Developing Industrial Case-Based Reasoning Applications
Using the INRECA Methodology

Ralph Bergmann
University of Kaiserslautern
Department of Computer Science
PO-Box 3049
D-67653 Kaiserslautern
EMail: bergmann@informatik.uni-kl.de
wwwagr.informatik.uni-kl.de/~bergmann

Mehmet Göker
DaimlerChrysler Research and Technology
Adaptive Systems Group
1510 Page Mill Road, Palo Alto
CA 94304
mehmet.goeker@daimlerchrysler.com

Abstract

This paper presents an overview of the INRECA methodology for building and maintaining Case-Based Reasoning applications. It is based on the experience factory and the software process modeling approach from software engineering. CBR development experience is documented using software process models and stored in different levels of generality in a three-layered experience base. The current INRECA experience base contains three so-called recipes, each of which describes the development of one kind of CBR application. The INRECA methodology has been used for developing several new CBR applications and significant benefits in terms of productivity, quality, communication, and management decision making could be identified.

1 Introduction

Today, contemporary IT companies are facing a market that demands large-scale CBR projects and CBR software that fulfills quality standards. Therefore, a systematic and professional *methodology* for developing CBR applications is mandatory. A methodology combines a number of methods into a philosophy that addresses a number of phases of the software development life cycle [Booch, 1994]. The methodology gives the guidelines for the activities that need to be performed in order to successfully develop a certain kind of software, in our case, a CBR application. These guidelines must be expressed in a well-defined terminology that enables the exchange of software development experience to continuously improve the development process. Using an appropriate methodology should provide significant, quantifiable benefits in terms of

- *productivity*, e.g., reducing the risk of wasted efforts,

- *quality*, e.g., including quality deliverables,

- *communication*, i.e., formal and informal communication among members of the development team, and

- *management decision making,* e.g., planning, resource allocation, and monitoring [Bergmann et al. 1997].

Currently, there are several activities with the goal of establishing a methodology for building case-based reasoning applications. Contributions can be found in books on CBR [Kolodner 1993, chapter 15; Wess 1995, chapter 6] and in papers collecting the experience of people who have successfully developed CBR applications (e.g. [Kitano and Shimazu 1996, Lewis 1995; Bartsch-Spörl 1996a; Curet and Jackson 1996]); most valuable experience-based contributions arose from projects where methodology development was explicitly included as a project task, like INRECA[1] [Althoff, Wess et al. 1995; Johnston, Breen, and Manago, 1996] or APPLICUS[2] [Bartsch-Spörl 1996b, 1996c].

One of the main driving forces behind the development and use of a methodology is the need for quality in both the products and processes during development of computer-based systems. This general observation holds for software development in general and for CBR application development in particular. Some methodologies and approaches to software development in general already exist, but software engineering methodology is still a major issue in current Software Engineering (SE) research. By adapting techniques originating in this area, we present a methodology based on recent SE techniques, which have been enriched by the experience of building and maintaining CBR applications.

---

[1] INRECA: Esprit Project P6322. Partners: Acknosoft (prime contractor, France), TECINNO (Germany), Interactive Multimedia Systems (Ireland), University of Kaiserslautern (Germany)

[2] APPLICUS: Esprit Trial Application P20824. Partners: Acknosoft (prime contractor, France), BSR Consulting (Germany), Sepro Robotique (France)
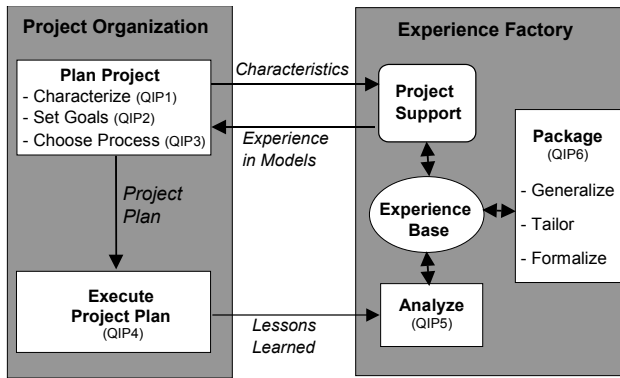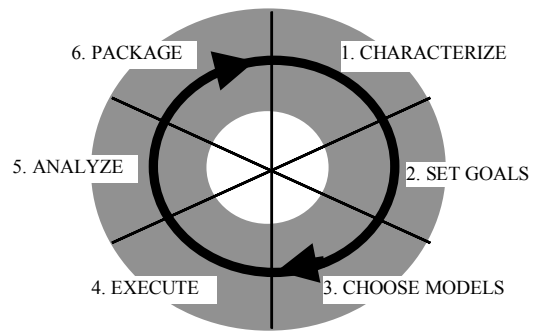
Figure 1. The experience factory

Figure 2. The quality improvement paradigm (QIP).

This paper gives a summary of the INRECA methodology. This methodology was developed during the ESPRIT project INRECA-II[3] and collects the CBR development experience from four industrial partners and the University of Kaiserslautern. For a complete description please consult the book [Bergmann et al. 1999] as well as the Web page of the INRECA Center at http://www.inreca.org.

2 Foundations of the INRECA Methodology

The INRECA methodology has its origin in recent software engineering research and practices. It makes use of a software engineering paradigm that enables the reuse of software development experience by an organizational structure called *experience factory* [Basili et al., 1994]. An *experience factory* is an organizational unit within a software development company or department that supports capturing and reusing software development experience and thereby supports project planning. It links with project execution so that lessons learned from previous projects can be reused. In the INRECA methodology, the experience factory provides the organizational framework for storing, accessing, and extending the guidelines for CBR application development, which are the core assets of the methodology.

The guidelines themselves are documented in a process-oriented view. *Software process modeling* [Rombach & Verlage, 1995] is a well-established area in software engineering that provides fundamentals for this. Software process models describe the flow of activities and the exchanged results during software application development.

In a nutshell, the INRECA methodology consists of collected CBR development experiences, represented as

software process models and stored in the experience base of an experience factory. The experience factory itself provides the organizational structure to access this experience and to keep it alive and up-to-date. We now describe the experience factory paradigm and the basics of software process modeling in more detail.

2.1 The Experience Factory and the Quality Improvement Paradigm

The experience factory approach [Basili, et al. 1994] is motivated by the observation that any successful business requires a combination of technical and managerial solutions: a well-defined set of product needs to satisfy the customer, assist the developer in accomplishing those needs, and create competencies for future business; a well-defined set of processes to accomplish what needs to be accomplished, to control development, and to improve overall business; and a closed-loop process that supports learning and feedback.

An experience factory is a logical and/or physical organization that supports project development by analyzing and synthesizing all kinds of experience, acting as a repository for such experience, and supplying that experience to various projects on demand. An experience factory packages experience and collects it in an *experience base*. The experience consists of informal or formal models and measures of various software processes, products, and other forms of knowledge. Figure 1 shows a schematized description of the experience factory, its steps, and its relation to the project organization. The project organization is in charge of *planning* and performing (*plan execution*) the software development project in an IT company. The experience factory *analyzes* the lessons learned from executed projects and records them in the experience base. To enable best reuse, these experiences are further *packaged*, i.e., they are generalized, tailored, and formalized. Further, the experience factory provides access to its experience base and thereby *supports* the project planning, which goes on in the project organization.

All these steps are organized into a cycle that enables the quality of the software development to be improved continuously. This cycle is called *the quality improvement paradigm* (QIP, see Fig. 2) and orders the six basic steps

---

from the experience factory. In more detail, the six QIP steps are:

Characterize (QIP1)

The aim of this step is to characterize the project and its environment based on the available information. Normally, a large variety of project characteristics and environmental factors can be used for this characterization, such as the application domain, susceptibility to changes, problem constraints, techniques, tools, and so on.

Set Goals (QIP2)

There are a variety of viewpoints for defining goals, like those viewpoints of the user, customer, project manager, corporation, and so on. Goals should be measurable, depending on the business models.

Choose Process (QIP3)

On the basis of the characterization, the goals, and the previous experience from the experience base, appropriate processes for implementing the project must be chosen. This results in the overall project plan.

Execute (QIP4)

The project plan is enacted, causing the development project to be carried out. For further analysis, respective records of the development process must be made.

Analyze (QIP5)

At the end of each specific project, the data collected must be analyzed to evaluate current practices. Valuable, reusable experience must be identified.

Package (QIP6)

The experience in the experience base consists of a variety of models. These models have to be defined and refined. Such models can be, e.g., resource models, process definitions and models, quality models, lessons learned, and so on. On the basis of the new experience, these models might get generalized, tailored to a particular kind of situation, or formalized, so that they can be reused in other projects.

The experience factory and the quality improvement paradigm, provide a mechanism for continuous improvement through the experimentation, packaging, and reuse of experiences based on the needs of a business.

2.2 Software Process Modeling

Within the INRECA methodology, software process modeling [Rombach and Verlage, 1995] is the means for documenting CBR experience. It provides a well-defined terminology. Software process models describe the engineering of a product, e.g., the software that has to be produced. Figure 3 shows a graphical representation of its main elements, i.e., processes, products, and methods.

Processes

A process is a basic step that has to be carried out in a software development project. It is an activity that has the goal of transforming some *input product*(s) into some
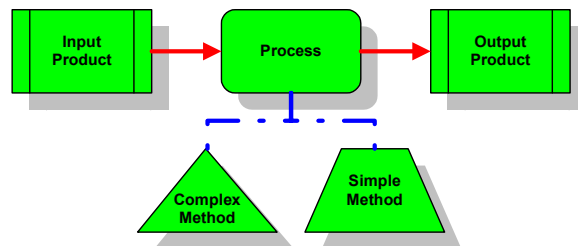

Figure 3. Basic elements of a process model.

output product(s). To enact a process, different alternative *methods* may be available that allow the required output products to be created in a systematic way, a process is typically defined by the following properties:

- A particular *goal* of such a basic step. The goal specifies *what* has to be achieved.

- A set of different alternative *methods* that can be used to implement the step. Such a method specifies one particular way of carrying out the process, i.e., one way of reaching the process's goal.

- *Input*, *output,* and *modified products* that describe which products are required at the beginning of the step, which products must be delivered at the end, and which products are changed during enactment.

- A set of *resources* (agents or tools) that are required to perform the step. Here the necessary qualifications or specifications that an agent or a tool must have so that (s)he/it can be assigned to the process are defined.

Methods

Methods contain a detailed specification of a particular way of reaching the goal of a process. A method can be either *simple* or *complex*. While a simple method provides only a description of what to do to reach the goal of the associated process, a complex method specifies a set of subprocesses, a set of intermediate products (called *byproducts*), and the flow of products among the subprocesses. This allows the definition of very flexible process models in a hierarchical manner, because very different process refinements can be described by using alternative subprocess models.

Products

The main goal of processes is to create or modify products. Products include the executable software system, as well as the documentation, like design documents or user manuals.

Resources

Resources are entities necessary to perform the tasks. Resources can be either *agents* or *tools*. Agents are models for humans or teams (e.g., managers, domain experts, designers, or programmers) that can be designated to perform a processes. The most relevant properties of agents are their qualifications. Tools (e.g., a modeling tool, a CBR tool, or a GUI builder) are used to support the en-
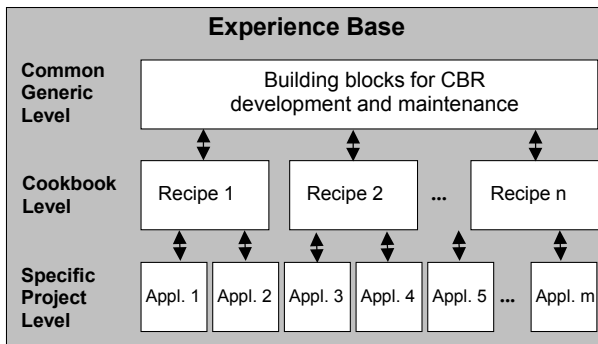
Figure 4. Structure of the experience base.

actment of a process and can be described by a specification. Therefore, by using the required qualifications and specifications defined in the generic process, it is possible to determine available agents and tools that can be assigned to a certain process.

3 The INRECA Experience Base

The basic philosophy behind the INRECA methodology is the experience-based construction of CBR applications. The approach is particularly suited because CBR application development is an activity that relies heavily on experience. In certain areas, application development is already a routine task with clearly identified processes. This holds, for example, in the area of simple electronic product catalogs or simple help-desk applications. However, in recent years, CBR has entered new, important application areas that require large-scale application development. During the course of INRECA, three major application areas have been explored and large-scale industrial applications developed. The experiences of these developments have been captured and are the core of the INRECA methodology (see [Bergmann et al., 1999] and http://www.inreca.org for details). We expect the market for many new CBR applications to open up in the future. Therefore, the involved IT companies need to build up their own area-specific experience. To explore such new fields systematically, the experience must be captured, represented, and accumulated for reuse. The experience factory idea in the INRECA methodology is perfectly suited for this purpose. A second big advantage in the context of a CBR methodology is that the basic idea behind the experience factory is closely related to the general idea of CBR, i.e., retrieving, reusing, revising, and retaining experience [Althoff et al., 1998; Althoff and Wilke, 1997]. The experience factory stores specific experience, namely software engineering experience, in a case base, which we call an *experience base*. Thus, the motivation and rationale behind the experience factory approach should be easily understandable by people who are familiar with the Case-Based Reasoning approach in general.

3.1 Experience Captured in Software Process Models

In the INRECA methodology, software process models represent the CBR development experience that is stored in the experience base. Such software processes can be very abstract, i.e., they can represent some very coarse development steps such as domain model definition, similarity measure definition, and case acquisition. Or they can be very detailed and specific for a particular project, such as analyzing data from Analog Device Inc. operational amplifier (OpAmp) product database, selecting relevant OpAmp specification parameters, and so on. The software process modeling approach allows the construction of such a hierarchically organized set of process models. Abstract processes can be described by complex methods, which are themselves a set of more detailed processes. We make use of this property to structure the experience base.

The experience base is organized on three levels of abstraction: a *common generic level* at the top, a *cookbook level* in the middle, and a *specific project level* at the bottom [Bergmann et al. 1998]. These levels are shown in Fig. 4.

3.2 Common Generic Level

At this level, processes, products, and methods are collected that are common for a very large spectrum of different CBR applications. The documented processes usually appear during the development of most CBR applications. The documented methods are very general and widely applicable, and give general guidance for how the respective processes can be enacted. At this common level, processes are not necessarily connected to a complete product flow that describes the development of a complete CBR application. They can be isolated entities that can be combined in the context of a particular application or application class.

The current common generic level of the INRECA methodology covers managerial, technical, and organizational aspects of development processes for analytical CBR applications. It defines processes such as: *project definition*, *feasibility study*, *management & monitoring, organizational development*, *training*, and *technical development,* including *knowledge acquisition and modeling*, *GUI development*, and *integration with existing IT environment*. Overall, 120 processes, products, and methods are defined.

3.3 Cookbook Level

At this level, processes, products, and methods are tailored for a particular class of applications (e.g., help desk, technical maintenance, product catalogue). For each application class, the cookbook level contains a so-called *recipe*. Such a recipe describes how an application of that kind should be developed and/or maintained. Thus process models contained in such a recipe provide specific guidance for the development of a CBR application of this application class. Usually, these items are more con-

crete versions of items described at the common generic level. Unlike processes at the common generic level, all processes that are relevant for an application class are connected and build a product flow from which a specific project plan can be developed.

The current cookbook-level of the INRECA methodology consists of three recipes:

- Help desk support for complex technical equipment
- Intelligent catalog search applications
- Technical Maintenance applications

Each of these recipes contains an elaborated and proofed process model for application development, each of which consists of more than 100 processes, products, and methods. In the following we give a brief overview of the existing recipes.

Help Desk Support for Complex Technical Equipment

This recipe describes the processes that must be performed to develop a case-based system to support help-desk operators in diagnosing complex technical equipment. The recipe was created during the development of the HOMER system [Göker et al. 1998] and has been applied to the development of several other help-desk systems. The goal of developing a case-based help-desk support system is to create a knowledge repository that contains problem-solving experiences for a complex technical domain that changes over time. This knowledge repository will be used in an organization, by a group of people with varying levels of expertise, in a time-critical operation. It is obvious that the development and use of such a system does not only involve technical processes, but also raises managerial and organizational issues. The recipe describes the tasks that must be performed to develop a case-based help-desk support system and the processes that have to be put into place to make such a system operational. Some more details about this recipe are given in section 5.

Intelligent Catalog Search Applications

Intelligent catalog search applications are an important application of CBR to electronic-commerce. When developing an intelligent catalog search application we simply treat the product list as the case base and the user's requirements for a product as the new problem. The characteristics of this problem are as follows:

- users need to search the product space in its entirety,
- products in the product space will be similar is terms of their characteristics and their structure,
- the user generally does not need an exact match; the optimum match will suffice,
- the user needs the ability to *negotiate* with the search system,
- no match found is an unacceptable outcome from a search, (i.e., "sellers want to sell."

This recipe was developed from the operational amplifiers catalog application developed for Analog Devices [Vollrath et al. 1998]. This catalog is available via CDROM and the Web.

Technical Maintenance Applications

In technical maintenance, operators and manufacturers of complex equipment often collect data about the equipment they maintain: technical follow-up files, breakdown files, intervention reports, preventive maintenance reports, and records of requests from their clients or distributors. Unfortunately, this information is usually not well exploited. This is due both to the complexity of the equipment itself and to the type of technicians who carry out the maintenance processes. The recipe for this kind of applications suggests a specific case structure and modeling approach. It also addresses typical problems for linking the developed maintenance system to both on-board equipment and technical information. This recipe emerged from the development of a maintenance application for the French TGV trains.

3.4 Specific Project Level

The specific project level describes experience in the context of a single, particular project that has already been carried out. It contains project-specific information, such as the particular processes that were carried out, the effort that was required for these processes, the products that were produced, the methods that were used to perform the processes, and the people who were involved in executing the processes. It is a complete documentation of the project, which is more and more important today to guarantee the quality standards (e.g., ISO 9000 or SPICE) required by industrial clients. During the course of the INRECA project, more than 12 specific projects have been documented. For each of the above mentioned recipes one specific project documentation is publicly available on the CDROM attached to [Bergmann et al., 1999] and on the INRECA Center Web site at http://www.inreca.org.

4 Sketch of an Example Recipe

This help-desk recipe is one out of three recipes currently contained in the INRECA experience base. It describes the processes that must be performed to develop a case-based system to support help-desk operators in diagnosing complex technical equipment. It was created during the development of the HOMER application [Göker et al., 1998]. One of the goals of the implementation of HOMER was to observe and record the processes that have to be enacted during system development and use. In the course of developing the methodology, we documented the tasks that were performed, their duration, the resources needed, and the outcome of the tasks. While monitoring our own actions, we discovered redundancies as well as dependencies among processes that were not obvious. This enabled us to structure, optimize, and understand some of the processes better.

The overall process for the development and introduction of a case-based help-desk support system can be divided into four sub-processes:

- project planning and initialization
- rapid prototype development & evaluation
- system implementation
- system utilization

Figure 5 shows the overall development process showing these four sub-processes as well as the products that are exchanged. The complete recipe contained in the experience base describes all those processes, products, and methods in large detail. Due to the space limitations of this paper we now only sketch the four sub-processes here. Each of the methods to realize these sub-processes is again described by a complete process model each of which contains between 4 and 6 sub-sub-processes.

Project Planning and Initialization

Based on the project vision, the initial project plan for the development of the case-based help desk support system has to be created. This process includes the exact definition of the project goals, awareness creation and motivation, the selection of the initial domain, the selection and training of the project team and the test users, the creation of the initial requirements definition document and the selection of the CBR Tool to be used.

Rapid Prototype Development & Evaluation

The implementation of the rapid prototype enables the project team to test the validity of the goals and the re-
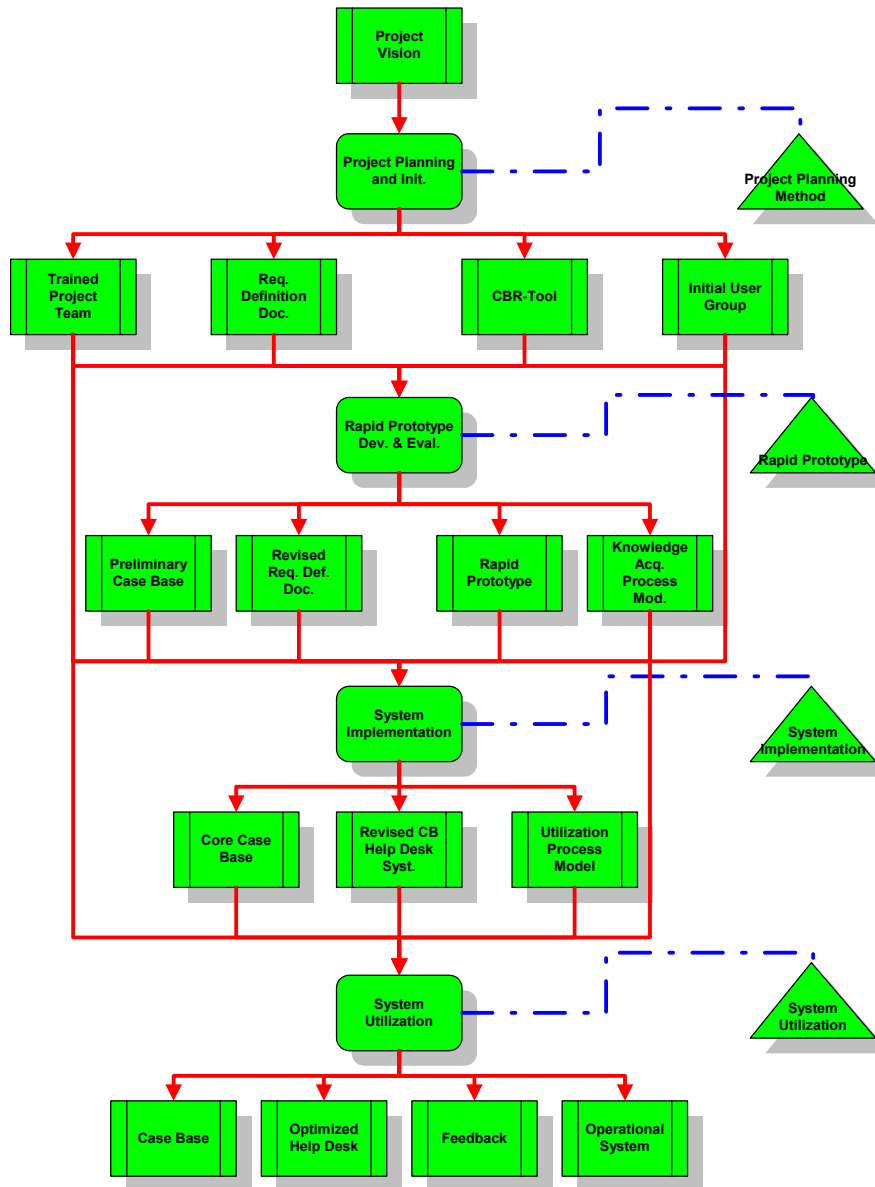
Figure 5. Overview Process Model for Help-Desk

quirements set forth in the project-planning phase. It also

serves to train the project team in knowledge acquisition techniques and in using the CBR tool. The processes that will be used during knowledge acquisition can be defined and refined.

System Implementation

The implementation of the integrated case-based help-desk support system contains the development of the actual system to be used at the help desk. The implementation is based on the preliminary case base, the revised requirements, the rapid prototype, input from the project team, and the process model for knowledge acquisition. While the system is implemented, the project team acquires the core-case base to be deployed with the system after implementation is complete. The processes to be used when the system is used are also developed in close connection with the implementation and the knowledge acquisition in this phase. During the evaluation and revision of the case-based help-desk support system the system is evaluated by the initial users and revised according to the results of this evaluation.

System Utilization

The integrated case-based help desk support system is being put into operation. Continuous knowledge acquisition and system maintenance are performed according to the utilization process model. The impact of the system is continuously monitored and the system as well as the utilization process is revised accordingly.

5 Applying the INRECA Methodology

When a new CBR project is being planned, the relevant experience from the experience base must be selected and reused. This relates to the steps QIP1 to QIP3 of the quality improvement paradigm (see Fig. 2). The recipes at the cookbook level are particularly useful for building a new application that directly falls into one of the covered application classes. The recipes are the most valuable knowledge captured in the methodology. Therefore, one should first investigate the cookbook-level to identify whether a cookbook recipe can be reused directly. If this is the case, the new CBR project can be considered a standard CBR application, and the processes from the respective recipe can provide immediate guidance for setting up the new project. On the other hand, if the new project does not fall into one of the application classes covered by the recipe, then the new project enters a new application area. In that case, the new project plan must be constructed from scratch. However, the processes described in the common generic level can be used to assemble the new project plan. Figure 6, shows the overall procedure for applying the INRECA methodology. It is a special variant of steps 1-4 of the quality improvement paradigm.

1. Characterize New CBR Project: Identify the application area of the new CBR project. The goal is to decide whether an existing recipe from the cookbook level of the methodology covers this application.

2. Recipe Available: Identify whether an appropriate recipe is available. If this is the case then continue with step 3a; otherwise continue with step 4a.

3a. Analyze Processes from Recipe: An appropriate recipe is available. The process model contained in this recipe must now be analyzed to see whether it can be mapped and tailored to the application at hand.

3b. Select Similar Specific Project: Analyze the project descriptions on the specific project level. If a similar project is available then it should be identified as to whether some of the specific experience can be reused within the scope of the current project. This might be a specific way to implement a certain process or software component. It can help to identify existing software components from previous projects that can be reused immediately on the code level.

3c. Develop New Project Plan by Reuse. Develop a project plan for the new project. This project plan is based mainly on a process model from the selected experience recipe. However, application-specific tailoring and pragmatic modifications are typically required. If components from similar projects can be identified for reuse, the project plan must take care of this fact to avoid re-development efforts.

4a. Analyze Processes from Common Generic Level. The set of processes described at the common generic level should be analyzed in the context of the new project. The goal is to identify those processes that are important for the new application. These processes can be considered the building blocks from which the new project plan can be assembled.
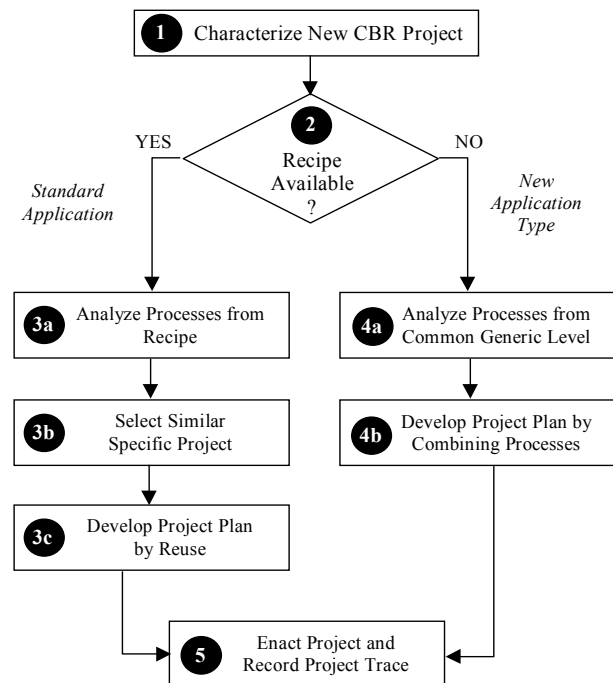

Figure 6. How to use the experience base.

4b. Develop Project Plan by Combining Processes. Based on the selected processes, a new project plan must be assembled. For this purpose, the processes must be made more precise and operational. Depending on how innovative the new application area is, it might even be necessary to develop new methods or software components.

5. Enact Project and Record Project Trace. Execute the project by enacting the project plan. Document the experience during the enactment of this project. Particularly, note all deviations from the developed project plan. This is important for two reasons. First, to ensure that the development is performed according to the necessary quality standards and, second, to feedback the new experience into the experience base for reuse.

After a project is finished, it is very important that the experience and the lessons learned are not lost, but are captured for inclusion into the experience base, which is the core of the INRECA methodology. This is necessary for continuous improvement of the CBR software-development process. This maintenance activity is related to steps 5 and 6 of the quality improvement paradigm (see Fig. 2). For the INRECA methodology, a particular extension and update procedure for the experience base has been developed (see Fig. 7).

6. Analyze Project Trace. Collect all information that had been captured about the finished project. Analyze this information and identify the processes that were actually performed during the enactment of the project.

7. Add Specific Project. Document the project by creating a specific process description that accurately describes what has been done in the project. Add this process description to the specific project level of the INRECA experience base. If appropriate, create links to the cook-book-level recipes from which this project had been derived.

8. Create or Update Recipe (optional). Based on the experience collected from the new project, it might become necessary to update an existing recipe or even to create a completely new one. An existing recipe might need to be updated if the following situation was encountered: A recipe was used to setup the project plan for the current project (step 3a in Fig. 6) but the experience from enact-ing the project indicated significant deviations from this plan. In this case, an update of the recipe is required to cover this newly learned lessons.

A new recipe might need to be created, if CBR was applied to a new type of application that was not covered by any of the existing recipes (steps 4a+b in Fig. 7). The process description for the new specific project can then be generalized into a new cookbook-level recipe. For this purpose, all project-specific information must be abstracted so that only those pieces remain that are likely to be reused for new, similar applications. It is very likely that this generalization process can be achieved only after several projects of this new application area have been realized. This avoids generalizing single experiences and is more reliable.

9. Update Generic Level (optional). If during the previous steps some new generic processes, methods, or products can be identified that are of more general interest, i.e., relevant for more than one application class, then they should be added to the common generic level of the experience base.

6 Tool Support for Documenting the Experience

We can document a specific CBR project, a recipe for an application class, or common generic CBR development experience using the INRECA methodology tool. This tool describes projects in terms of a process diagram where each element of the diagram links to information about the process or product in a database. Processes, products, methods, agents, and tools being stored in the experience packet are documented using a set of different types of sheets. A sheet is a particular form that is designed to document one of the items. It contains several predefined fields to be filled as well as links to other sheets. The tool supports the creation of a web site for a given project, allowing the documentation to be shared among distributed members of a development team. The INRECA methodology tool is implemented using the Visio modeling software from Visio Corporation. Visio is a graphical software tool that supports professionals in diagramming systems, structures, and processes in business environments. The INRECA experience base in all three levels is available via the Web pages of the INRECA Center at http://www.inreca.org.

7 Impact of the Methodology in Developing and Using Help-Desk Support Systems

After the help-desk recipe sketched in section 4 was complete, the methodology and the developed tools were used during the *project definition*, *application development,* and *system utilization* phases of new projects. The methodology has an impact on *productivity*, *quality*, *communication*, and *management decision making*. We could observe the advantages of using the methodology
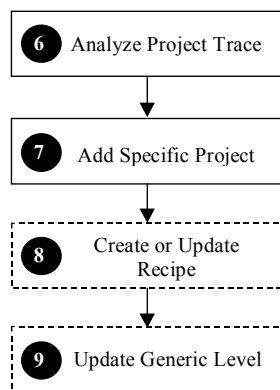
Figure 7. Adding new experience.

in each of these areas and in all three project phases, both to the customer (management and user) and to the developer.

7.1 Impact of the Methodology During Project Definition

During the definition of a project, the processes to be executed have to be detailed. For each process, this involves defining the methods to be used, the project's duration, the resources needed, the results to be produced, the interaction with other processes within the project, and the sequence in which the processes must be executed.

By means of the methodology we were able to create project definitions with structured process models, task definitions, roles and responsibilities, duration, and resource allocations. The methodology enabled us to make the case-based help-desk support system development process traceable both to the customers and to ourselves.

While the creation of the project definition for HOMER from scratch took us about three months, we were able to reuse the help-desk recipe to define three new projects in less than a week each. The overall duration of these projects varied between six months and three years. Since we reused a recipe that had been executed before, we could be sure that all aspects that needed be taken into account were covered. Since the basic recipe was available, we could concentrate on the peculiarities of each domain. The quality and level of detail we were able to achieve in these project descriptions were far beyond the level that can be achieved when a description is created from scratch.

Apart from these advantages in increased productivity and quality, the methodology was also very useful for communicating with the customer and conveying the message that the development of a CBR system is not an art, but, rather, solid technology. As we mentioned above, CBR is still considered an academic approach by some managers. Being able to describe each process in the development of a case based system in detail enabled us to convince managers of the validity of the approach, clarified the need for continuous maintenance and resource allocation, and let us raise realistic expectations. We were able to give customers figures on how much effort had to be spent by whom and when. Project progress became measurable. This gave them a basis for making decisions, enabled them to plan their resource allocation in advance, and prevented the loss of critical resources in the course of the project.

7.2 Impact of the Methodology During System Development

The detailed project plan that was created during project definition served as a guideline during the development of the system. The ability to trace a structured path and the use of the software tools that was developed to support the methodology allowed us to speed up the development process by a factor of 12. While the development

and testing of the first prototype in HOMER and comparable applications took up to six months from the initiation of the projects, we were able to create a prototype and test it at another site of DaimlerChrysler within two weeks.

Since the tasks that have to be performed and the results that have to be achieved during the implementation of the system were described in detail in the "recipe" and the project definition, some tasks could be transferred from one developer to the other without additional effort, and the progress that was achieved during the project could be measured. A clear-cut definition of the tasks to be performed and the availability of tools for creating and maintaining a domain model allowed us to use less-qualified personnel during the development of the system without compromising the quality of the resulting product. This allowed both the developer and the customer to optimize the allocation of personnel resources.

7.3 Impact of the Methodology During System Use

During the development and initial use of HOMER, we had to define and modify the tasks and qualifications of the personnel needed to operate the system several times. Since the processes for the acquisition, use, and maintenance of the knowledge in the case-based help-desk support system are defined in the methodology, we were able to introduce new help-desk systems in a much more efficient manner. While the domain modeling and maintenance task could only be performed by a highly qualified help-desk operator in the HOMER project, the domain modeling and model-maintenance tools, as well as the similarity editor that was developed to support the INRECA methodology, enabled us to use much-less-qualified personnel in the ensuing projects.

The detailed definition of the duties that have to be performed and the qualification that is needed for the project group also enabled the customers to allocate the necessary resources in advance and monitor the status of the project according to the goals that had been set when the project was initiated.

The methodology was also used to train users in the administration of a case-based help-desk support system. Using process charts, we explained the overall structure of the project to the operator who would be in charge of the project after it was completed. This allowed the operator to maintain and use the system without getting lost in details and neglecting important aspects, like maintaining the case base.

References

Althoff, K.-D., Wess, S., Weis, K.-H., Auriol, E., Bergmann, R., Holz, H., Johnston, R., Manago, M., Meissonnier, A., Priebisch, C., Traphöner R., & Wilke W.(1995). An evaluation of the final integrated system. *Deliverable D6 of the INRECA Esprit Project*.

Althoff, K.-D., Birk, A., Gresse von Wangenheim, C., & Tautz, C. (1998). CBR for experimental software engineering. In: M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, & S. Wess (Eds*.). Case-Based Reasoning Technology from Foundations to Applications*, Lecture Notes in Artificial Intelligence 1400, Chapter 9, Springer, 235-253.

Althoff, K.-D. & Wilke, W. (1997). Potential uses of case-based reasoning in the experience-based construction of software systems. In: R. Bergmann & W. Wilke (eds.), *Proceedings of the 5th German Workshop in Case-Based Reasoning (GWCBR'97),* LSA-97-01E, Centre for Learning Systems and Applications (LSA), University of Kaiserslautern.

Basili, V. R., Caldiera, G. & Rombach, H. D. (1994). The experience factory. In: J. J. Marciniak (ed.), *Encyclopedia of Software Engineering*, Vol. 1, Wiley, New York, 469-476.

Bartsch-Spörl, B. (1996a). Towards a methodology for how to make CBR systems work in practice. In: H.-D. Burkhard & M. Lenz (eds*.), Proceedings of the 4th German Workshop on Case-Based Reasoning (CBR-96)- System Development and Evaluation* - Informatik-Bericht No. 55, Humboldt Universität Berlin, 2-9.

Bartsch-Spörl, B. (1996b). How to introduce case-based reasoning in customer support*. Applicus Deliverable*.

Bartsch-Spörl, B. & Manago, M. (1996c). *Case-based reasoning: A leading edge technology for making active use of your experience*. Leaflet.

Bergmann, R., Breen, S, Göker, M., Manago, M, & Wess, S. (1999). *Developing Industrial Case-Based Reasoning Applications: The INRECA Methodology.* Lecture Notes in Artificial Intelligence, LNAI 1612, Springer, Berlin, Heidelberg.

Bergmann, R., Breen, S., Fayol, E., Göker, M., Manago, M., Schumacher, J., Schmitt, S., Stahl, A., Wess, S. & Wilke, W. (1998). Collecting experience on the systematic development of CBR applications using the INRECA-II Methodology. In Smyth, B. & Cunningham, P. (Eds.) *Advances in Case-Based Reasoning (EWCBR'98),* Lecture Notes in Artificial Intelligence, Springer, Berlin, Heidelberg, 460-470.

Bergmann, R., Wilke, W., Althoff, K.-D., Breen, S. & Johnston, R. (1997). Ingredients for developing a case-based reasoning methodology. In: Bergmann, R. & Wilke, W. (Eds.) *Proceedings of the 5th German Workshop in Case-Based Reasoning (GWCBR'97),* LSA-97-01E, Centre for Learning Systems and Applications (LSA), University of Kaiserslautern, 49-58.

Booch, G. (1994). *Object-oriented analysis and design. With applications*. Second Edition. Benjamin /Cummings Publishing Company.

Curet, O. and Jackson, M. (1996). Towards a methodology for case-based systems. *Expert Systems '96*. Proceed-

ings of the 16th annual workshop of the British Computer Society.

Göker, M., Roth-Berghofer, T., Bergmann, R., Pantleon, T., Traphöner, R., Wess, S., & Wilke, W. (1998). The development of HOMER: A case-based CAD/CAM help-desk support tool. In B. Smyth & P. Cunningham (Hrsg.) *Advances in Case-Based Reasoning (EWCBR'98),* Springer, 346-357.

Johnston, R., Breen, S., & Manago, M. (1996). Methodology for developing CBR applications. *INRECA Deliverable D30*.

Kitano, H. & Shimazu, H. (1996). The Experience-sharing architecture: A case-study in corporarte-wide case-based software quality control. In Leake (Ed.) *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press.

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

Lewis, L. (1995). *Managing computer networks: A case-based reasoning approach*. Artech House Publishers. London.

Rombach, H. D. & Verlage, M. (1995). Directions in software process research. In: M. V. Zelkowitz (Ed.), Advances in Computers, Vol. 41, Academic Press, 1-61.

Vollrath, I., Wilke, W. & Bergmann, R. (1998). Case-based reasoning support for online catalog sales. *IEEE Internet Computing*, **2** (4), 47-54.

Wess, S. (1995). *Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik*. Dissertation Universität Kaiserslautern. Infix Verlag, DISKI 126.