REMEX - A Case-Based Approach for Reusing Software Measurement Experienceware

Christiane Gresse von Wangenheim

Federal University of Santa Catarina - Production Engineering
Florianópolis, Brazil
gresse@eps.ufsc.br

Abstract. For the improvement of software quality and productivity, organizations need to systematically build up and reuse software engineering know-how, promoting organizational learning in software development. Therefore, an integrated support platform has to be developed for capturing, storing and retrieving software engineering knowledge. Technical support is complicated through specific characteristics of the software engineering domain, such as the lack of explicit domain models in practice and the diversity of environments. Applying Case-Based Reasoning, we propose an approach for the representation of relevant software engineering experiences, the goal-oriented and similarity-based retrieval tailorable to organization-specific characteristics and the continuous acquisition of new experiences. The approach is applied and validated in the context of the Goal/Question/Metric (GQM) approach, an innovative technology for software measurement.

Keywords. reuse, experience factory, case-based reasoning, software engineering, software measurement, GQM

1 Introduction

Today almost any business involves the development or use of software. However, state-of-the-practice is that software systems often lack quality and many software projects are behind schedule and out of budget [17]. In order to successfully plan, control and improve software projects, organizations need to continuously evolve software engineering (SE) know-how tailored to their specific characteristics and needs [8,10]. Experiences from their software projects have to systematically captured and reused across the organization. This enables the consolidation of organization wide SE know-how into competencies that empower the company to achieve considerable improvements and benefits [32]. Currently, reuse of SE knowledge is done in an ad-hoc, informal manner, usually limited to personal experiences. For the systematic acquisition and organization-wide communication of these experiences, corporate memories [2,8,18,20] have to be built (see Figure 1). In the software domain, the *Experience Factory* (EF) approach [8] proposes an organizational infrastructure for the analysis and synthesis of all kinds of software life cycle experiences or products. It acts as a repository for those and supplies these experiences to various software projects. However, for the operationalization of an EF in practice, we need a clever assistant that supplies the right experiences from the *Experience Base* (EB) to the user on demand.

In order to comprehensively support the software development process, various types of *experienceware* (EW) [18], including expertise and lessons learned (e.g., how to apply design inspections), quality models (e.g., distribution of rework effort per fault type), and deliverables (e.g., software measurement plans, requirement documents) related to several processes (e.g., design inspection, measurement) in different environments have to be retrieved addressing various purposes: facilitation of the planning or execution of software projects, prevention of past failures by anticipating problems, and guidance for the solution of occurring problems. And, since each software project is different, it is very unlikely to find an artifact fulfilling the needs of the actual project completely. Thus, experiences have to be retrieved from projects with "similar" characteristics, assuming that similar situations (or problems) require similar solutions. In the SE domain, for example, we assume, that measurement programs with similar goals use similar quality models or that similar problems occurring during design inspections have corresponding solutions. Due to the lack of general SE models in practice, organizational software know-how has to evolve in an incremental manner by learning from each new software project. Thus, the EF has to support continuous learning by capturing and integrating new experiences when software projects are planned and executed. In this context, *Case-Based Reasoning* (CBR) [5] plays a key role [10,18,20,27], as it provides a broad support for similarity-based retrieval for all kinds of EW and continuous incremental learning. However, the operationalization of the EF is not trivial, as relevant SE knowledge has to be identified, modeled and represented in the EB. Methods for goal-oriented retrieval providing support for different processes, objectives, and environments in the SE domain and for the continuous acquisition and integration of new experiences have to be developed. In this paper, we propose a case-based approach for an integrated support platform enabling organizational learning from SE experiences tailorable to organization specific characteristics. The approach is based on our experiences on reusing GQM-based measurement know-how (e.g., in the context of the industrial transfer and research projects [11,26]).

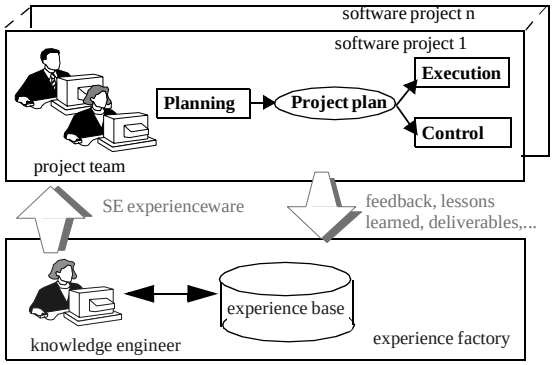

Fig. 1. Experience factory organization

2 Reuse of GQM Measurement Plans

In this section, we give a short overview on software measurement, the application domain of our approach, and provide scenarios illustrating the reuse of measurement EW. Software measurement is an essential infrastructure technology for the planning, control and improvement of software projects. Organizations have to collect quantitative and qualitative data concerning their software products and processes, to build an em-

pirical justified body of knowledge. A specific technology for goal-oriented measurement is the *Goal/Question/Metric* approach (GQM) [9], which supports the definition and implementation of operationalizable software improvement goals. Based on a precisely specified measurement goal, relevant measures are derived in a top-down fashion via a set of questions and models. This refinement is documented in a GQM plan, providing a rationale for the selection of the underlying measures. Data is collected wrt. the measures and interpreted in a bottom-up fashion in the context of the models, question and goals, considering the limitations and assumptions underlying each measure. The establishment of measurement programs, which in practice requires a significant planning effort, can be substantially facilitated by reusing measurement EW [16], as illustrated in the following scenario.

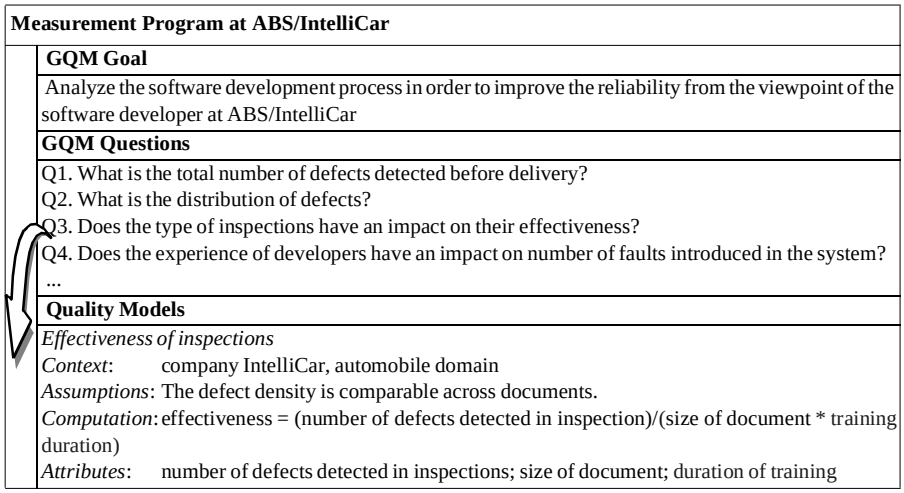

Fig. 2. Excerpt of simplified example of GQM plan

Suppose a company, IntelliCar, which produces embedded software for automobiles has two main departments: FI which develops software for fuel injection devices and ABS which develops software for ABS brake control devices. As the company produces embedded software, one of its most important goals is to produce zero-defect software. Therefore, department FI established successfully a quality improvement program based on measurement two years ago. Now, also department ABS wants to start measurement-based improvement. As the contexts of both departments are similar and the improvement goal is the same, experiences available in department FI can be reused at ABS in order to reduce the planning effort and to improve the quality of the measurement program. Based on the measurement goal «*Analyze the software development process in order to improve the reliability from the viewpoint of the software developer at ABS/IntelliCar*», relevant quality aspects and influence factors have been acquired during interviews with the developers of department ABS. These are represented as a set of questions in the GQM plan, as shown in Figure 2. Now, in order to operationalize the questions of the GQM plan, quality models have to be developed. Assume, for example, that the question «*Q3. Does the type of inspection have an impact on the effec-*

tiveness of inspections?», has also been evaluated in a similar measurement program in department FI. Then, the respective model can be reused, assessing its applicability based on its underlying assumptions. If necessary, the model is adapted to the specific characteristics of ABS. For example, assuming that inspector capabilities vary extensively between departments, the effectiveness of inspections is expected to depend not only on the size of the inspected document (as stated in the reused model), but also on the training of inspectors, then the new factor is included in the model.

While defining a model for question Q2, it turned out that an operational refinement of the question is impossible due to missing information concerning defect classification. The solution of this problem can be guided by experiences describing how a similar problem has been successfully solved at department FI (see Figure 3) by suggesting follow-up interviews in order to acquire the required information completely. In addition, reusing organizational glossaries can support the consistent usage of terms (e.g. *defect*) and reusing taxonomies representing generalization relations can help the refinement of abstract concepts (e.g. *«distribution of defects»* in Q2). Other phases of the measurement planning process can be supported accordingly through the reuse of measurement EW [16].

Context	company IntelliCar; department FI
Problem	Question of the GQM plan cannot be refined into an operational quality model due to missing information.
Cause of Problem	During the interviews the necessary knowledge has not been acquired completely from the project personnel.
Solution	A follow-up interview was performed with the person(s) who mentioned the respective quality aspects during the first interviews in order to clarify the formulation of the GQM question.
Outcome	The required knowledge was acquired completely and the respective quality model was defined.

Fig. 3. Example of problem experience

3 Representation of GQM Experienceware

In order to facilitate and improve the planning of GQM-based measurement programs through reuse of EW, an Experience Base is developed, modeling and representing relevant measurement EW.

3.1 GQM Experienceware Cases

As today wrt. most SE technologies no formal knowledge exists, the principal source are individual project experiences. Thus, SE EW is primarily captured in form of cases in the *GQM-Experience Base* (GQM-EB)[1], representing context-specific experiences gained in a particular software project in a specific organization. In order to provide comprehensive support, different types of EW cases are modeled by using a flexible, object-oriented frame-like representation formalism based on [24,28] and are stored in the GQM-EB [15,19]:

- *GQM Product Experienceware Case (GQM-PEC).* These cases include GQM products developed during the planning of a GQM-based measurement program. GQM-PECs are reused in similar software projects as a basis for the development of respec-

1.Here, we consider a specific instantiation of the experience base focusing on EW on the planning of GQM-based measurement programs.

tive products, resulting in a reduction of planning effort and improved quality of the GQM products.

- *GQM Problem-Solution Experienceware Case (GQM-PSEC).* GQM-PSECs explicitly capture problem solution strategies that have been adopted in past measurement programs (see Figure 3). Reusing GQM-PSECs can warn for potential failures in advance and guide a solution fitting the application context. Due to the specific nature of experiential knowledge, GQM-PSECs are represented as cases describing a specific problem, its cause, the solution applied and the outcome achieved.

Experienceware Cases are represented by a set of relevant attributes and interdependencies based on domain models (see Section 3.2) [29]. To enable the retrieval of EW cases from similar software projects, the environment from which the case has been obtained is characterized. This is done through a minimal set of characteristics (e.g., business sector, improvement goals, development process used), which allows to identify similar cases and to discriminate different ones. In order to assess the reuse potential of the case, cases are enhanced by basic information (e.g., viewpoint, representativeness). Information about past reuses of a case, such as preconditions for reuse, required adaptations, cost and frequency of reuse, are explicitly captured [19] in order to facilitate the reuse of experiences and their adaptation to specific project characteristics.

3.2 General Domain Knowledge

In order to model relevant EW and facilitate the consistent representation and acquisition of new experiences across software projects, general domain knowledge on GQM EW is represented in the GQM-EB [16,19].

GQM EW Models. Entities related to GQM EW are explicitly modeled in a hierarchy of classes [16,28] (see Figure 4). Each class is structured by a set of attributes represent-

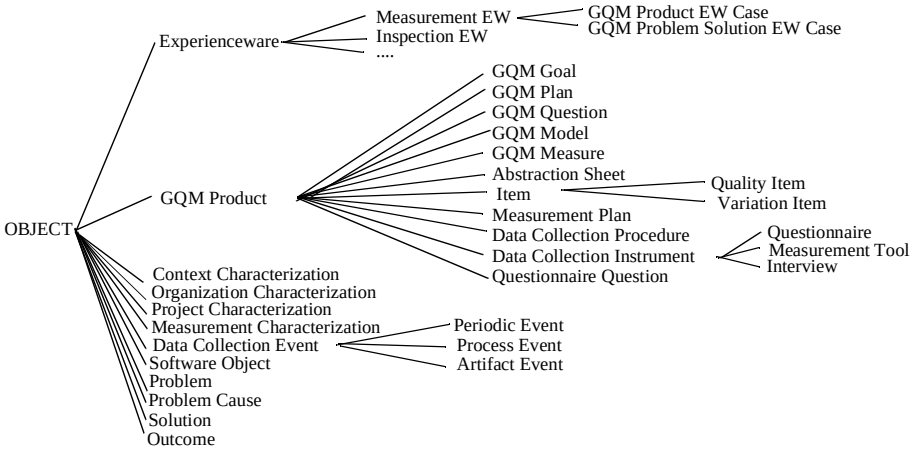

Fig. 4. GQM EW Classes (is_a relation)

ing basic values or relationships to other entities. Attributes are defined through an identifier, description, cardinality, its type or kind of relationship, a default value and explicitly stating if the attribute has to be specified (mandatory) when a new instance of this class is acquired [15].

Class	GQM Measure					
Description	defines data to be collected					
Attributes	**Identifier**	**Description**	**Cardinality**	**Type or Kind**	**Default**	**Mandatory**
	id	identifies the GQM measure	1	Identifier	-	yes
	assumptions	about the applicability of the measure	0..1	Text	none	no
	description	describes data to be collected	0..1	Text	-	yes
	scale	defines scale of the measure	0..1	Scale	-	yes
	unit	declares unit of the measure	0..1	Unit	-	no
	range	declares range of the values of the measures	0..1	Text	-	no
	model	references the corresponding model	0..*	defined-by [GQM-Model])	-	yes
	...					

A GQM measure defines which data has to be collected. It includes the explicit definition of assumptions concerning the application of the measure. Regarding the expected values, scale, unit (only in case of numerical values) and range have to be defined.As GQM measures are derived from models which determine the attributes to be measured, this is represented as a defined-by relation. Based on the GQM measure the collection procedure defining when, how, and by whom the data has to be collected is defined.

Fig. 5. Simplified example of the class GQM Measure

Type Definitions. Type definitions model qualities of SE entities, such as, developer experience, or categorize concepts, e.g., programming languages as Unordered Symbol with the possible values «Delphi, C++, etc.». Type definitions are used to type class attributes. They facilitate the situation assessment and support the manual adaptation of retrieved EW cases by explicitly indicating alternatives, as well, as the consistent acquisition of experiences across projects. For each type, its related supertype, range and the local similarity measure are specified. For example, the experience level of developers might be classified through the Ordered Symbols: none, low, medium, high, using the standard local similarity measure for ordered symbols. For symbol types, the meaning of each value is explicitly defined through range definitions, e.g., «high» experience may be defined as worked for more than 2 years in the application domain. In addition, for numerical types, the unit is explicitly stated, e.g., person-hours.

Glossaries. Glossaries define terminology and basic concepts related to software measurement [16,19]. For example, «Failure: is the inability of the software to perform a required function wrt. its specifications». A glossary supports the adequate use of terms, their consistency across an organization, and ensures that the reuse of GQM products is based on sound assumptions.

Taxonomies. Taxonomies represent ordered arrangements of entities according to their presumed relationships, e.g., organization hierarchy [16,19]. They guide the appropriate refinement of objects of interest during the situation assessment and acquisition of new experiences.

3.3 Knowledge Levels

Software products, processes, resources as well as characteristics and terminology vary between different organizations. Therefore, the domain model has to be tailored to the specific environment. Generally, we can identify different levels of knowledge valid in different scopes of domains:

- **Software measurement domain.** Here, general knowledge on GQM-based measurement is represented, which is transferrable between organizations. This level includes

GQM EW models, general valid types and range definitions (e.g., on measurement scale), and general valid terms in the glossary (e.g., software process).

- **Organization domain.** Here, organization specific knowledge related to software measurement is represented. If the GQM technology is modified in a particular organization, the respective knowledge from the upper level is adapted accordingly. Type and range definitions are enhanced by organization specific definitions. For example, one organization could classify «*experience of the developers*» into the categories *(expert-participated in system development; medium-participated in training; none)*, whereas another organization might classify experience into (*high-working for more than 2 years, medium-worked once, low-never worked in application domain*). The glossary and taxonomies are completed by organization specific terms.
- **Project domain.** At this level, instantiations of GQM EW cases are represented gathered from particular software projects. For example, a GQM-PEC including a GQM plan from a measurement program of the project HYPER at the department ABS/IntelliCar.

4 Experience-Based Support of GQM Planning

4.1 Determination of Retrieval Goals

During the planning of GQM measurement programs the GQM-EB can be inquired to find useful EW to guide, support and improve various SE tasks in a specific environment. In order to provide comprehensive support for several SE tasks, various types of experiences have to be retrieved, from different viewpoints in different environments addressing various purposes: support of software projects by reusing similar products developed in the past, prevention of failures by anticipating problems, guidance for the solution of problems by reusing solution strategies adopted in past similar problems, and the identification of patterns of experiences for the maintenance and evolution of the EB. Thus, a goal-oriented retrieval method [14] is developed that retrieves a set of relevant experiences wrt. a specific reuse goal. Based on reuse scenarios, retrieval goals are determined explicitly specifying the following dimensions:

Retrieve <object>
to <purpose>
concerning <process>
from the <viewpoint>
in the context of <environment>

For example, «*retrieve lessons learned to guide problem solution concerning software measurement from the viewpoint of quality assurance personnel at IntelliCar*».
Based on the retrieval goals, reusability factors are determined. This includes the specification of relevant indexes[1] and their importance and the parametrization of the similarity measure. For example, for the retrieval of a solution strategy, relevant indexes

1.As index we denote attributes of the case, which predict the usefulness of the case concerning the given situation description, and which are used for retrieval and determination of the similarity value.

might be the problem description and the task when the problem occurred, whereas potential problems wrt. a specific task might be identified based on the task only.

4.2 Retrieval Process

Considering different retrieval goals, a goal-oriented method for similarity based retrieval is defined, including the following steps [14]:

Step 1. Situation assessment. The current situation is described by the user specifying the retrieval goal and a set of indexes related to the specific retrieval goal based on a predefined indexing scheme. The importance of each index wrt. the specific retrieval goal is stated through a *relevance factor* assigned to each index. Relevance factors are stored in the EB and can be manually adapted by the user. To facilitate the assignment, relevance factors are classified into «essential, important, less important, irrelevant». Indexes marked as essential are perfectly matched to the ones in the situation assessment, the ones marked as important or less important are partially matched and the ones marked as irrelevant are not further considered. Unknown indexes are explicitly marked. Table 1 illustrates a situation assessment with an exemplary set of indexes. The situation assessment is further supported by general domain knowledge [19]: glossaries can be used for a consistent usage of terminology across projects and taxonomies guide and direct the appropriate definition of indexes. Type and range definitions facilitate the identification of the present values and guarantee a consistent description across projects.

Reuse goal			**GQM Experience Base (excerpt)**		
object	lesson learned (PSEC)		**CASE PSEC_003**	**CASE PSEC_007**	**CASE PSEC_011**
purpose	guide problem solution				
process	sw measurement				
viewpoint	quality assurance personnel				
environment	IntelliCar				
Indexes					
department	irrelevant	ABS	Fuel Injection	Fuel Injection	Fuel Injection
staff size	less important	10	15	100	50
application domain	essential	automobile	automobile	automobile	automobile
improvement goal	important	improvement of sw system reliability	improvement of sw system reliability	improvement of sw system reliability	cost reduction in sw development
programming language	irrelevant	Ada	Fortran	Ada	C
dev. experience	less important	high	medium	low	low
sw system size	less important	*unknown*	15 KLOC	80 KLOC	60 KLOC
measurement maturity	important	initial	--	--	--
task	essential	measurement goal definition	measurement goal definition	development of measurement plan	measurement goal definition

Table 1. Simplified retrieval example

Step 2. Exact matching of indexes marked as essential. In a first step, the cases of the EB are perfectly matched with the situation assessment wrt. the indexes marked as essential, determining a set of potential reuse candidates. Table 1 shows a simplified example: while comparing the cases of the EB with the situation assessment, case PSEC_03 and PSEC_11 are considered as potential reuse candidates, because the values of the indexes marked as essential («application domain» and «task») are equal to the present ones. PSEC_07, which describes an experience regarding the develop-

ment of the measurement plan, is not further considered, as the value of the index «task» is different to the one of interest.

Step 3. Partial matching of similar cases. For all potential reuse candidates a similarity value is computed by partially matching the indexes (except the ones marked as *essential)* using a specific similarity measure wrt. the retrieval goal (see Section 4.3). Cases with a higher similarity value than a given threshold are considered as sufficiently similar and proposed to the user as reuse candidates ranked by their similarity values. Continuing the example shown in Table 1, case PSEC_03 is considered more similar to the given situation than PSEC_11, because the values of the indexes of PSEC_03 marked as important or less important are more similar to the current ones (especially regarding «staff size», «improvement goal»).

Step 4. Selection of reuse candidate(s). Based on the proposed reuse candidates the user can select the most appropriate case(s) and, if necessary, manually adapt them to fit the current needs. Informed decisions are further supported by experiences explicitly captured in the EB about the reuses of a particular case in the past [19] (see Section 3.1). If the system fails to propose reuse candidates, general domain knowledge, e.g., GQM product models, is available to support the SE tasks.

4.3 Similarity Measure for the Retrieval of Experienceware

For the identification of «similar» EW cases concerning various retrieval goals (see step 3/Section 4.2), we define a generic similarity measure sim(Sit´,E_k´) [14] that can be parameterized for a specific goal. Taking into account specific characteristics of the SE domain, such as the lack of explicit domain models in practice, diversity of environments, incompleteness of data, and the consideration of «similarity» of experiences, the similarity measure is based on the following assumptions (see [14] for details):

- Depending on the retrieval goal, a particular set of indexes is defined for situation assessment and matching. A set of indexes C is represented as a list of features C_g={C_{g1}, C_{g2},...} wrt. the particular retrieval goal g. The range of the value c_i of the feature C_{gi} is defined by the respective range definition W_i (see Figure 6).

Example Index Set		**Type/Range**	**Relevance Vector** R_g	**Present Situation** Sit´={(C_{gi}, s_i)	
C_1	staff size	Interval of numbers [0,50]	less important (0.15)	s_1	10
C_2	improvement goal	String	important (0.35)	s_2	"improvement of system reliability"
C_3	measurement maturity	Ordered Symbol: {initial, low, routine}	important (0.35)	s_3	initial
C_4	sw system size	Number [0,100]	less important (0.15)	s_4	*unknown*

Case E_k	**FS**
15	W
"improvement of system reliability"	E
unknown	U
15KLOC	R

Fig. 6. Example

- The present situation is assessed based on the set of indexes wrt. the retrieval goal, represented as a list of feature-value pairs Sit´={(C_{gi}, s_i) ∈ Sit | relevance factor (S_i) ≠ essential} including the features $C_{gi} \in C_g$ and their values $s_i \in W_i$.
- In the EB, an EW $case_k = (E_k,\varepsilon_k)$ represents an experience by feature-value[1] pairs (experience E_k={(E_{k1}, e_{k1}), (E_{k2}, e_{k2}),...} with the features E_{ki} and their values $e_{ki} \in W_i$

(and with $E_k' \subseteq E_k$ and $\forall\ E_{ki}' \in C_g$ and their respective values e_{ki}'), describing the know-how gathered in a software project, the context from which its originates, and its relationships (see Figure 6). In addition, a threshold $\varepsilon_k \in [0,1]$ is stated for each case that determines, if the case is sufficiently similar to the situation assessment to be proposed as a reuse candidate.

- In the SE domain, many cases may have a low similarity value, due to few identical values, although they might be quite similar (e.g. programming languages C and C++). Thus, local similarity measures are introduced. Generic local similarity measures $\upsilon'(s_i,e_{ki}') \in [0,1]$ for basic value types W(v) are defined in [19,28]. Local similarity thresholds $\theta_i \in [0,1]$ are introduced for each index C_{gi} determining if the values are considered as (sufficiently) similar.
- *Relevance factors* are defined, which reflect the importance of a feature concerning the similarity of cases wrt. a specific retrieval goal (see Figure 6). Here, for each retrieval goal *g* a specific index set C_g is used. Thus, for each index $C_{gi} \in$ index set C_g, a relevance factor $\omega_{gi} \in [0,1]$ is defined in dependence on the specific retrieval goal g. For each retrieval goal, those relevance factors are represented by a *relevance vector* $Rg = \{\omega g_1, \omega g_2, ...\}$ with $\sum \omega g_i = 1$ normalized in the EB.
- In order to explicitly deal with incomplete knowledge, the similarity of two objects is expressed through a linear contrast of weighted differences between their common and different features [6,30]. The following *Feature Sets* (FS) are distinguished:
  - E: Set of corresponding features of the given situation and the stored case ($E = \{Cg_i \mid (Cg_i \in Sit' \cap E_{ki}') \text{ and } (\upsilon'(s_i,e_{ki}') \geq \theta_i)\}$). For example, if both, the situation assessment and the stored case state the feature «experience of developer» as high.
  - W: Set of contradicting features of the given situation and the stored case ($W = \{Cg_i \mid (Cg_i \in Sit' \cap E_{ki}') \text{ and } (\upsilon'(s_i,e_{ki}') < \theta_i)\}$). For example, if in the past no effort reporting tools were available, but now in the given situation the feature «effort reporting tools» is stated as available.
  - U: Set of unknown features in the actual situation description ($U = \{Cg_i \mid (Cg_i \in E_{ki}' - Sit')\}$). For example, when initiating a software project certain information, such as «software system size» may be stated as unknown in the situation description.
  - R: Set of redundant features not contained in the stored case ($R = \{Cg_i \mid (Cg_i \in Sit' - E_{ki}')\}$). For example, the feature «developer experience» may not have been considered initially, but later become important for the identification of relevant cases.

  For each set, a specific weight $\alpha, \beta, \gamma, \delta \in [0,1]$ is defined.

The global similarity measure is defined as:

$$sim(Sit',E_k') = (\alpha \sum_{si \in E} \omega_{ik}\upsilon'(s_i,e_{ki}')) / ((\alpha \sum_{si \in E} \omega_{ik}\upsilon'(s_i,e_{ki}')) + (\beta \sum_{si \in W} \omega_{ik}(1-\upsilon'(s_i,e_{ki}'))) + (\gamma \sum_{si \in U} \omega_{ik}(1-\upsilon'(s_i,e_{ki}'))) + (\delta \sum_{si \in R} \omega_{ik}(1-\upsilon'(s_i,e_{ki}'))))$$

Based on the similarity value calculated, a $case_k$ is considered as reuse candidate, if all features marked as essential in the given situation exactly match the respective features

1.Here, values represent atomic values or relations to other entities.

of the case, and if sim(Sit´,Ek´) ≥ global similarity threshold ε_k of $case_k$.

5 Continuous Acquisition and Integration of Experienceware

The incremental evolution based on feedback from industrial applications is essential for continuously building and improving SE know-how. Consequently, the knowledge in the GQM-EB has to be enhanced and updated each time a new measurement program is run in the organization. This means that we have to continuously capture new experiences from the quality assurance personnel. In order to keep the effort related to the knowledge acquisition minimal, this process is intertwined in the retrieval/reuse process (see Figure 7): Information provided by the user as input to the retrieval process, such as a context characterization, and reused experienceware from the GQM-EB are in parallel used for the creation of new EW cases.

For example, while reusing EW in order to support the solution of a problem encountered (see Section 2) concerning the definition of a quality model, the user provides the following situation assessment: «*organization:* IntelliCar; *application domain:* automobile; *problem:* Question of GQM plan cannot be refined into model». This information is used for the retrieval process, and in parallel for the description of a new case documenting experiences regarding the present situation. Information contained in a similar case retrieved and reused in order to solve the current problem, is used to supplement the new case description (see Figure 8). The generated new case is reviewed by the user before storage in the EB. Additional information (e.g. basic information) is added, and if necessary, deviations from the reused case are adjusted, e.g., if a solution different to the one stated in the reused case was applied. The acquisition of new experiences is further guided through GQM EW models, which explicitly address relevant dimensions to be captured. Glossaries and taxonomies facilitate the consistent description of experiences across software projects.

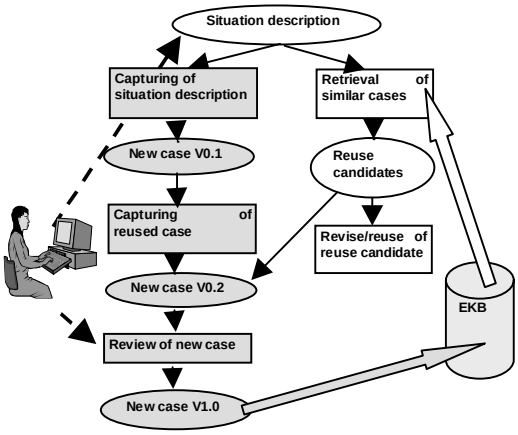

Fig. 7. Integration of acquisition process

The new acquired experiences are integrated into the existing EB to be available for future reuse. This implies that EW cases have to be stored, domain models enhanced and, if necessary, generic patterns of cases have to be created or modified.

Project-specific cases (GQM-PECs or GQM-PSECs) acquired in parallel to the retrieval process are stored as instances of GQM EW cases in the GQM-EB. Based on protocols on the retrieval/reuse process and comparisons of the reused and new case, reuse information is added to the reused case. For example, the date of reuse is added, the frequency of reuse is increased, and attributes which have been adapted to fit the new situation are explicitly listed.

In addition, project-specific cases are evaluated wrt. their similarity to other cases of the GQM-EB. If the new case differs only in small details from a case reused, an abstract case subsuming the project-specific cases is created through case generalization. The development of generic patterns through the knowledge engineer can be guided by taxonomies which provide a basis for the derivation of abstractions.

Based on an evaluation of new terms defined in the specific GQM EW case through the knowledge engineer, the organizational glossary and taxonomies are enhanced.

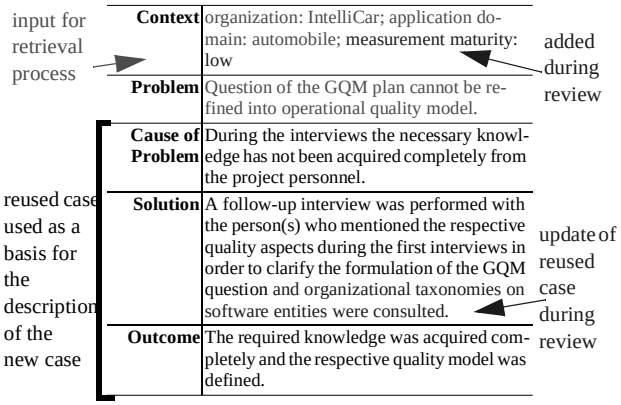

Fig. 8. Simplified example of acquisition

The continuous evolution and customizing of the EF to a specific environment may also require the modification of the representation of EW cases, the indexing scheme and similarity measure based on user feedback from the application. Due to the fact, that indexes depend on the specific environment and may change over time, the continuous tailoring of the indexing scheme needs to be supported during the whole life cycle of a GQM-EB through the knowledge engineer. For example, supplemental context characteristics of software projects may become relevant for the discrimination of cases. As shown in Figure 8, the attribute «*measurement maturity*» had not been considered as a relevant characteristic for the context description of a case in the past, because all experiences were related to projects without variations concerning the maturity. Since a new measurement program is established in a project with a different level of maturity, this attribute has become relevant for the distinction of cases and is added to the context characterization.

Continuous learning has also to take place wrt. the similarity measure and its parametrization for specific retrieval goals in order to improve and optimize its performance. Therefore, the retrieval and reuse process is supervised and, based on the feedback, appropriately tailored to the specific environment through the knowledge engineer. Here, protocols documenting the user´s (re-)actions and user-provided critics and suggestions can serve as a basis for the maintenance through the knowledge engineer (see Table 2).

Feedback	Implication for update
Index manually added for retrieval	•Addition of index to the indexing scheme
Relevance factor manually modified	•Modification of weight assigned to the index •Index frequently marked as irrelevant might be removed from the index scheme
Increasing number of retrieved reuse candidates	•Changing optimistic strategy for similarity measure into a more pessimistic •Increase of tresholds

Table 2. Examples of retrieval feedback and its implications

Frequent rejection of cases suggested as reuse candidates	•If a specific case is affected: increase of global treshold of the case •If different cases are affected: review of indexing scheme and similarity measure under consideration of additional critics and suggestions of the user

Table 2. Examples of retrieval feedback and its implications

Based on a careful analysis of the causes, the selection of indexes and/or the similarity measure have to be adapted accordingly in order to improve retrieval results in the future.

6 Discussion

In the software domain, various approaches exist for reuse primarily focusing on software code, e.g, based on library and information science, knowledge-based systems, or database management technologies [12]. However, the majority of those approaches fails to recognize the complexity of SE experience in general, often requires a thorough classification of the domain, or does not provide any means for similarity-based retrieval.

Recently, CBR has been recognized as a promising approach for the operationalization of learning organizations in the SE domain [2,3,18,22,27]. Applications are developed in different SE areas, like capturing and formalizing best practices (e.g., [20]), effort prediction (e.g., [13]), change management [23], and requirements acquisition (e.g., [25]). However, so far there does not exist an approach on reusing software measurement EW. Only few approaches offer flexible similarity-based retrieval methods, for example, through a context concept as a "similarity environment for the retrieval" [1,31], dynamic ranking of importance ratings of indexes [21], or partitioning the case base through the use of *prototypes* [7]. However, if multiple retrieval goals have to be supported by a case base, this is not sufficient. The creation of distinct case bases for test selection and diagnosis in PATDEX [6,31], can be seen in analogy to different retrieval goals, although inefficient due to administration and maintenance reasons. In contrast, our approach, systematizes the concept of goal-oriented retrieval through a flexible and tailorable retrieval method and similarity measure based on the advanced similarity model of PATDEX which explicitly deals with unknown information, filter attributes, and local similarity measures.

Besides integrating experiential knowledge (in form of cases) and general domain knowledge as in several CBR systems, our approach explicitly models different levels of knowledge focusing on different scopes.

Concerning the tailoring and continuous evolution of the EF to organization specific characteristics, only a few systems offer mechanisms for the systematic and integrated acquisition of user feedback and learning possibilities regarding the similarity measure as, e.g., the tailoring of relevance factors (see [4] for an overview), which represent the basis for the continuous evolution of our approach.

7 Conclusion

For the successful planning and improvement of software measurement, EW has to be captured in corporate memories and reused across the organization. Based on our expe-

riences on the application of the GQM approach in practice, we develop a case-based approach for the operationalization of organizational learning in software measurement focusing on the technical aspects. Relevant measurement EW is modeled, a goal-oriented method for similarity-based retrieval tailorable to specific environments is developed, and an acquisition process intertwined into the retrieval/reuse process described. Currently, we are implementing the approach. Further empirical research will have to be carried out in experiments and industrial transfer projects to assess strengths and weaknesses of the approach.

References

1. Althoff, K.-D., et al.: Case-Based Reasoning for Decision Support and Diagnostic Problem Solving: The INRECA Approach. Proc. 3rd German Workshop on Case-Based Reasoning, Germany (1995)
2. Althoff, K.-D., Bomarius, F., Tautz, C.: Using Case-Based Reasoning Technology to Build Learning Software Organizations. Proc. of Workshop on Building, Maintaining, and Using Organizational Memories at the 13th European Conference on AI (1998)
3. Althoff, K.-D., et al.: CBR for Experimental Software Engineering. In M. Lenz et al. (eds.), Case-Based Reasoning Technology - From Foundations to Applications, LNAI 1400, Springer Verlag (1998)
4. Althoff, K.-D.: Evaluating Case-Based Reasoning Systems: The Inreca Case Study. Postdoctoral Thesis, University of Kaiserslautern, Germany (1997)
5. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, 17(1) (1994)
6. Althoff, K.-D., Wess, S.: Case-based Knowledge Acquisition, Learning and Problem Solving in Diagnostic Real World Tasks. Proc. of the 5th European Knowledge Acquisition for Knowledge-Based Systems Workshop, Scotland/UK (1991)
7. Barletta, R.: A Hybrid Indexing and Retrieval Strategy for Advisory CBR Systems Built with ReMind. Proc. of the 2nd European Workshop on Case-Based Reasoning (1994)
8. Basili, V. R., Caldiera, G., Rombach, H. D.: Experience Factory. In J. J. Marciniak (ed.), Encyclopedia of Software Engineering, John Wiley & Sons (1994)
9. Basili, V. R., Caldiera, G., Rombach, H. D.: Goal Question Metric Paradigm. In J. J. Marciniak (ed.), Encyclopedia of Software Engineering, John Wiley & Sons (1994)
10. Barr, J.M., Magaldi, R.V.: Corporate Knowledge Management for the Millennium. In I. Smith, B. Faltings (eds.), Advances in Case-Based Reasoning, Springer Verlag (1996)
11. CEMP Consortium. Customized Establishment of Measurement Programs. Final Report, ESSI Project Nr.10358 (1996)
12. Frakes, W. B., Gandel, P. B.: Representing Reusable Software. Information and Software Technology, 32(10) (1990)
13. Finnie, G. R., Wittig, G. W., Desharnais, J.-M.: Estimating Software Development Effort with Case-Based Reasoning. Proc. of the 2nd Int. Conf. on Case-Based Reasoning, RI (1997)
14. Gresse von Wangenheim, C., Althoff, K.-D., Barcia, R.M.: Intelligent Retrieval of

Software Engineering Experienceware. Proc. of the 11th Int. Conf. on Software Engineering and Knowledge Engineering, Germany (1999)
15. Gresse von Wangenheim, C.: REMEX - A Case-Based Approach for Reuse of Software Measurement Experienceware. Technical Report PPGEP-C3002.99E, Graduate Program in Production Engineering, Federal University of Santa Catarina, Brazil (1999)
16. Gresse, C., Briand, L. C.: Requirements for the Knowledge-Based Support of Software Engineering Measurement Plans. Journal of Knowledge-Based Systems, 11 (1998)
17. Gibbs, W.W.: Software´s Chronic Crisis. Scientific American (1994)
18. Gresse von Wangenheim, C.: Knowledge Management in Experimental Software Engineering - Create, Renew, Build and Organize Knowledge Assets. Proc. of the 10th Int. Conf. on Software Engineering and Knowledge Engineering, San Francisco, California (1998)
19. Gresse von Wangenheim, C., von Wangenheim, A., Barcia, R. M.: Case-Based Reuse of Software Engineering Measurement Plans. Proc. of the 10th Int. Conf. on Software Engineering and Knowledge Engineering, San Francisco, California (1998)
20. Henninger, S.: Capturing and Formalizing Best Practices in a Software Development Organization. Proc. of the 9th Int. Conf. on Software Engineering and Knowledge Engineering, Spain (1997)
21. Kolodner, J. L.: Case-Based Reasoning. Morgan Kaufmann, San Francisco, California (1993)
22. Kitano, H., Shimazu, H.: The Experience-Sharing Architecture. In D. Leake (ed.), Case-Based Reasoning Experiences: Lessons Learned & Future Directions (1996)
23. Lam, W., Shankararaman, V.: Managing Change During Software Development: An Incremental, Knowledge-Based Approach. Proc. of the 10th Int. Conf. on Software Engineering and Knowledge Engineering, San Francisco, California (1998)
24. Manago, M. et al.: Casuel: A Common Case Representation Language. Technical Report Deliverable D1, Esprit Project Inreca P6322 (1994)
25. Maiden, N.A., Sutcliffe, A. G.: Exploiting Reusable Specifications Through Analogy. Communications of the ACM, 35(4) (1992)
26. Kempter, H., Leippert, F.: Systematische Software-Qualitätsverbesserung durch zielorientiertes Messen und Bewerten sowie explizite Wiederverwendung des Software-Entwicklungs-Know-how. Proc. of the BMBF-Seminar Software Technology, Germany (1996)
27. Tautz, C., Althoff, K.-D.: Using Case-based Reasoning for Reusing Software Knowledge. Proc. of the 2nd Int. Conference on Case-Based Reasoning, Springer Verlag (1997)
28. Tautz, C., Gresse von Wangenheim, C.: REFSENO: A Representation Formalism for Software Engineering Ontologies. Proc. 5th German Conf. on Knowledge-Based Systems, Germany (1999).
29. Tautz, C., Gresse von Wangenheim, C.: REFSENO: A Representation Formalism for Software Engineering Ontologies. Technical IESE-Report 015.98/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany (1998).
30. Tversky, A.: Features of Similarity. Psychological Review, 84 (1977)

31. Wess, S.: Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik. Ph.D. Thesis, University of Kaiserslautern, Germany, infix Verlag (1995)
32. Zand, M., Samadzadeh, M.: Software Reuse: Current Status and Trends. Journal of Systems and Software, 30 (3) (1995)