

Table 2: Domain - Complexity of Problem.

Complexity	Description
Low	Problems that does not affect the project cost and timetable.
Medium	Problems that affects or the cost either the timetable of the project.
High	Problems that affects the cost and the timetable of the project.

Table 3: Project Technology.

Technologies		
BPM	JSF	PrimeFaces
DotNet	MSProject	RMC
Glassfish	MySQL	TomCat
Java	Oracle	
Jdeveloper	PHP	

Table 4: Similarity Degree.

Degree	Similarity
Very Similar	$60\% \leq x < 100\%$
Similar	$40\% \leq x < 60\%$
Low similarity	$0\% \leq x < 40\%$

4.2 Case Study

In order to evaluate the environment, a controlled experiment was carried out involving a group of professionals working in the area of software development. This experiment is divided into three stages: (i) real data collection; (ii) use of DecisionMaker, and (iii) evaluation of the experience.

Stage 1
With the support of ten experts, a set of problem situations was raised and recorded in DecisionMaker. We have collected cases experienced by those involved in order to represent the most common situations, including the use of tools, technologies, or even strategies of operation. Such representation covered the description and solution of the situation, including the criteria used for choice adoption. The alternatives considered for decision making was also recorded, even though they were not adopted. Thus, it was identified a problem and the considered alternatives, along with their justifications for choice or disposal, after the definition of the solution.

Seventy seven problem cases were surveyed, from which 17% were considered *highly complex*, 73% *medium complexity*, and 10% *low complexity*. Regarding the categories of problems, 39% corresponded to the *Programming* category, 29% to the *Solution Architecture* category, 17% *Management* and the rest also distributed in *Realization*, *Business Strategy* and *Activity Execution*. Project Technologies focused on the use of Java (32%), Oracle (23), MySQL (16) and TomCat (11%), remembering that a case may involve more than one technology.

Stage 2
In the second stage, a restricted set of 7 specialists had access to the environment to carry out queries based on the simulation of needs described according to the characteristics of both the available cases in the available collection and in totally different cases. The group included professionals with experience in software development in the ranges 1 to 3 years, 3 to 5 years, 5 to 10 years, and more than 10 years. Each participant was asked to describe two cases, one similar to the content of the collection and another totally different, registering in a form the evaluation of the process. The form included the following attributes to be evaluated with respect to recovered cases: (i) relevance of information about cases; (ii) relevance of the cases for the solution of the problem; and (iii) potential application. For each attribute of each case, the specialist assigned a score from 1 to 5 on the likert scale.

Stage 3
From the information obtained in the first two stages, the results were processed and shown in Table 5. The three issues considered were evaluated above 3, denoting a trend toward the relevance of the overall process. Globally, among the 21 evaluations, 66% are above 3 what can point the proposal as promising.

Table 5: Evaluation of the Proposed Decision Support Support Process.

	Participants/Answers						
	1	**2**	**3**	**4**	**5**	**6**	**7**
Level of relevance of cases information	5	3	3	3	4	2	5
Level of relevance of suggested cases	4	3	1	2	4	2	5
Usefulness of the information to solve the problems	4	4	1	1	3	2	4

5 CONCLUSION AND FURTHER WORKS

The results obtained in the expert evaluations indicate that the DR-based software reuse process implemented by the CBR technique is capable of aiding professionals in the area of software development. The deepening of the experience, with a broader case base and evaluation in a real working environment can strengthen this perception with more emphatic results. In this case, similarity functions should be calibrated due to the natural adherence of the contents of the case base to specific development environments.

Despite the gains provided by the flexibility of the methodology used, it was found too much resistance in obtaining answers from the participants to the survey of problem situations. Corroborates to this difficulty the subjective format of the information needed to be raised (from the application of the argument perspective to DR representation) and the way in which the focal group was performed, outside the work routine of the participants. This only reinforces the need to test the tool in a real-world environment.

It can be concluded that the combined use of DR and CBR to support the decision-making process in scenarios comprised by software development can provide an improvement in the maturity of a new decision, the adaptation of a case, the speed in problems solving, influencing positively the final result of a project.

The integration of the proposed tool into a development suite can be an interesting research effort, as mentioned by experiment participants. This integration would collaborate with the

institutionalization and adherence of the necessary procedures for the collection and representation of problem situations, making the process involved to capture and recover a case, as part of the collaborator's work environment.

Finally, the association of artifacts related to the problem or even alternatives, including text documents, spreadsheets, emails or even videos and audios, that collaborate to the understanding of the explicit case could be included as a functionality to the proposed platform.

REFERENCES

Aamodt, A., Plaza, E., 1994. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*. 7(1): 39–59.

Burge, J.E., 2008. Design rationale: Researching under uncertainty. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 22(4):311-324.

Burge, J., Brown, D.C., 2000. Reasoning with design rationale. In *Proceedings of the Artificial Intelligence Design Conference*.

Burge, J., Brown, D.C., 2004. An integrated approach for software sedign checking using rationale. In J. Gero (Ed.), *Design computing and cognition '04*, Kluwer Academic, p. 557-576..

Conklin, J., Begeman, M.L., 1988. gIBIS: a hypertext tool for exploratory policy discussion. *ACM Transactions on Information Systems*, 6(4):303-331.

Fischer, A., Greiff, S., Funke J., 2012. The Process of Solving Complex Problems. The Journal of Problem Solving, 4(1):19-42

Francisco, S. D., 2004. *DocRationale - uma ferramenta para suporte a Design Rationale de artefatos de Software*. Master Dissertation. Instituto de Ciências Matemáticas e de Computação de São Carlos, Universidade de São Paulo, Brazil.

Gopalakrishnan, A., 2015. Improving decision making and reuse in software systems using domain specific reference architectures. In *IEEE International Conference on Electronics, Computing and Communication Technologies* (CONECCT).

Horner, J., Atwood, M.E., 2006. Effective Design Rationale: Understanding the Barriers, In Dutoit, A.H. et al., *Rationale Management in Software Engineering*, Springer, p. 73-90.

Kolodner, J.L., 1993 *Case-Based Reasoning*. Morgan Kaufmann.

Kruchten, P., Lago, P., van Vliet, H., 2006. Building up and Reasoning about Architectural Knowledge. In Hofmeinster et al (Eds.) *2nd International. Conference on the Quality of Software Architectures*, LNCS 4214, Springer Verlag.

Lee, J., 1990. SYBIL: a tool for managing a group decision rational. In *Proceedings of the 1990 ACM Conference on Computer-Suppoerted Cooperative Work*, p. 79-92

Moaven, S., Habibi, J., Ahmadi, H., Kamandi, A., 2008. A Decision Support System for Software Architecture-Style Selection. In *Sixth International Conference on Software Engineering Research, Management and Applications*, SERA '08.

Parreiras, F.S., Bax, M.P., 2003. A gestão de conteúdos no apoio à engenharia de software. In: *KMBrasil*, São Paulo, Brazil.

Rittel, H.W.J., Webber, M.M., 1973. Dilemmas in a general theory of planning. *Policy Sciences*, 4(2):155-169.

Rus, I., Lindvall, M., Sinha, S.S., 2001 *Knowledge Management in Software Engineering*. Nova York: The Data and Analysis Center for Software.

Santos Jr, G.P. dos, 2009. *Integração de um Sistema de Raciocínio Baseado em casos e um Agente Inteligente de Diálogo para Resolução de Problemas de Programação*. Master Dissertation. Universidade Federal de Campina Grande, Brazil.

Shipman, F., McCall, R., 1997. Integrating Different Perspectives on Design Rationale: Supporting the Emergence of Design Rationale from Design Communication. *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing*, 11(2):141-154.

Shiu, S., Pal, S.K., 2004. *Foundations of Soft Case-Based Reasoning*. Wiley-Interscience.

Shum, S., 1991. Cognitive Dimensions of Design Rationale. In D Diaper and N V Hammond (Ed.) *People and Computers VI*: Proceedings of HCI'91, Cambridge University Press: Cambridge., p. 331-344.

Simon, H. A., et al., 1987. Decision making and problem solving. *Interfaces*, 17(5):11-31.

Souza, C.R.B, et al., 1998. A Model Tool for Semi-Automatic Recording of Design Rationale in software Diagrams. In *Proceedings of the Strict Processing and Information Retrieval Symposium*, p. 306-313.

Toulmin, S. The Uses of Argument. Cambridge: Cambridge University Press, 1958.

Tyree, J., Akerman, A., 2005. Architecture Decisions: Demystifying Architecture. IEEE Software, 22(2):19-27.

Watson, I., Marir, F., 1994. Case-based reasoning: a review. Knowledge Engineering Review, 9(4):327-354.

Watson, I., 2003. *Applying Knowledge Management: Techniques for Building Corporate Memories*, Morgan Kaufmann.