The INRECA-II Methodology for Building and Maintaining CBR Applications

Ralph Bergmann[1], Sean Breen[2], Mehmet Göker[3],
Michel Manago[4], Jürgen Schumacher[1], Armin Stahl[1],
Emmanuelle Tartarin[4], Stefan Wess[5], & Wolfgang Wilke[1]

[1] University of Kaiserslautern
Centre for Learning Systems and Applications (LSA)
PO-Box 3049
D-67653 Kaiserslautern, Germany
{bergmann|jschuma|stahl|wilke}@informatik.uni-kl.de

[2] Interactive Multimedia Systems (IMS)
Clara House, Glenageary,
Co. Dublin, Ireland
sbreen@imsgrp.com

[3] Daimler Benz AG - Forschung und Technik
Evolutionäre Systeme (F3S/E)
Postfach 2360,
D-89013 Ulm
mehmet_goker@ep.mbag.sifi.daimlerbenz.com

[4] AcknoSoft
58 a, rue du Dessous des Berges
75013 Paris, France
manago@ibpc.fr

[5] TECINNO GmbH
Sauerwiesen 2
D-67661 Kaiserslautern, Germany
wess@tecmath.de

Abstract

This paper presents a brief overview of the INRECA-II methodology for building and maintaining CBR applications. It is based on the experience factory and the software process modeling approach from software engineering. CBR development and maintenance experience is documented using software process models and stored in a three-layered experience packet.

1. Introduction

Today, there are already a few companies which are specialized in developing CBR applications. Their problem is that they mostly develop their applications in an ad-hoc-manner: They do not have guidelines or methods which could help their developers in performing a new project and no ways to preserve experience made in performed projects for future use. This can cause serious problems when members of the staff leave, taking their experience with them, and new staff has to be trained. The result is an inefficient or ineffective system development, which cannot be sustained by contemporary organizations. From these problems, the need for a *methodology* to support the development and maintenance of CBR applications arson a few years ago and several approaches in that direction have been proposes (see Bergmann et al., 1997, for an overview). A methodology describes the development of a software system using a systematic and disciplined approach. It gives guidelines about the activities that need to be performed in order to successfully develop a certain kind of product, e.g. any kind of software system, as in our case, a CBR application. A methodology shall use a well-defined terminology, which makes it also possible to collect experiences made in past projects in a structured and reusable way to improve future projects. One of the main driving forces behind the development and the use of a methodology relates to the need for quality in both the products and processes of the development of computer-based systems. The use of an appropriate methodology will provide significant quantifiable benefits in terms of *productivity* (e.g. reduce the risk of wasted efforts), *quality* (e.g. inclusion of quality deliverables) and *communication* (a reference for both formal and informal communication between members of the development team and between the developer and the client) and will provide a solid base for *management decision making* (e.g. planning, resource allocation and monitoring.

This paper describes the INRECA-II[1] methodology approach which is based on two relatively new areas in software engineering (SE): *experience factory* (Basili, Caldiera, & Rombach, 1994) and *software process modeling* (Rombach & Verlage, 1995). We developed a methodology based on recent SE techniques which is enriched by up to date experience on building and maintaining CBR applications. This CBR experience was identified by analyzing several successful industrial applications developed by the industrial Partners of the INRECA-II[1] consortium.

2. The INRECA-II Methodology Approach

Our approach to a CBR development methodology is itself very “CBR-like“. In a nutshell, it captures previous experience from CBR development and stores it in a so-called experience packet (a term from the experience factory approach). The entities being stored in the experience packet are software process models, or

[1] Funding for this work has been provided by the Commission of the European Union (INRECA-II: Information and Knowledge Reengineering for Reasoning from Cases; Esprit contract no. 22196) to which the authors are greatly indebted. The partners of INRECA-II are AcknoSoft (prime contractor, France), Daimler Benz (Germany), TECINNO (Germany), Irish Multimedia Systems (Ireland), and the University of Kaiserslautern (Germany).

fragments of it such as processes, products, or methods. The experience packet 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.

2.1 Experience Factory

The experience factory idea is motivated by the observation that any successful business requires a combination of technical and managerial solutions which includes 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; a closed-loop process that supports learning and feedback. The key technologies for supporting theses requirements include: modeling, measurement, the reuse of processes, products and other forms of knowledge relevant to the (software) business. An experience factory is a logical and/or physical organization that supports project developments by analyzing and synthesizing all kinds of experience, acting as a repository for such experience, and supplying that experience to various projects on demand (see Figure 1). An experience factory packages experience by building informal, formal or schematized models and measures of various software processes, products and other forms of knowledge via people, documents and automated support. The main product of an experience

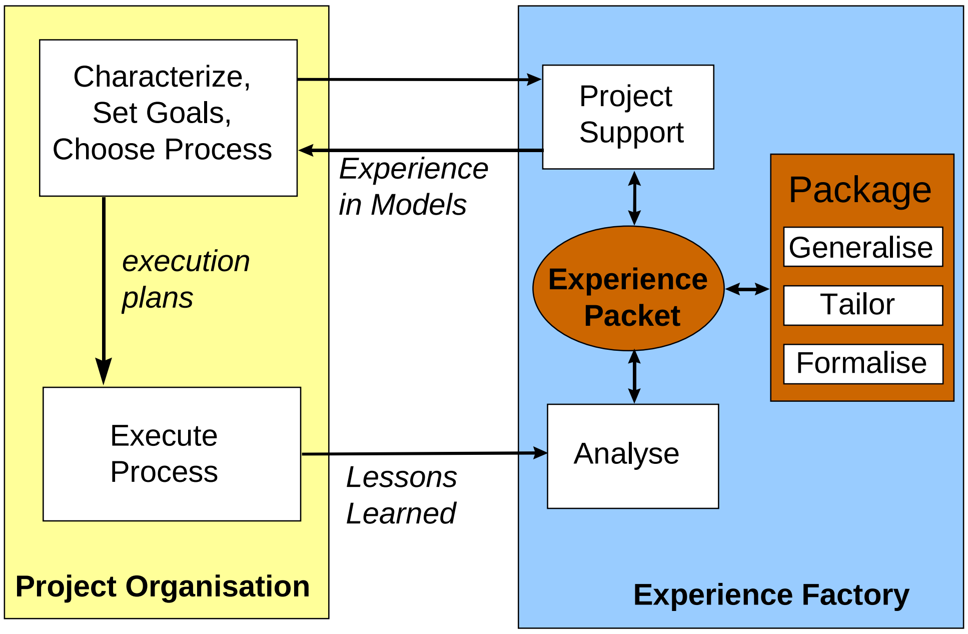


Fig. 1. The Experience Factory Approach (Basili, Caldiera, & Rombach, 1994)

factory is an *experience packet*. The content and the structure of an experience packet vary based upon the kind of experience clustered in the packet.

Althoff and Wilke (1997) discuss the relationship between CBR and the experience factory approach in some detail.

2.2 Software Process Models

Software process modeling is an approach that is highly important in the context of the experience factory approach. Software process models describe the engineering of a product, e.g., the software that has to be produced. Unlike early approaches in SE, the software development is not considered to follow a single fixed process model with a closed set of predefined steps. A tailored process model particularly suited for the current project must be developed in advance. Software process models include *technical SE processes* (like requirements engineering, design of the system to be built, coding, etc.), *managerial SE processes* (like management of product related documentation, project management, quality assurance, etc.), and *organizational processes* (covering those parts of the business process in which the software system will be embedded and that need to be changed in order to make best use of the new software system). From time to time, such a model must be refined or changed during the execution of the project if the real world software development process and the model do not match any longer.

Several representation formalisms for process models have been already developed. Although the particular names that are used vary from one representation to another, they all have a notation of *processes*, *methods*, *products*, *goals* and *resources*. A *process* is a single step that has to be carried out in a software development project. Each process has a defined goal and it consumes, produces or modifies certain products. Usually, the goal of a processes is to create or modify the *products*. Products include the executable software system as well as the documentation like design documents or user manuals. For enacting a process, there can be several alternative *methods* that describe how to actually enact the process. When the process is enacted, an appropriate method must be chosen. We distinguish between simple and complex methods. A *simple method* can be a textual description like a guideline of what has to be done to reach the goal of the process. A *complex method* decomposes a process into a set of sub-processes that exchange certain by-products in the course of achieving the goal of the main process. For a detailed description of the software process modeling approach used in the INRECA-II methodology see (Bergmann et al. 1997b).

In the INRECA-II methodology, software process models are used to represent the CBR development experience that is stored in the experience packet. Software processes being represented can be either very abstract, i.e., they can just represent some very coarse development steps such as: domain model definition, similarity measure definition, case acquisition. But they can also be very detailed and specific for a particular project, such as: analyze data from Analog Device Inc. operational amplifier (OpAmp) product database, select relevant OpAmp specification parameters, etc. The software process modeling approach allows to construct 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 packet.

2.3 The Structure of the Experience Packet

The experience packet 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 (see Figure 2).

Common Generic Descriptions. At this level, processes, products, and methods are collected that are common for a large spectrum of different CBR applications. These descriptions are the basic building blocks of the methodology. 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 of 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.

Cookbook-Level: Experience Modules. At this level, processes, products, and methods are tailored for a particular class of applications (e.g., help desk, technical maintenance, product catalog). For each application class, the cookbook-

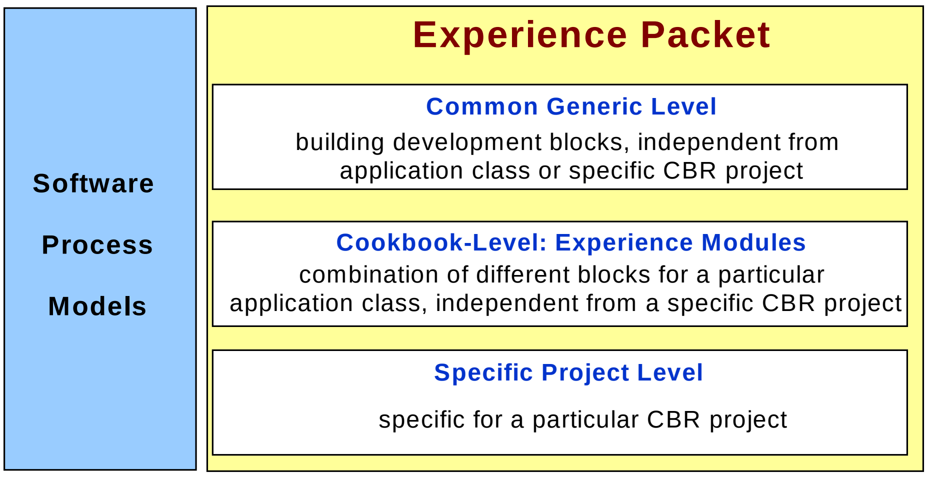

Software
Process
Models
Experience Packet
Common Generic Level
building development blocks, independent from application class or specific CBR project
Cookbook-Level: Experience Modules
combination of different blocks for a particular application class, independent from a specific CBR project
Specific Project Level
specific for a particular CBR project

Fig. 2. Structure of the Experience Packet

level contains an *experience module*. Such an experience module is a kind of recipe describing how an application of that kind should be developed and/or maintained. Thereby, the items (processes, methods, and products) contained in such a module provide specific guidance for the development of a CBR application of this application class. Usually, these items are more concrete versions of items described at the common level. Unlike processes at the common level, all processes which are relevant for an application class are connected and build a product flow from which a specific project plan can be developed.

Specific Project Level. The specific project level describes experience in the context of a single particular project that had already been carried out in the past. It contains project specific information such as the particular processes that were carried out, the effort that was spent for these processes, the products (e.g. domain model) that have been produced and methods that have been selected to actually perform the processes and the people that had been involved in executing the particular processes.

2.4 Documentation of the Experience Packet

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 from that is designed to document one of the items. It contains several predefined fields to be filled as well as links to other sheets (see example in the Appendix). We have developed four types of sheets (for products, processes, simple methods and complex methods) for documenting generic processes that occur on the top and the middle layer of the experience packet and 6 types of sheets (four sheets for products, processes, simple methods and complex methods, and two additional sheets for tool and agent descriptions) for documenting specific processes for the specific project level of the experience packet. Figure 3 shows the 4 generic description sheets. One kind of sheet is used to describe generic processes. Generic process sheets contain references to the respective input,

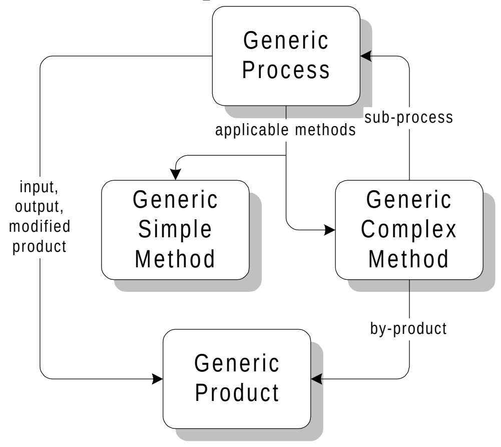


Fig. 3. Overview of generic description sheets

output, and modified products of the process. Each product is documented by a separate generic product description sheet. Each process description sheet also contains links to one or several generic methods. A generic method can either be a generic *simple* method (which is elementary and does not contain any references to other description sheets) or it can be a generic complex method. Such a generic complex method connects several sub-processes (each of which is again documented as a separate generic process description) which may exchange some by-products (documented as separate generic product descriptions).

As part of the INRECA-II project, a particular methodology tool was implemented which supports the management of the experience packet and the different modules it consists of. It supports the filling of the different sheets, checks consistency and creates the required links. It exports the experience packet as HTML network in which each sheet becomes a separate HTML page that includes links to the related pages. Therefore it is possible to investigate the experience packet via Intratnet/Internet using a standard Web browser.

2.5 Using and Maintaining the Experience Packet

When a new CBR project is being planned, the relevant experience from the experience packet must be selected and reused. The experience modules of the cookbook-level are particularly useful for building a new application that directly falls into one of the covered application classes. We consider the experience modules to be the most valuable knowledge of the methodology. Therefore, we suggest to start the “retrieval“[2] by investigating the cookbook-level and only using the common generic level as fall-back. Furthermore, it is important to maintain the experience packet, i.e., to make sure that new experience is entered if required. For using and maintaining the experience packet we propose the following procedure:

1. Identify whether the new application to be realized falls into an application class that is covered by an experience module of the cookbook. If this is the case then goto step 2a; else goto step 3.

2a. Analyze the generic processes, products and methods that are proposed for this application class.

2b. Select the most similar particular application from the specific project level related to this module and analyze the specific description sheets in the context of the current application.

2c. Develop a new project plan and workflow for the new application based on the information selected in steps 2a and 2b. Goto step 4.

[2] Up to now, this retrieval is not supported by a tool, but through an index schema. However, support for retrieval (e.g. a CBR approach) is considered important for the future.

3. Develop a new project plan and workflow for the new application by selecting and combining some of the generic processes, products and methods from the common generic level; make these descriptions more concrete and modify them if necessary.
4. Execute the project by enacting the project plan. Record the experience during the enactment of this project.
5. Decide whether the new project contains new valuable information that should be stored in the experience packet. If this is the case, goto step 6, else stop.
6. Document the project using the specific description sheets and enter them into the specific project level of the experience packet (supported by the methodology tool).
7. If possible, create a new experience module by generalizing the particular application (together with other similar applications) to an application class and generalize the specific descriptions into generic descriptions. Add the new to the current cookbook (supported by the methodology tool).
8. If new generic processes, methods, or products could be identified that are of a more general interest, i.e., relevant for more than the application class identified in step 7, then add them to the common generic level (supported by the methodology tool).

3. Current Status

Up to now, the INRECA-II project has achieved an initial experience packet (Bergmann et al. 1997c, Bergmann et al. 1997d) and a first revision of. Currently, the common generic level, consists of about 150 different sheets documenting technical, organizational and managerial aspects. These descriptions are very general and have been identified based on our general experience of how to build a CBR system and on specific experience on certain projects. Right now, this level is more of a tutorial style which is most useful for people without any or with only less experience in building CBR applications.

The cookbook level currently consists of about 150 different sheets from three modules:

Product Catalog Search Module. The basic action is a parametric search by a potential client, or a sales person in the presence of a client, in a product-base or catalogue (see example in the Appendix).

Complex Help Desk Module. The task is to introduce a CBR system for trouble shooting and diagnosis for some complex technical equipment, typically via a hot-line telephone service. This module was build based on the HOMER application currently under development for the Daimler-Benz Hotline.

Technical Maintenance Module. The task is to support the maintenance of technical equipment and is based on several related applications developed at AcknoSoft.

At the specific project level, the experience from four projects is captured and stored in about 350 sheets.

4. Future work

Future work in this area will focus on entering more experience into experience packet. It is important to include more experience modules for other kinds of applications and to extend the existing modules to cover the maintenance aspect in more detail. Additionally, we want to motivate the CBR community (researchers and application developers) to use our approach and to provide at least some of their experience into a public experience base. Another important aspect to be addressed in the future is the development of tools for efficiently accessing the experience packet, e.g. by applying CBR and information retrieval techniques. This will become particularly crucial when the experience packet grows further.

References

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.

Bergmann, R., Wilke, W., Althoff, K.-D., Breen, S., Johnston, R. (1997). Ingredients for Developing a Case-Based Reasoning Methodology. In: R. Bergmann & W. Wilke (eds.), *Proceedings of the 5th German Workshop in Case-Based Reasoning (GWCBR'97),* LSA-97-01E, University of Kaiserslautern, pp. 49-58.

Bergmann, R., Wilke, W., & Schumacher, J. (1997b). Using software process modeling for building a case-based reasoning methodology: Basic approach and case study. In: D. Leake & E. Plaza (eds.) Case-Based Reasoning Research and Development (ICCBR'97). Lecture Notes in AI. Springer, pp. 509-518.

Bergmann, R., Breen, S., Göker, M., Johnston, R., Schumacher, J., Stahl, A., Traphöner, R., Wilke, W. (1997c). Initial methodology for building and maintaining a CBR appliation. INRECA-Deliverable D2+3, Version 1.

Bergmann, R. Breen, S., Göker, M., Johnston, R., Schumacher, J., Traphöner, R., Wilke, W. (1997d). Cookbook for building and maintaining a CBR application. INRECA-Deliverable D4, Version 1.

Basili, Caldiera, & Rombach (1994). The Experience Factory. In J. Marciniak (Ed.) *Encyclopedia of Software Engineering - Vol 1*. New York: Wiley.

Rombach & Verlage (1995). Directions in Software Process Research. *Advances in Computers*, Vol. 41, Academic Press.

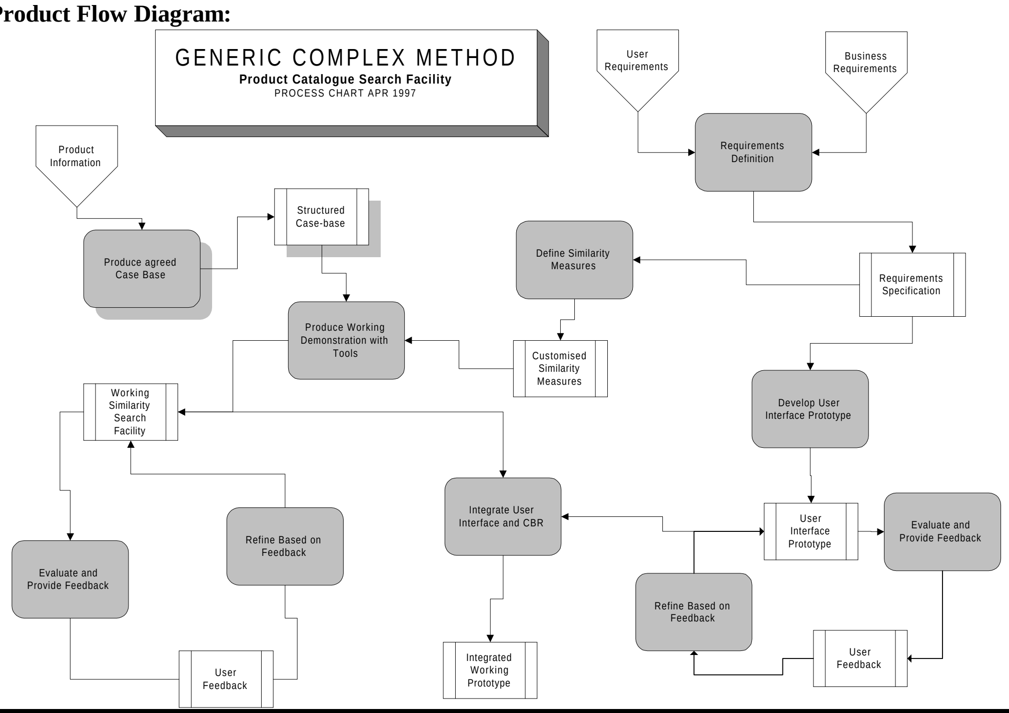
Appendix: Example Sheet[3]

Generic Complex Methods Description Sheet

Project: Product Catalogue Search	**Version:** 2.0	**Date:** 8/5/1997	**Context:**
Author: Sean Breen, IMS			

Method Name: Prototype Development of Product Catalog Search

Product Flow Diagram:



GENERIC COMPLEX METHOD
Product Catalogue Search Facility
PROCESS CHART APR 1997
User Requirements
Business Requirements
Requirements Definition
Product Information
Structured Case-base
Produce agreed Case Base
Define Similarity Measures
Requirements Specification
Produce Working Demonstration with Tools
Customised Similarity Measures
Working Similarity Search Facility
Develop User Interface Prototype
Integrate User Interface and CBR
User Interface Prototype
Evaluate and Provide Feedback
Refine Based on Feedback
Evaluate and Provide Feedback
Refine Based on Feedback
User Feedback
Integrated Working Prototype
User Feedback

Sub-processes

Name	**Generic Process**
Produce a Case-base for the Application	Produce Agreed Case base
Demonstrate the similarity Search using the CBR tools	Working Demonstration with tools
Define the user and business requirements	Define Requirements
Define and specify the similarity functions	Define Similarity Measures
Produce a working prototype of the User Interface	Develop User Interface Prototype
Evaluate a software product and provide feedback	Evaluate and Feedback
Refine the software product based on feedback from the users.	Refine Prototype based on Feedback
Integrate the User Interface with the tools based prototype	Integrate UI and CBR

By-Products

Name	**Generic Product**
Structured Case Base	Casebase
Customized Similarity Functions	Specification Document
Working Similarity Search facility using tools	Similarity Search Facility
Requirements specification for the system	Requirements Document
User Interface Prototype	UI Prototype
User Feedback	User Feedback

[3] This example sheet was produced by *Interactive Multimedia Systems (IMS), Dublin*.