ORGANIZATIONAL KNOWLEDGE MANAGEMENT THROUGH SOFTWARE PROCESS REUSE AND CASE-BASED REASONING

Viviane A. Santos and Mariela I. Cortés
State University of Ceará, Fortaleza, Brazil

Keywords: Case-based Reasoning, Process Reuse and Improvement, Process Asset Repository, Management Tool.

Abstract. Software process reuse involves different aspects of the knowledge obtained from generic process models and previous successful projects. The benefit of reuse is reached by the definition of an effective and systematic process to specify, produce, classify, retrieve and adapt software artifacts for utilization in another context. In this work we present a formal approach for software process reuse to assist the definition, adaptation and improvement of the organization's standard process. A tool based on the Case-Based Reasoning technology is used to manage the collective knowledge of the organization.

1 INTRODUCTION

The organization knowledge is defined, in general, by the tacit and explicit knowledge from several subunits or groups combined and used to create new know-ledge (Schulz, 2002). In the software development area it is considered a best practice and is highly recommended since successful and even unsuccessful experiences can help the organization learn from the past (ISO, 2006) (PMI, 2004).

The purpose of the process reuse technology is to support the process definition and continuous improvement on the basis of standard processes, accor-ding to norms and quality models, and learned experiences (Perry, 1996).

Dynamic and context-depending aspects of the knowledge in software development turn the Case-Based Reasoning approach (CBR) (Kolodner, 1993) useful as it provides a broad support for the dynamic management of the organizational knowledge and continuous incremental learning.

The systematic reuse and the incremental capture of feedback may lead to process consolidation. In this work we describe an approach for building a reusable processes assets repository in accordance with the organizational reality to facilitate the organizational learning and the continuous processes improvement.

This work is organized as follows: in Section 2 the CBR technology is briefly explained. In Section 3 the process reuse using CBR is presented. In Section 4 the tool is briefly described. Finally, considerations are presented.

2 CASE-BASED REASONING

The CBR technology solves problems in a specific situation through previous similar situations (Pal and Shiu, 2004). A case comprises a pair problem that describes the context of an actual case occurrence, and solution that presents the problem solution. Past cases are used to hint strategies to solve new similar problems (Mille, 2006).

A CBR system is composed by 4 basic elements (Kolodner, 1993): knowledge representation, similarity measurement, adaptation and learning. In this work, the CBR technology is used to manage the assets repository, the organizational learning and the retrieval and retention of assets.

3 PROCESS REUSE APPROACH

In the proposed approach (Santos et al., 2009), the main component is the *Processes Assets Repository* (Figure 1), which is designed to store reusable process models and their feature-value representations. The feature-value representation involves a set of relevant case properties and their values. The Search Engine uses CBR technology to retrieve similar

Santos V. and Cortés M. (2010).
ORGANIZATIONAL KNOWLEDGE MANAGEMENT THROUGH SOFTWARE PROCESS REUSE AND CASE-BASED REASONING.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Databases and Information Systems Integration*, pages 223-228
DOI: 10.5220/0002899802230228

cases through the similarity measurement on the basis of process and project features.

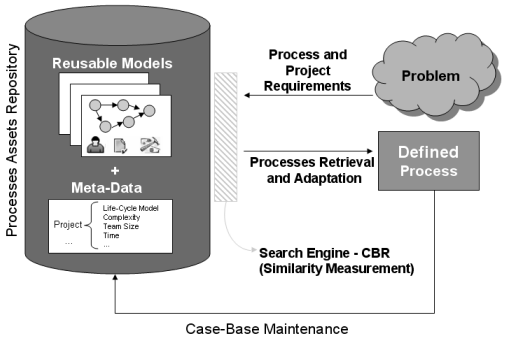

Figure 1: Approach for process reuse.

The reutilization involves the adaptation of a previous solution for a similar case, using an appropriate method (Mille, 2006). After its definition and execution in the new project, the new case is evaluated in order to examine its effectiveness and capture its new representation. Then, the new case can be stored into the repository, improving its capacity.

3.1 Representation of Organizational Assets in the Repository

The similarity concept consists of establishing an estimation of the utility of a previous case stored in the repository against the current case (Kolodner, 1993). Table 1 presents the assets representation in the repository grouped by process and project scopes.

Table 1: Representation of the software process assets.

Scope	j	Feature	Similarity Type
Project	1	Life-Cycle Model	QVI
	2	Complexity	QFI
	3	Size	QFI
	4	Team Size	NUM
	5	Time	NUM
	6	Software Engineering Knowledge	QFI
	7	Development Paradigm	QVI
Process	8	Development Model	QVI
	9	Maturity Model	QVI
	10	Maturity Level	QVI
	11	Complexity	QFI
	12	Process	QVI
	13	Experience on Process Usage	QFI

The similarity types are restrictions applied to the feature representations, to establish its correpondence or co-occurrence among case (Reis et al., 2001). The similarity types used in this work are:

- Numeric (NUM). Positive integer or real numbers
- Qualitative for Fixed Items (QFI). Predefined Terms
- Qualitative for Variable Items (QVI). Registered terms with possibility of new items

3.2 Retrieval Process

In CBR, several techniques can be applied for data retrieval. In (Pal and Shiu, 2004) the algorithm to calculate the similarity is based on k-NN technique, where the global similarity (SIM) between two cases (*a* and *b*) is defined by the weighted sum of the local similarities (sim_j) for each feature (A_j).

$$SIM(a,b) = \sum_{j=1}^{n} w_j \times sim_j(A_j(a), A_j(b)) \quad (1)$$

The weight (w_j) reflects the relevance of a feature (A_j) concerning the similarity of cases. This factor is determined by the user and is measured by the values: High (100), Medium (50) and Low (10). The features considered more important for the problem resolution from the user's viewpoint, possess higher weights.

The local similarity is calculated in accordance with the similarity type of each feature. For features of NUM and QFI similarity types, it considers the computation of distance (d_j) between each feature values in the cases *a* and *b*, as presented in the formula (2).

$$sim_j = \frac{1}{1 + d_j(a,b)} \quad (2)$$

This measurement must be normalized (Ricci et al., 2002) to avoid over influence of a metric by the great range of values of the features. The normalization process uses smallest and greatest values in the repository to linearly produce values between 0 and 1. The distance between two features of NUM or QFI similarity types are calculated on the basis of a proportionality relation between the values, as expressed below:

$$d_j(a,b) = \left| \left(\frac{A_j(a) - \min(A_j)}{\max(A_j) - \min(A_j)} \right) - \left(\frac{A_j(b) - \min(A_j)}{\max(A_j) - \min(A_j)} \right) \right| \quad (3)$$

Finally, to calculate the distance between features of QVI similarity type, a taxonomy is used to hierarchically represent the relationships among the terms. In a taxonomy, as deeper the nodes are located in the hierarchy, greater is the similarity value. In the same way, whenever the nodes are closer to the root of the

taxonomy the similarity goes to zero.

Considering n_a and n_b different nodes in a taxonomy, the similarity between those nodes, $sim_j(n_a, n_b)$, proposed by (Wu and Palmer, 1994) consists of:

$$sim_j(n_a, n_b) = \frac{2 \times N(n_p, n_r)}{N(n_a, n_p) + N(n_b, n_p) + 2 \times N(n_p, n_r)} \quad (4)$$

where $N(n_a, n_p)$ and $N(n_b, n_p)$ are the number of edges in the path from the corresponding nodes and their common parent in the hierarchy, and $N(n_p, n_r)$ is the number of edges among this common parent node and the root of the taxonomy. If the nodes n_a and n_b are common, the formula (4) isn't applicable. In this case, the similarity between those nodes is considered equal to 1.

3.3 Adaptation Process

Adaptation involves the process to transform the retrieved results into an appropriated solution for the current problem. The adaptation process can be realized following different approaches (Pal and Shiu, 2004). In this sense two approaches can be suggested: if the similarity measurement of the retrieved process in the top of the ranking is satisfactory, a minimal or null adaptation can be required. In other case, when none of the retrieved processes fulfills the requirements in appropriate manner, a compositional approach can be used.

3.4 Learning Process

The learning process is done through the feedback about the performance of the new case, when the project is closed (Pal and Shiu, 2004). The case performance evaluation consists of 3 steps detailed as follows.

3.4.1 Global Similarity Comparison

When the project is closed, the representation for the executed process can be different from the representation used in the recovery phase.

The comparison between both representations, called *Global Similarity Comparison* (*GSC*), is based on a proportionality measurement attending the occurrences of changes in the representation along the project execution. The measurement is obtained on the basis of the global similarity measurement (*SIM),* calculated according to the Section 3.2. The GSC measurement is presented in (5) and evaluates the similarities between the selected base-case representation (*a*) against the preliminary representation (*b*) and the representation of the executed process (*b'*).

$$GSC = \left(100 \times \frac{SIM(a,b')}{SIM(a,b)}\right) - 100 \quad (5)$$

When the GSC returns zero, it means that the contexts similarity values stay the same and the user's choice about the selected base-case should not be changed and the user evaluation should contribute to the learning process. In the other hand, if the *GSC* is a value greater than zero, it means that the real context is more similar to the selected base-case than the preliminary context, which possibly the user has match the appropriate case to meet the project or organization needs. Otherwise, if the *GSC* is a value less than zero, it means that the real context is less similar to the selected base-case than the preliminary context, and denotes that the user's choice was inappropriate and several modifications were required.

3.4.2 Reuse Degree

The *Reuse Degree* (*RD*) is another evaluation metric that attends the reuse percentage of the selected base-case (*a*), against the new case after the project's end (*b'*). It is obtained by the mapping of all activities components contained in cases *a* and *b'*, such as name, type, artifacts, resources, roles, connections, etc., in order to establish the reuse level. The *RD* formula is presented below:

$$RD = \frac{\sum_{q=1}^{n} N_{SimC_q(a,b')}}{m \times N_C} \quad (6)$$

where n is the number of activities from *b'* and m is the number of activities from *a*. N_{SimC} is a function that returns the number of similar components of a specific activity (q) between *a* and *b'*. To consider a component similar, it is necessary to establish a similarity threshold (90% considered). Since this metric evaluates the level of reuse, then great variation in the new case should result in low reuse. The N_C represents the number of activity components contained in a case and is useful to identify the level of reuse of the selected base-case.

3.4.3 Success Level

The success level is a subjective metric fed by the user whenever an executed process is evaluated. This metric is stored as feedback information about the base-case. This evaluation is represented by a value in the range 0 to 10.

This information is useful to the future adoption of the base-case, and contributes for the continuous improvement of the process, since cases with greater success levels will be prioritized in the search engine results.

3.5 Retention

The retention consists in the incorporation process of what is useful in a new problem resolution (Kolodner, 1993) (Pal and Shiu, 2004). Retain continually is fundamental to increment the repository with new solutions. In this research, that phase occurs after the evaluation of the executed process, in such way to extract the knowledge for later use and to integrate cases in the repository.

Depending on the user evaluation, the user may choose to transform reused process in a base-case by removing its specific project details and leaving only the suitable information to reuse in other projects and also store its context representation.

4 KNOWLEDGE MANAGEMENT TOOL OF PROCESS ASSETS

A component to support the proposed approach was implemented in the context of an existing Process-centered Software Engineering Environment (PSEE), called WebAPSEE PRO (QR Consult, 2009). WebAPSEE PRO which aims to provide automated support for software process management, including process reuse infrastructure and functionalities. WebAPSEE PRO was the chosen tool to implement the proposed approach because it has a complete metamodel and a graphic formalism that allows to design a variety of functionalities for the process reuse management.

The extension proposed by this approach aims to support the dynamic management of the organizational knowledge and continuous incremental learning. This component allows the definition of the organizational assets representation, in order to retrieve the ranking of the most similar and successful base-cases. In addition, the tool is already support the process adaptation and allows the evaluation of the reused process model performance through the metrics presented in Section 3.4. In this step, a minimum Success Level can be specified to the executed process. Finally, the retention process is provided to promote the improvement of the process interpretations in solving new problems.

4.1 Component Specification

The proposed approach was specified by UML diagrams such as use cases, class diagrams and activities diagrams (Santos, 2009).

The WebAPSEE PRO state machine involves four states. The *draft* state consists of the template initial state. The *defined* state means standard template, which allows template reuse. The *pending* state means old template version when there is a template in *draft* state. And *outdated* state means old template version when there is a template in *defined* state.

4.1.1 Use Case Approach

Table 2: Overview of the use cases approach.

Use Case	Description	Status
1- Add Scope	The user adds the scope used to group the context representation features of the software process assets. The approach predefined scopes are *Process* and *Project*, but actually one can add new scopes.	New
2- Add Feature	The user adds context representation feature of the software process assets.	New
3- Make Standard Template	The user selects the desired template to become standard. After that, the template changes its state from *draft* to *defined* (WebAPSEE PRO State Machine, Figure 2). It is an existing functionality (Costa and Sales, 2007).	Kept
4- Classify Template	The user registers the base-case context classification (Table 1).	New
5- Retrieve Templates	The system searches for the similar base-cases in the assets repository. It is an existing functionality which was updated to use the proposed context representation.	Updated
6- Evaluate Closed Process	The user evaluates the process after its closure.	New
6.1- Classify Closed Process	After its closure, the user registers the real context representation.	New
6.2- Evaluate Reuse	The system evaluates the reuse in the executed process related to the selected process model.	New
6.3- Evaluate Contexts	The system evaluates the contexts by comparing the preliminary and real global similarity values.	New
6.4- Evaluate Success Level	The user evaluates the executed process by providing a note in the range 1 to 10.	New
7- Retain Process	The user chooses to retain the executed process.	New
7.1- Generalize Closed Process	The system generalizes the executed process and transforms it in a template by removing the details and instances (Costa and Sales, 2007).	Kept
7.2- Create New Template Version	The user can choose to create a new template version. It is an existing functionality (Costa and Sales, 2007).	Kept
7.3- Create New Template	The user can choose to create a new template (Costa and Sales, 2007).	Kept
7.4- Assign Characteristics	The system saves the provided template characteristics.	New
7.5- Assign Success Level	The system saves the provided template success level.	New

In order to attend the new requirements, the WebAPSEE PRO architecture incorporates new functionalities. The main actor of the tool is the *Manager*

Console User, which, according to the WebAPSEE PRO architecture, executes the available services. The new functionalities of the proposed approach and the related pre-existing WebAPSEE PRO functionalities are illustrates in Table 2, indicating which one is new, updated or kept.

4.1.2 Class Diagram

According to the WebAPSEE PRO architecture, an abstract process is represented by the class *Template*, and concrete process by the class *Process*. The *Project* and *ProcessModel* classes represent the project data and the process model, respectively. Figure 2 presents the class diagram to support the proposed approach.

In the *Template* class was included a new attribute, called *sucessLevel*, which aims to store the user assigned evaluation grade for the template. This evaluation is later used in the calculation of the template success level average in the instantiated processes.

The new classes were created (in gray) and grouped in the package *Reuse*. The *SimilarityType* class represents the similarity types of the Section 3.1. The *Feature* class represents the features of the Table 1.

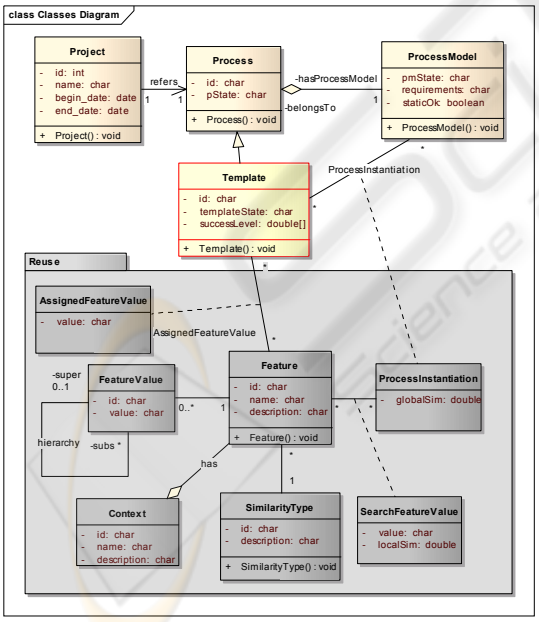

Figure 2: The proposed approach class diagram.

The *FeatureValue* class represents the possible values for the features. The *AssignedFeatureValue* associative class represents the registered values for the features to a specific template. The *ProcessInstantiation* associative class represents the global similarity value from the selected base-case against the new case. The *SearchFeatureValue* associative class represents each feature value of the new case and its local similarity against the base-case.

4.2 The Component Prototype

On the basis of the previous sections, the prototype of the component was implemented. In this section, the most relevant screenshots are illustrated to allow the visualization of the previous specification. The tool was previously fed with the contexts, the similarity types, the features and its values (based on the Section 3.1).

Each template in the repository should become standard, state change from *draft* to *defined*, to proceed the characterization by assigning its feature values to be used in the search engine (Figure 3).

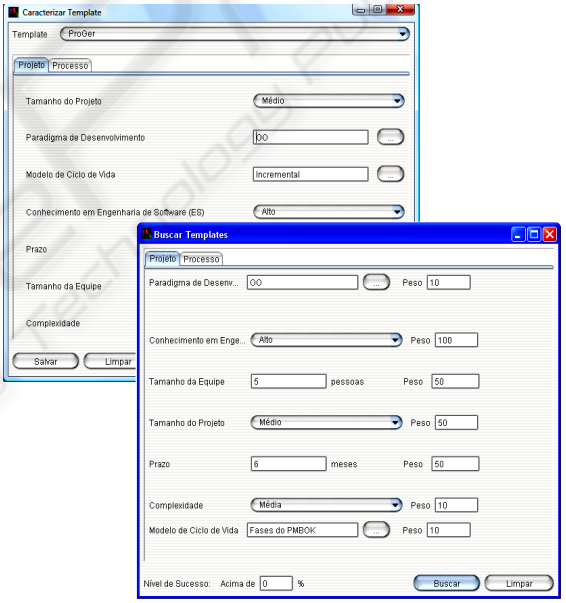

Figure 3: Template characterization and overview.

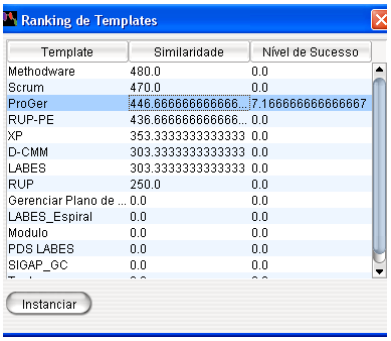

Figure 4: Ranking of the templates matching.

The search engine, using the retrieval process, returns a ranking of similar base-cases matching the new case (Figure 4). With the ranking of similar templates, the user can select and instantiate a base-case, adapting the new case to the organizational/project needs.

At the end of the process execution, the user may evaluate the performance of the new case and choose if it is worth to retain displayed in Figure 5.

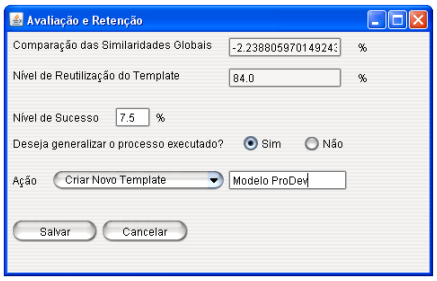

Figure 5: Process performance evaluation and retention.

When the user chooses to retain the new case, the repository is incremented with new context representations and new success level, improving the future search engine results to the organization (Figure 6).

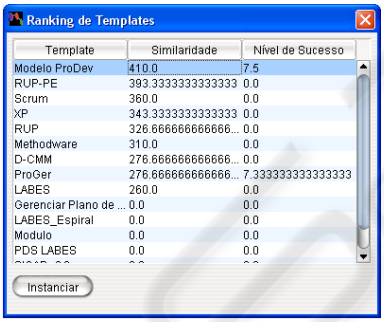

Figure 6: Search engine results after retention.

Finally, depending on the requirements provided to the search engine and the success level filter, the most similar and successful base-cases will be prioritized, promoting the reuse of successful experiences and the process continuous improvement in the organization.

5 FINAL CONSIDERATIONS

The proposed approach is based on CBR and promotes the reutilization of process assets as a start point for the elaboration of a standard process to meet the organizational needs. The cases are classified according to a set of relevant features to allow an efficient normalized retrieval. To ensure the learning process, it provides a case evaluation at the project's end. After that, the organization may decide the purpose of the new case.

This management tool allows the construction of the dynamic organizational knowledge and foresees the continuous improvement of the process through the permanent feedback to the repository involving the incorporation of its successes and failures. The learning capability of CBR systems contribute to the adoption of better and more efficient solutions.

REFERENCES

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

Mille A., 2006. From case-based reasoning to traces-based reasoning. *Annual Reviews in Control 30(2)*. Elsevier.

Pal S. and Shiu S., 2004. Foundation of soft case based reasoning. *Wiley series in intelligent systems*, 5th ed.

Perry D., 1996. Practical Issues in Process Reuse. In ISPW, International Software Process Workshop. *IEEE Computer Society Press*. France.: Int. J. Digit. Libr. 1 (1997).

PMI Project Management Institute, 2004. A Guide to the *Project Management Body of Knowledge*: PMBOK Guide. PMI, 3rd. edition.

QR Consult, 2009. WebAPSEE Pro. Available in: *http://www.qrconsult.com.br/index.php?option=com_content&view=article&id=48&Itemid=63*

Reis R., Reis C., Nunes, D. J., 2001. Automated Support for Software Process Reuse: Requirements and Early Experiences with the APSEE model. In *7th International Workshop on Groupware*. *IEEE Computer Society Press*. Darmstadt, Germany.

Ricci F., Arslan B., Mirzadeh N., Venturini A., 2002. Detailed Descriptions of CBR Methodologies. *Information Society Technologies.* available in: http://dietorecs.itc.it/PubDeliverables/D4.1-V1.pdf.

Santos V., 2009. Aprendizado Organizacional e Melhoria Contínua de Processos de Software através de Reuso de Processos de Software. *Master Dissertation in Computer Science*. State University of Ceará, Brazil.

Santos V., Cortés M., Brasil M, 2009. Reuse and Adaptation of Software Process Using Similarity Measurement. In: *Proceedings of the International Conference on Evaluation of Novel Approach to Software Engineering*. Italy.

Schulz, M., 2002. The Uncertain Relevance of Newness: Organizational Learning and Knowledge Flows. *Academy of Management Journal*. University of Washington.

ISO. The International Organization for Standardization and the International Electrotechnical Commission, 2006. ISO/IEC 15504 *Information Technology Process Assessment Part 5*.

Wu Z., Palmer M., 1994. Verb Semantics and Lexical Selection. In *32nd Annual Meeting of the Association for Computational Linguistic*, New Mexico, USA.