Experiences with Systematic Reuse: Applying the EF/QIP Approach*

Raimund L. Feldmann, Jürgen Münch, Stefan Vorwieger

Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 67653 Kaiserslautern, Germany
r.feldmann@computer.org, {muench, vorwiege}@informatik.uni-kl.de

Abstract. *Reusing experience in the form of processes, products, and other forms of knowledge is essential for improvement, that is, reuse of knowledge is the basis for improvement [1]. Comprising approaches to systematically support reuse are the Quality Improvement Paradigm and the Experience Factory. This paper describes experiences with the application of these approaches to the project CoDEx which aimed at developing a real-time house automation system. A prototype implementation of an experience base which is tailored for real-time systems is shown. Subsequently the conduction of the project is described. The focus is set on the reuse and experience-based modification of an effort model. Lessons learned and achieved benefits are discussed. First experiences showed on the one hand that the application of the approaches supported the project planning enormously. On the other hand process, product, knowledge and quality models need to be better defined according to the evolutionary nature of software development. Based on experiences, possible improvements of our experience base implementation are described.*

1 Introduction

Key technologies for supporting quality improvement include modeling, measurement and reuse of software development knowledge in form of products, processes and experience originating from the software life cycle. Software Engineering offers a framework based on an evolutionary quality management paradigm tailored for software development, the *Quality Improvement Paradigm (QIP)* [2]. It is supported by an organizational approach for building software competencies and transferring them to projects, the *Experience Factory (EF)* [1].

The EF/QIP approach has been applied to a project called CoDEx which aimed at developing a real-time house automation system. The project was conducted in the context of the Sonderforschungsbereich 501 (SFB), a long-term strategic research activity. Its goal is to develop and evaluate a set of techniques, methods and tools that support the fast and reliable customization of complex domain-specific software systems. SFB projects are regarded as *experiments* because their main focus is to learn about the development techniques and methods, and the minor focus is on the product development itself. Two types of experiments are considered: case studies (such as CoDEx) and controlled experiments. The key discriminator between case studies and controlled experiments is the control over the independent variables.

During the execution of SFB projects knowledge kept in a prototype *Experience Base of the SFB (SFB-EB)* is used and maintained. For the SFB-EB and CoDEx we stated an experience model (see Fig. 1) which is an adaption of basic principles of the EF/QIP approach. Experience elements are regarded as all kinds of software engineering experience, especially models (such as process models, product models, quality models), instances (such as process traces, products, measurement data, techniques, tools) and qualitative experience (such as lessons learned). For each experience element the scope of its validity is described. The scope consists of a context vector and the significance. The context vector characterizes the environment in which the experience element is valid. The significance describes how the experience element has been validated and to which extent.

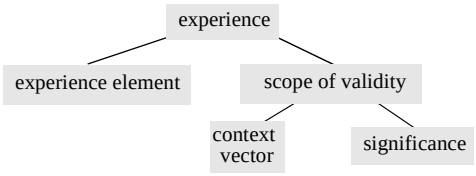

Fig. 1: Experience model

2 Adoption of the EF/QIP Approach

This section sketches our instantiation of the EF approach and how the approach was applied in the reuse process of the CoDEx project that was performed according to the QIP.

2.1 An Instantiation of the EF Approach

Reuse activities in the SFB are based on the EF approach. A central component is the *Experience Base of the SFB* which acts as a repository for all kinds of experience. In our prototype implementation we distinguish between an *experiment-specific section* and an *organization-wide section.*

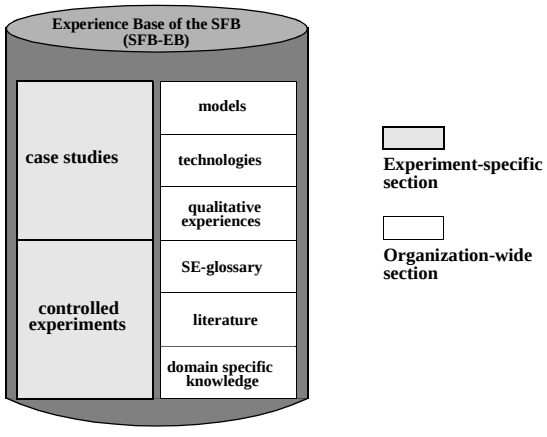

Fig. 2: Structure of the SFB-EB

As shown in Fig. 2 the experiment-specific section contains data of *case studies* and *controlled experiments*. For

* This work is supported by the Deutsche Forschungsgemeinschaft as part of the Sonderforschungsbereich 501 “Development of Large Systems with Generic Methods”.

each experiment the corresponding experience elements (such as hypothesis and measurement data) are stored in a separate location. Up to now the experiment-specific section holds 41 MB of data, 30,8 MB from five case studies and 10,3 MB from two controlled experiments.

The organization-wide section stores experience relevant to different variants of experiments (such as generic process models). It consists of six different areas holding a total of 38,4 MB of data. In the *model area* process models, product models, resource models and quality models are stored using the process modeling language MVP-L [3]. Most of these models have been adopted from outside the SFB to provide an initial set of experience elements. One example is an MVP-L model describing how to develop real-time systems according to the method of Bræk & Haugen [4]. For different techniques, methods and tools so-called technology packages are stored in the *technologies area*. Those packages contain basic information about the technologies and help to select the appropriate techniques when setting up a new experiment. Lessons learned from all experiments are stored in the *qualitative experiences area*. The *SE-glossary*, providing consistent definitions of Software Engineering terminology throughout all experiments, and relevant *literature* are also forming separate areas in the organization-wide section. Finally the *domain specific knowledge area* provides expert knowledge about real-time house automation such as a collection of thermodynamic formulas for heating systems.

Both the experiment-specific and the organization-wide section are connected by links between the experience elements according to some "*is a*" structures and "*used*" structures. For example the project plan of CoDEx was derived from the waterfall model which is stored in the model area. So the CoDEx project plan *is a* waterfall model instance. Furthermore CoDEx *used* OMT for the analysis. So there is a link from CoDEx to the technology package for OMT in the technologies area. The links are implemented in both directions, from experience elements in the experiment-specific section to those in the organization-wide section and vice versa.

2.2 Applying the QIP

With focus on the quality model "effort" the following gives an insight in how the six steps of the QIP cycle were performed in CoDEx.

QIP 1: Characterize. In the SFB-EB every experiment is bound to a *context vector* consisting of context factors for characterization and identification. In case of a new experiment the context vector is instantiated as complete as possible by the project planner. This information is needed to search for similar experiments in the SFB-EB in order to retrieve appropriate reuse candidates (see Tab. 1). To reduce the number of search hits the context vector of planned experiments may contain hypothetical entries. For CoDEx the context vector was instantiated completely and the most similar project was chosen as our reuse candidate. Both vectors are shown partially in Tab. 1 with differences marked grey.

	Context factors	CoDEx	Reuse candidate
Context vector	Application domain	Reactive systems	Reactive systems
	# of developers	4 research assistants, 1 student	3 students
	Experience of developers	Medium	Medium
	Method of analysis	OMT (with StP)	OMT (with StP)
	Range of functionality	• Building automation system • GUI, …	• Building automation system
	Life cycle model	Waterfall	Waterfall
	Reliability	High	Medium/low

Tab. 1: Context definition for CoDEx and a search hit

QIP 2: Set goals. Usually case studies examine several experimental goals. For CoDEx one experimental goal was the examination of the effort distribution formally defined in Tab. 2 according to the GQM paradigm [2]: Given the

Object	Purpose	Quality Focus	Viewpoint	Context
Development Process	Characterization	Effort	Developer	SFB 501

Tab. 2: Goal definition in accordance with GQM

CoDEx development process the effort should be measured for characterization. To gain a hypothesis the effort model of the most similar project identified in QIP 1 was reused as a basis. Differences in both context vectors pointed out that the original effort model had to be adapted. From the viewpoint of the developers a closer look to the context factors resulted in the assumption that in all three cases of deviation the estimated effort had to be increased. QIP 2 resulted in the hypothetical effort model shown in Fig. 3.

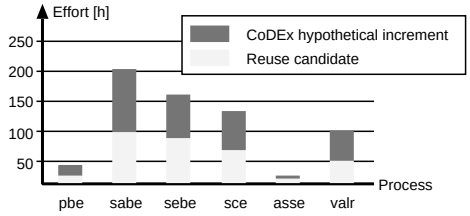

Fig. 3: Hypothetical effort model

QIP 3: Choose process. The coarse-grained process model underlying the effort model was refined and adapted in accordance with the special context of the new project. In CoDEx this was modeled in MVP-L. The hypothetical effort model was integrated into the process model in a way that an *invariant* was bound to every process. Invariants are of the form

process_id.effort ≤ *x*, e.g. sabe.effort ≤ 210 h

and serve as a trigger to show any deviation from the hypothesis. Additionally every process was instrumented for measurement. This supports the developer to record his effort on corresponding questionnaires. Every adaption of the process model and effort model was marked to support traceability of changes.

QIP 4: Execute. During the development of the software system, effort data were captured according to the project plan via questionnaires, and the process trace was recorded. Because the effort model depends highly on the process model the process trace is used to additionally explain deviations in the effort model. In CoDEx the process trace exhibited a process called *kebe* that was originally not considered in the project plan. Likewise the effort data for *kebe* had to be gathered for future reuse of the effort model and the process model.

QIP 5: Analyze. The analysis of the collected data included the comparison of the hypothetical effort model

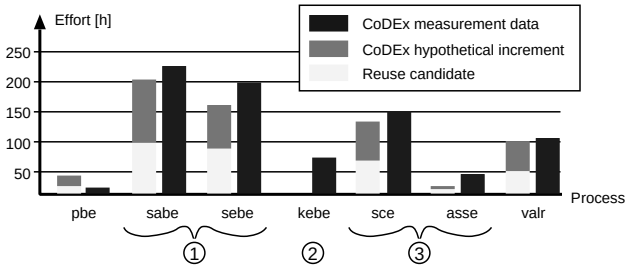

Fig. 4: Hypothesis versus measurement data

and measured data as well as the explanation of deviations found between them. It had to be decided which deviation could be relevant for future projects. On that basis proposals for new reusable effort models were made. Comparing the hypothesis with the measurement data in CoDEx indicated the following deviations:

a.) No hypothetical effort baseline for process *kebe* existed (see ¡ in Fig. 4).
b.) The effort was significantly higher for the processes *sabe*, *sebe*, *sce* and *asse* (see ¿ and ¬ in Fig. 4).

The explanation for a.) was given in QIP 4 by the process trace: in CoDEx a new process was needed for the system development. This was considered as a necessary change that should be taken into account for future planning. The explanation for b.) was a dynamic replanning cycle from *asse* back to *sabe* which was considered to be a unique event. In this repetition a subprocess of *sabe* (*sabe-opt*) was omitted. It was not clear whether *sabe-opt* should be executed in future projects or not. Consequently QIP 5 resulted in two new effort models prepared as entries for the organization-wide section of the SFB-EB. Both models, called *CoDEx eff1* and *CoDEx eff2* (see Fig. 5), show 10% less effort - compared to the measured data - for those processes affected by the replanning, and take the process *kebe* into account. They differ from each other in the way that *CoDEx eff1* considers *sabe-opt* as a part of the development process which *CoDEx eff2* does not.

QIP 6: Package. The adapted models *CoDEx eff1* and *CoDEx eff2* needed to be integrated into the SFB-EB to be available for reuse. The context vector had been extended with a new context factor, e.g. “use of subprocess *sabe-opt*”. For *CoDEx eff1* this factor was instantiated with “yes”, for *CoDEx eff2* with “no”. Related effort models in the SFB-EB (such as the reuse candidate) were given the default value “yes” because they were performed with *sabe-opt* (see Fig. 5). Additionally the significance entry was updated for the reused and the newly gained experience elements.

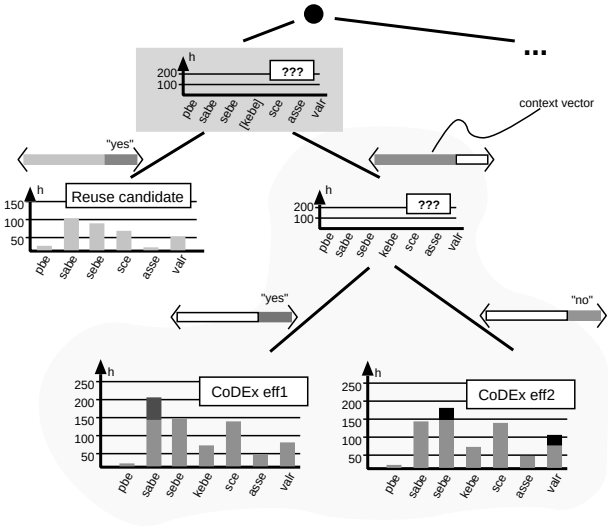

Fig. 5: Visualizing the structure of the organization-wide section

3 Lessons Learned and Achieved Benefits

Due to the basics of the EF/QIP approach reuse and maintenance of experience could be performed in a systematic and explicit manner. Experiences with CoDEx and other SFB projects showed that it was difficult to formulate an appropriate context vector because many of the factors which may influence the project were only assumed as relevant by subjective judgement. Context vectors of similar projects in the experience base gave a useful orientation. Planning and goal setting was very well supported by the availability of baseline models in the SFB-EB. Additionally, deviations in the context vectors gave valuable hints on how to tailor these models to the specific project characteristics. The exact definition of measurement goals and activities allowed a precise acquisition of the relevant measurement data. Non-anticipated events (such as replanning activities) should be carefully documented, e.g. by a process trace, so that they can be considered during analysis. An analysis of the threats to validity has to be performed to become aware of the unknown factors that may influence the results without our knowledge. Problems that prevent generalizing the results should also be identified. The packaging revealed a lack of appropriate mechanisms to cope with variants of experience elements. Models should be extended with (empirically derived) guidelines on how to adapt the models to project-specific goals and characteristics. Appropriate representation styles for generic knowledge and suitable tailoring mechanisms are needed.

4 Outlook

Future activities concerning the SFB-EB implementation are twofold: First, we will have to improve the schema of the SFB-EB. A more specific attribute framework has to be defined to provide an appropriate configuration management and a better access to the single experience elements. Correspondingly an advanced search and retrieval system based on the factors of the context vectors must be provided. A system that rates the quality of the retrieved experience elements in terms like “element fits to 90%” would be especially helpful. It should be a long term activity to investigate the quantitative impact of known factors of the context vectors as well as the identification of new factors. Second, we have to add more experience elements into the SFB-EB to provide a larger search space, so the returned sets of reuse candidates will offer more alternatives to choose from. This problem can only be solved by conducting more experiments in the context of the SFB.

Bibliography

[1] Victor R. Basili, Gianluigi Caldiera and H. Dieter Rombach. The Experience Factory. In *Encyclopedia of Software Engineering.* John. J. Marciniak (Ed.), Volume 1, pp. 469-476, John Wiley Sons, 1994.

[2] Victor R. Basili and H. Dieter Rombach. The TAME Project: Towards Improvement–oriented Software Environments. *IEEE Transactions on Software Engineering*, SE-14(6):758–773, June 1988.

[3] Ulrike Becker, Dirk Hamann, Jürgen Münch and Martin Verlage. MVP-E: A Process Modeling Environment. IEEE TCSE Software Process Newsletter, Volume 10, Khaled El Emam (Ed.), 1997.

[4] R. Bræk and O. Haugen. Engineering Real-time Systems: An Object-oriented Methodology using SDL. Prentice Hall, New York, London, 1993.