Developing maintainable Case–Based Reasoning Systems: Applying SIAM to empolis orenge

Thomas R. Roth–Berghofer

German Research Center for Artificial Intelligence DFKI GmbH,
Knowledge Management Department,
Erwin–Schrödinger–Straße 57, 67655 Kaiserslautern, Germany
thomas.roth-berghofer@dfki.de

Abstract: Developing industrial Case–Based Reasoning (CBR) applications has become much easier since the advent of the INRECA methodology which employs software process modelling techniques to describe the development tasks, and which uses the experience factory approach to store the experience gained during the implementation of CBR projects. But the INRECA methodology does not describe how to maintain the developed systems in detail. This paper describes how to develop maintainable CBR systems by applying the six step CBR process model of the SIAM methodology to CBR applications using empolis orenge.

1 Introduction

Developing industrial Case–Based Reasoning (CBR) applications has become much easier since the advent of the INRECA methodology [BBG+99]. The INRECA methodology provides a *data analysis framework for developing CBR solutions for successful applications in real–world industrial contexts*. It employs software process modelling techniques to describe the development tasks [VR95], and it uses the experience factory approach [BCR94] to store the experience gained in the realization of CBR projects.

empolis developed many applications using the INRECA methodology, and those applications are running for several years now, making structured maintenance processes like the SIAM methodology [RB02] and the maintenance manual MAMA [RBR01] a necessity. But structured maintenance processes are not enough. The maintained system must be maintainable by design. empolis orenge [Sch02a] provides such a maintainable system core, not only for commercial use but also for academic research.

This paper is structured as follows: After introducing maintenance in the next section, and a look at related work, the six step process model and the broader context of the SIAM methodology is revisited in section 4. Section 5, then, relates empolis orenge to the six step process model and maps the terminology of empolis orenge to that of SIAM. The paper closes with some concluding remarks.

2 Maintenance

The control loop is the essential metaphor for the maintenance of any system (Figure 1). An ideal system becomes faulty because of defects. In software systems, the defects are not caused by parts wearing out but by the ever changing environment. And knowledge–based systems are especially sensitive to those changes.

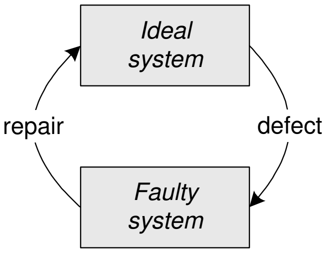

Figure 1: The control loop of system maintenance

Obviously, changes must be discovered before one can react to them. As soon as changes are recognized, especially those changes that have negative effects on the system, the faulty system can be repaired. It can be brought back to a desired state of functionality. Figure 2 shows just one possible sequence of CBR system states, where defects and repairs are following each other. As soon as the quality level of the system drops beyond some limit, repair operations are executed until the system reaches some satisfactory quality level.

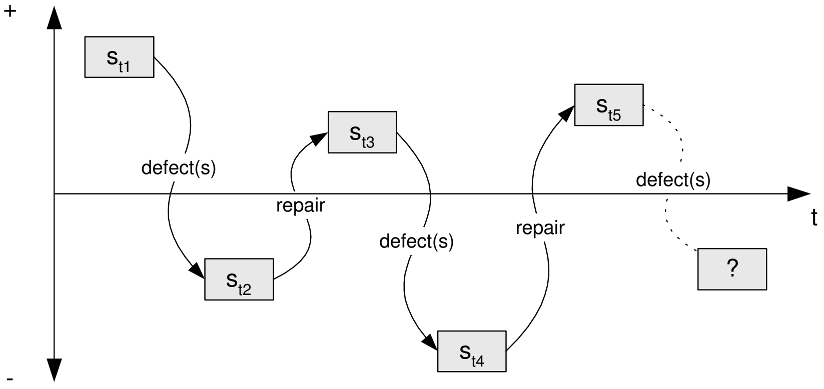

Figure 2: The changing quality level (+/−) of a system s over time t

The CBR system very well may stay unchanged during the time its quality drops, because the states are *system states in relation to the environment*. The repair operations change the CBR system (and, normally, not the environment). For example, in electronic commerce scenarios similarity measures often model user preferences (i.e., the more a customer prefers a product, the more similar is the product to the customer's query).

Over time, the customers may change their minds on the products (i.e., their preference relation, which is part of the environment of the CBR system, may change). Assumed, there are no maintenance actions in the meantime and the similarity model also stays the same. When those customers return to the CBR system and ask the same questions they will get the same answer. But this time, they may be unsatisfied with the results and reject the answers because they do not match their preference models anymore.

3 Related Work

Many single maintenance methods have been developed over the years, but a comprehensive process model was missing. A first comprehensive work was provided by Wilson [Wil01]. In his dissertation, he focuses on *the overall case base maintenance problem in CBR* and describes *new maintenance techniques within that paradigm.* He presented a framework for describing case base maintenance techniques and classified existing systems according to this schema. The theoretical work is supported by new methods and experiments. Wilson generalized the framework to cover all knowledge containers, and provided an example of similarity maintenance.

The framework is analytical in nature and does not directly support one in developing better maintainable CBR systems. It is based on existing CBR process models. It also does not directly help in developing maintenance policies for particular CBR applications.

Another field, where research is conducted, is the field of *experience base maintenance* [NA00a]. The experience base is a storage facility for *experience packages* in an organizational unit called *experience factory* [BCR94]. An experience base is also the basis of the INRECA methodology.

Experience packages are cases with a complex structure. Therefore, maintenance efforts focus on maintaining and improving the value of single experience packages and, following from that, maintaining and improving the value of the experience base. This is done based on the Goal–Question–Metric paradigm for goal–oriented software engineering measurement [BCR94]. The Corporate Information Network (COIN) which is the Fraunhofer IESE experience factory is used to validate the maintenance efforts [JAD$^+$01] and to develop guidelines for evaluation and improvement of experience bases [NF00, NA00b] as well as for support for acquiring new cases using already gained maintenance knowledge [NA01].

Maintaining an experience base, is maintenance of a class of CBR applications. Therefore, experience base maintenance, in principle, can be described using the SIAM methodology within a particular context.

4 SIAM — A Broader Perspective on Case–Based Reasoning

Looking at maintenance, in general, and having the control loop metaphor in mind, one discovers that the traditional CBR process models (i.e., the CBR flowchart according to Riesbeck and Bain [RB87], the CBR cycle of Kolodner [Kol93], and the four steps process model according to Aamodt and Plaza [AP94]) cannot sufficiently describe maintenance issues. Therefore, the most influential and widely acknowledged process model of Aamodt and Plaza was enhanced by the two additional steps Review and Restore [RIRB01]. This six step process model, then, was embedded into the SIAM methodology for knowledge maintenance of CBR systems.

4.1 The six step process model

The process model of Aamodt and Plaza [AP94] comprises the four steps *Retrieve*, *Reuse*, *Revise*, and *Retain*, with the first three steps grouped as *problem solving* phase and Retain as *learning* phase. The two phases are the basis for the six step process model with the two phases renamed to *application* phase and *maintenance* phase in the context of SIAM [RBI01, RIRB01].

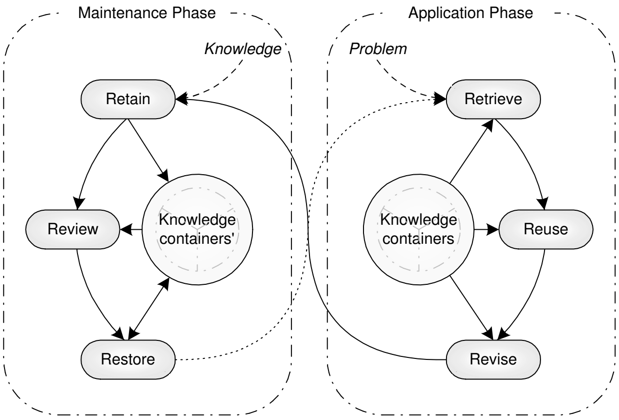

Figure 3: The six RE cycle (adapted from [RIRB01])

Figure 3 illustrates the six steps cycle. During the application phase, the knowledge of the system is not changed. But learning (i.e., retaining new cases) introduces change into the system. Of course, that is the intended behavior, but it is a behavior that must be responded to accordingly. And change is not only introduced by Retain, there is also a changing environment that demands continuous attention.

To collect information useful for maintenance, the existing steps were enhanced because each step provides opportunities to gather information as how often a case was retrieved, automatically adapted, or revised by users.

But the minor enhancement of the existing steps was not enough. There was no possibility to describe the necessary operations on how to assess the quality of the current system state, and to express necessary repair operations. Thus, the two new steps Review and Restore were added to the maintenance phase.

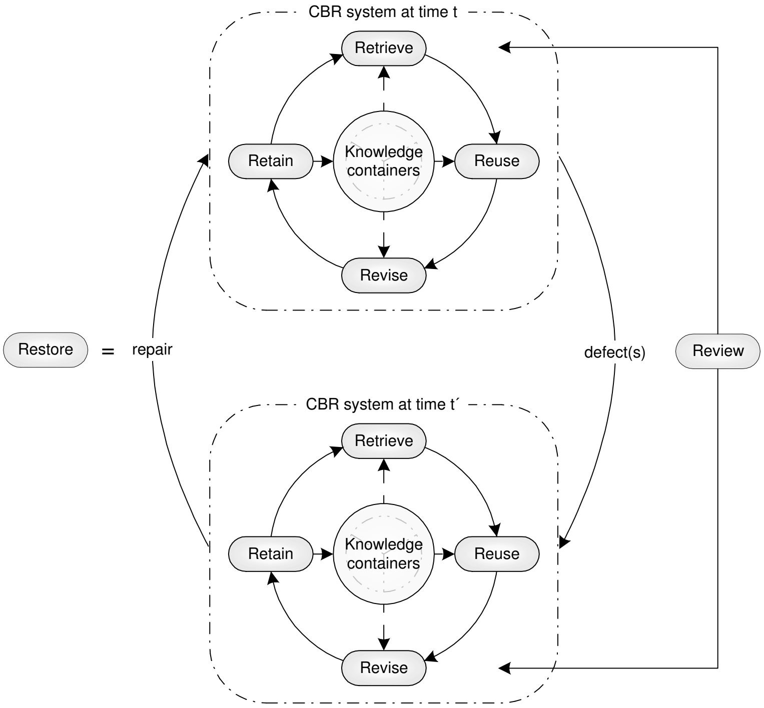

Figure 4: The control loop and the six step process model

The Review step considers the current state of a CBR system (cf. Figure 4, which shows just one possible flow of control in SIAM). It assesses its quality, and invokes Restore within the maintenance phase if necessary. The Restore step, then, changes the contents of the CBR system to bring it back to a desired level of quality. If there is no need to go to the Restore step, since the quality values are still in good shape, this step is simply skipped.

But the enhanced steps and the extended process model were only a first step. The six steps needed to be part of a broader context.

4.2 The SIAM methodology

The development of CBR systems and applications has to be part of every maintenance effort. The SIAM methodology provides this broader context, comprising *Setup*, *Initialization*, *Application* and *Maintenance* of a CBR project [RBR01]. SIAM drew concepts from two already existing methodologies: the INRECA methodology [BBG$^+$99] and the CRoss Industry Standard Process for Data Mining CRISP–DM [CCK$^+$00]. Both methodologies describe, how to develop software projects, but, of course, regarding two different fields.

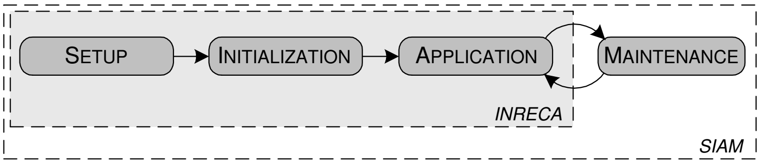

Figure 5: Coverage of INRECA and SIAM

Figure 5 shows the four phases of SIAM and the coverage in relation to INRECA. The phases Setup, Initialization, and Application are well covered by the recipes of the INRECA methodology. But SIAM covers more. SIAM enhances INRECA regarding the maintenance of CBR systems, including enhancements for building *better maintainable* CBR systems by also enhancing the Setup, Initialization, and Application phase [RB02].

Each of the phases is decomposed into tasks that are described on three levels of abstraction (see [RBR01] for examples): *generic*, *specific*, and *instance*. The specific level is identified with a maintenance manual that, in principle, describes all the necessary tasks to perform maintenance. It describes when to do what. The 'when' is decomposed into an event and a condition. An event is easy to recognize and can be a time running out, a new document coming in, or some user feedback. The event, then, triggers the evaluation of some (usually more complex or time consuming) condition such as a check for all outdated or incorrect cases of the case base. As soon as a condition is satisfied the action, described in the manual, is performed.

The generic level describes each task in an, of course, generic way. To make use of the generic task, the notion of the SIAM context was introduced that maps the generic to the specific level. The SIAM context specifies the *type of CBR system*, the *type of CBR application*, the *CBR tool* used, and the *most affected knowledge container*. These are regarded the most important dimensions for the description of the specific level.

The instance level further specializes the specific level. It is an instantiation of the specific maintenance policies. This level is identified with particular projects where parameters of the specific level are set to concrete values.

4.3 Operationalizing SIAM

Obviously, the SIAM methodology would be useless without operationalizing it, but the utilization of SIAM is already contained in the structure of the methodology. SIAM is operational by design. A major driving force for the development of SIAM always was to have an immediately applicable methodology. The construction of SIAM with its three levels of abstraction was a result of reusing practical experiences with methodologies, experiences both with developing and using the INRECA methodology in CBR projects as well as with developing and using CRISP–DM in data mining projects.

SIAM fully describes maintenance policies on a generic level. As soon as a maintenance manual is described on the specific level every project developed using the INRECA methodology and performed in that particular SIAM context can be maintained easily. Additionally, computer–based maintenance support can be provided by a management system that implements SIAM (an example system is described in [Max01]). But, in general, maintenance is organized and performed by humans rather than computers.

The INRECA methodology already is used at empolis to develop CBR applications and to constantly improve the quality of the CBR application development process. Consequently, the SIAM methodology (which enhances the INRECA methodology) is also being used, supporting the Total Quality Management efforts of empolis by maintaining the quality of running CBR applications.

5 empolis orenge — A maintainable CBR platform

empolis orenge [Sch02b] is a Structural and Textual Case–Based Reasoner.[1] It implements all four steps of the traditional Case–Based Reasoning cycle [AP94]. The orenge: Controller provides the steps Retrieve, Reuse, and Revise (of the problem solving or application phase) in form of the orenge:Services *Retrieval* (for Retrieve and Reuse), and *Adaptation* (for Revise). The *orenge: Controller* also implements the Retain step (the only step of the learning or maintenance phase of the traditional CBR cycle). This allows for *online* integration of cases. But most of the time, cases are integrated *offline* using the orenge: CaseBaseBuilder. The resulting new case base then replaces the older version in the productive system.

5.1 Six steps with empolis orenge

Additionally to the *orenge: Controller* with its fixed flow of control, a new configurable reasoner has been developed. Since empolis orenge Release 3.2, this reasoner, called *orenge: ProcessManager*, allows to build complex reasoning pipelines. The reasoning pipelines are a sequence of *pipelets*. Each pipelet provides one of the existing orenge:Services such as Retrieval and Adaptation. A uniform programming interface allows for an easy addition of customized pipelets. For instance, after retrieving cases similar to a query, the results could be checked if there are enough cases retrieved. A second retrieval (or as many retrievals as needed) can be started until enough cases are available. The cases can be adapted, altered algorithmically, or tested if they conform to any constraints. In an electronic commerce scenario, the stock could be checked if the retrieved products are available, before presenting the retrieval results to the customer. There are as many possibilities as can be programmed.

[1] A description of CBR approaches can be found in, e.g., [Len99] or [BBG$^+$99]

Furthermore, with the *orenge: ProcessManager*, empolis orenge is capable to implement appropriate support for the two steps Review and Restore. The enhancements of the original four steps, as described in [RBI01], could be implemented easily as pipelets, and according to the needs of the respective CBR project. If performance requirements do not allow for jointly collecting maintenance information and executing assessment tasks during problem solving, the queries could be sent to a second instance of the *orenge: ProcessManager* that is running in parallel. This reasoner could be reserved for collecting maintenance information such as performance measures or the amount of unknown concepts in queries, whereas the other instance is used for problem solving. The load balancing, in this scenario, would be handled by the *orenge: RequestBroker*.

Review and Restore are, then, covered by the specific level of the SIAM methodology.

5.2 Terminology

In 1995, Richter [Ric95] introduced the notion of the *knowledge containers* that contain and structure the knowledge of a case–based reasoner. A knowledge container is a collection of knowledge that is relevant to many tasks rather than to one. Prominent knowledge containers in rule–based systems, for instance, are *facts* and *rules*. Richter identified the following four knowledge containers of Case–Based Reasoning systems: The *vocabulary* (attributes, predicates etc.) comprises the domain model. The *similarity measures* are used to compare cases with queries. The *adaptation knowledge* accommodates past solutions to current problems. The *case base* stores the cases.

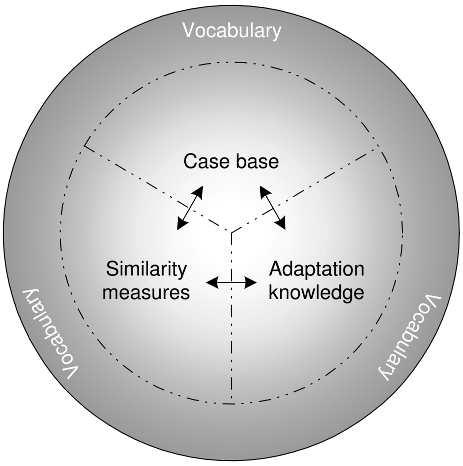

Figure 6: The four knowledge containers and their relation to each other

The knowledge for the first three containers is described and used during development of a CBR system (i.e., at *compile time*), while the knowledge in the cases is utilized only during actual problem solving (i.e., at *runtime*).

The four knowledge containers and their relation to each other are shown in Figure 6. The arrows depict that knowledge can be transferred from one knowledge container to the other. The vocabulary knowledge obviously is the foundation of all of the other three containers.

In empolis orenge, the knowledge containers could be identified quite easily. The knowledge containers are mapped onto empolis orenge's XML–based languages as shown in Table 1. The acronyms are explained in the following.

Table 1: Mapping of SIAM and empolis orenge notions

SIAM	**empolis orenge**
Vocabulary	Classes and concepts (OMML) Keys (OAML)
Similarity measures	Orderings, similarity tables, and taxonomies (OVML) Completion rules (ORML)
Adaptation knowledge	Adaptation rules (ORML)
Case base	OOML case base Index SQL database
Textual CBR	Service *orenge: Textminer* used during case base building
Structural CBR	Service *orenge: Textminer* not used during case base building

The vocabulary consists of *classes* and *concepts*. They comprise the *data model* and are defined using the *orenge model markup language* (OMML). As soon as the textmining[2] capabilities of empolis orenge are used, keys as synonyms to corresponding concepts must be defined. The keys are used to identify the concepts in queries given as free texts or during case base building. The keys comprise the *analysis model* and are defined using the *orenge analysis markup language* (OAML).

The case base exists in three flavors: as a list of cases, as an index, or as an SQL database.

- The list of cases is described in the *orenge object markup language* (OOML). This type of case base is used by the retrieval component *orenge: KnowledgeServer/Linear*. This retriever is the most capable one. There are no limitations on the similarity measures. Complex calculations are possible, but, then, the performance of this retriever can be a problem if the number of cases is too big.

- The index basically is a Case Retrieval Net (CRN) [Len99]. It is used by the *orenge: KnowledgeServer/Index*. This kind of retriever is limited regarding the similarity calculations to some degree (For more details, please, refer to the empolis orenge documentation). The index as well as the linear case base usually are created using the *orenge: CaseBaseBuilder*. It transforms text documents or structured data, such as data from a database or XML repository, into the appropriate representation, i.e.,

[2] The *orenge: Textminer* provides information extraction capabilities for Textual CBR.

into a CRN representation or into an OOML file (where the *orenge: CaseBaseBuilder* is an implementation of the Retain step). In the following the terms index and case base are used interchangeably.

- Whereas the first two kinds of case bases require a transformation step to get the cases, the original data of an SQL database is used directly. Here, CBR is performed on top of the relational database [SB00b, SB00a]. This kind of retrieval is used by the *orenge: KnowledgeServer/SQL.*

The similarity measures (for all of the three kinds of case bases) are described by the valuation model. It is defined using the *orenge valuation markup language* (OVML). The valuation model is extended with general knowledge provided by completion rules that are defined using the *orenge rule markup language* (ORML). Completion rules are used to modify the query, to infer additional information from that given by the user. They also could be used during the case base building process.[3]

empolis orenge uses adaptation rules for the execution of the Revise step. The adaptation rules also are defined using ORML. Usually, the completion rules and the adaptation rules are different, but in principle they could be used for both purposes, for the completion of queries and for adapting retrieved cases because cases and queries share the same structure.

6 Concluding remarks

This paper presented the SIAM methodology for knowledge maintenance of Case–Based Reasoning systems [RB02]. The paper shortly revisited the six step process model in the broader context of SIAM. It showed the relation between INRECA and SIAM, and applied the concepts of SIAM to empolis orenge.

The SIAM methodology is the basis for strong industrial Case–Based Reasoning applications at empolis, because it not only supports in developing such applications but also in maintaining them. empolis orenge is an advanced product coming from research and providing many of the features developed by the CBR community over the years. Its flexibility also makes it a good starting point for further research.

References

[AP94] Agnar Aamodt and Enric Plaza. Case–Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59, 1994.

[BBG+99] Ralph Bergmann, Sean Breen, Mehmet Göker, Michel Manago, and Stefan Wess. *Developing Industrial Case–Based Resoning Applications: The INRECA Methodology*. Lecture Notes in Artificial Intelligence, State–of–the–Art–Survey, LNAI 1612. Springer–Verlag, Berlin, 1999.

[3] Additional information about the completion of cases and queries using rules can be found in [Wes95].

[BCR94] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. The Experience Factory. In J. Marciniak, editor, *Encyclopedia of Software Engineering*, pages 469–476. Wiley, New York, 1994.

[CCK$^+$00] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rüdiger Wirth. *CRISP-DM 1.0: Step–by–Step Data Mining Guide*. CRISP-DM consortium: NCR Systems Engineering Copenhagen (USA and Denmark) DaimlerChrysler AG (Germany), SPSS Inc. (USA) and OHRA Verzekeringen en Bank Groep B.V (The Netherlands), 2000.

[JAD$^+$01] Andreas Jedlitschka, Klaus-Dieter Althoff, Björn Decker, Susanne Hartkopf, and Markus Nick. Corporate Information Network (COIN): The Fraunhofer IESE Experience Factory. In Rosina Weber and Christiane Gresse von Wangenheim, editors, *Proceedings of the Workshop Program at the Fourth International Conference on Case–Based Reasoning, ICCBR 2001, Vancouver, Canada*, pages 9–20, Washington, DC, 2001. Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Code 5510.

[Kol93] Janet Kolodner. *Case–Based Reasoning*. Morgan Kaufmann Publishers, Inc., 2929 Campus Drive, Suite 260, 1993.

[Len99] Mario Lenz. *Case Retrieval Nets as a Model for Building Flexible Information Systems*. Dissertation, Mathematisch–Naturwissenschaftliche Fakultät II der Humboldt–Universität zu Berlin, Humboldt University, Berlin, 1999.

[Max01] Rainer Maximini. Base system for Maintenance of a Case–Based Reasoning System. Diploma thesis, University of Kaiserslautern, 2001.

[NA00a] Markus Nick and Klaus-Dieter Althoff. The Challenge of Supporting Repository-Based Continuous Learning with Systematic Evaluation and Maintenance. *IESE Report No. 017.00/E*, 2000.

[NA00b] Markus Nick and Klaus-Dieter Althoff. Systematic Evaluation and Maintenance of Experience Bases. In Mirjam Minor, editor, *ECAI Workshop Notes – Flexible Strategies for Maintaining Knowledge Containers*, pages 14–21, Berlin, 2000. Humboldt University.

[NA01] Markus Nick and Klaus-Dieter Althoff. Acquiring and Using Maintenance Knowledge to Support Authoring for Experience Bases. In Rosina Weber and Christiane Gresse von Wangenheim, editors, *Proceedings of the Workshop Program at the Fourth International Conference on Case–Based Reasoning, ICCBR 2001, Vancouver, Canada*, pages 38–41, Washington, DC, 2001. Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Code 5510.

[NF00] Markus M. Nick and Raimund L. Feldmann. Guidelines for Evaluation and Improvement of Reuse and Experience Repository Systems Through Measurement Programs. In *Proceedings of the 3rd European Software Measurement Conference (FESMA-AEMES 2000)*, 2000.

[RB87] C. Riesbeck and W. Bain. A Methodology for Implementing Case–Based Reasoning Systems. Technical report, Lockheed, 1987.

[RB02] Thomas R. Roth-Berghofer. *Knowledge Maintenance of Case–Based Reasoning Systems — The SIAM Methodology*. Dissertation, University of Kaiserslautern, Kaiserslautern, Germany, 2002. Submitted.

[RBI01] Thomas Roth-Berghofer and Ioannis Iglezakis. Six Steps in Case–Based Reasoning: Towards a Maintenance Methodology for Case–Based Reasoning Systems. In Hans-Peter Schnurr, Steffen Staab, Rudi Studer, Gerd Stumme, and York Sure, editors, *Professionelles Wissensmanagement — Erfahrungen und Visionen (Includes Proceedings of the 9th German Workshop on Case–Based Reasoning, GWCBR 2001), Baden–Baden, Germany*, pages 198–208, Aachen, 2001. Shaker–Verlag.

[RBR01] Thomas Roth-Berghofer and Thomas Reinartz. MaMa: A Maintenance Manual for Case–Based Reasoning Systems. In David W. Aha and Ian Watson, editors, *Case–Based Reasoning Research and Development: Proceedings of the Fourth International Conference on Case–Based Reasoning, ICCBR 2001, Vancouver, Canada*, pages 452–466, Berlin, 2001. Springer–Verlag.

[Ric95] Michael M. Richter. The Knowledge Contained in Similarity Measures. Invited Talk at the First International Conference on Case–Based Reasoning, ICCBR'95, Sesimbra, Portugal, 1995. `http://wwwagr.informatik.uni-kl.de/~lsa/CBR/Richtericcbr95remarks.html` [Last access: 2002-10-18].

[RIRB01] Thomas Reinartz, Ioannis Iglezakis, and Thomas Roth-Berghofer. Review and Restore for Case Base Maintenance. *Computational Intelligence: Special Issue on Maintaining Case–Based Reasoning Systems*, 17(2):214–234, 2001.

[SB00a] Jürgen Schumacher and Ralph Bergmann. An Efficient Approach to Similarity–Based Retrieval on Top of Relational Databases. In Enrico Blanzieri and Luigi Portinale, editors, *Advances in Case–Based Reasoning, Proceedings of the 5th European Workshop on Case–Based Reasoning, EWCBR 2000, Trento, Italy*, pages 273–284, Berlin, 2000. Springer–Verlag.

[SB00b] Jürgen Schumacher and Ralph Bergmann. Similarity–Based Retrieval on Top of Relational Databases. In Mehmet H. Göker, editor, *Proceedings of the 8th German Workshop on Case–Based Reasoning, GWCBR 2000, Lämmerbuckel, Germany*, pages 75–86, Ulm, Germany, 2000. DaimlerChrysler, Research and Technology, FT3/KL.

[Sch02a] Jürgen Schumacher. empolis orenge — an Open Platform for Knowledge Management Applications. In Mirjam Minor and Steffen Staab, editors, *1st German Workshop on Experience Management: Sharing Experiences About the Sharing of Experience, Berlin, March 7-8, 2002, Proceedings*, pages 61–62. Gesellschaft für Informatik GI, 2002.

[Sch02b] Jürgen Schumacher. Whitepaper: empolis orenge — an Open Platform for Knowledge Management Applications, 2002. Available on request from orenge@empolis.com.

[VR95] Martin Verlage and H. Dieter Rombach. Directions in Software Process Research. *Advances in Computers*, 41:1–61, 1995.

[Wes95] Stefan Wess. *Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik*. Dissertation, Universität Kaiserslautern, Kaiserslautern, Germany, 1995. [In German].

[Wil01] David C. Wilson. *Case–Base Maintenance: The Husbandry of Experience*. Dissertation, Faculty of the University Graduate School in the Department of Computer Science Indiana University, 2001.