

**Workshop Proposal for the
Third International Conference on Case-Based Reasoning
ICCBR 99
July 27-30, 1999**

***Enhancing the Usage of Case-Based Reasoning
- Business Processes involved in applying CBR Technology in Industrial Organisations -***

Case-Based Reasoning (CBR) has long been considered as an AI technology with low maintenance efforts. However, with the advent of CBR systems in industrial applications, issues that have to do with the processes involved in putting a knowledge repository in operation arise.

Especially

- the processes involved in initial and continuous knowledge acquisition,
- the processes involved in case-base and model maintenance, and
- the organisational impact of, and impact of the organisation on a CBR system

have not been analysed and understood completely and are currently not supported adequately in all commercially available CBR systems.

This workshop will bring together CBR users, researchers and tool-vendors to discuss the problems they have, to evaluate possible solutions, to exchange experiences, and to come to a better understanding of unsolved problems in the area of maintenance.

Submissions to the Workshop

Position Papers of up to four pages on one of the topics given above can be submitted. Accepted papers will be published in workshop proceedings. Two papers from each category will be selected to be presented in 15 minutes during the four hour workshop. Each subtopic will have a dedicated slot of 60 minutes. The last hour of the workshop is dedicated to discussion. The workshop will close with a wrap-up session. Attendance is limited to 25 participants.

Submissions should be made to roth@tecinn.com in UNIX-printable Postscript format in Springer LNAI Style.

Program Committee

Mehmet Göker	DaimlerChrysler (Co-Chair)
Thomas Roth-Berghofer	tecInno (Co-Chair)
Brigitte Bartsch-Spörl	BSR-Consulting
Hans-Dieter Burkhard	Humboldt University Berlin
Michel Manago	Acknosoft

Dates

April 30, 1999	Submission deadline for workshop papers
May 28, 1999	Notification of Acceptance
June 22, 1999	Deadline for camera-ready copy of workshop paper
July 28, 1999	Workshop

Cases as Knowledge Assets

Brigitte Bartsch-Spoerl

BSR Consulting GmbH

Wirtstrasse 38

D-81539 Muenchen

Germany

Email: brigitte@bsr-consulting.de

Abstract. This paper tries to start a discussion about new ways and methods for calculating the potential return on investment (ROI) for case-based reasoning (CBR) projects and systems in use. It is motivated by the fact that as soon as CBR is regarded as an established technology, the return on investment question is one of the first serious hurdles that have to be taken. From my experience, prospective CBR users are not quick in either answering this question or in arguing why this question is not adequate – which means that this topic needs more attention, arguments and support for how to deal with it.

1 How is this Topic Related to Business Processes?

The topic of this paper does not correspond exactly to one of the three workshop categories but it has relations to all three of them.

The question “Will it pay off to build and use this CBR system?” should be answered at least roughly at the very beginning of any CBR project and it can (and should) be repeated after each prototype development cycle in order to incorporate new insights gained during the last prototyping cycle. The calculation of the return on investment (ROI) is – besides the system development and system integration costs – to a great deal influenced by the efforts for case knowledge acquisition, for the case model and case base maintenance and last not least by organisational impacts.

2 Why do we need New Ways of Looking at this Topic?

With the internet age, the classical methods for calculating assets are becoming somewhat old-fashioned.

“Systematic management of intellectual capital creates growth in shareholder value. This is accomplished, among other things, through the continuous recycling and creative utilisation of shared knowledge and experience.” – Leif Edvinsson, Skandia AFS [Bach et al. 99, page 25]

There are first examples for firms like Skandia that have started to integrate knowledge management activities into their daily business processes and even publish a knowledge balance-sheet with their annual report.

Österle [Bach et al. 99] states that “*knowledge creates shareholder value*” and examples for this hypothesis are e.g. the market quotations for firms like Yahoo or Amazon or eBay and a lot of other internet start-ups that went public.

All of these firms

- do not possess significant assets of the classical kind like ground, buildings, machines, materials etc. and
- do not earn money but produce rather big losses – at least during their early years.

The only thing these firms really “possess” are innovative ideas and business models that come together with sufficient knowledge, abilities and power for gaining a substantial market share within the new markets they have helped to create.

From these internet start-up cases I think we can see that classical ROI methods are not suitable for grasping or explaining what is happening here.

3 Why can't we just Borrow or Inherit Appropriate Approaches from Knowledge Management Projects ?

About one year ago, I thought that all these KM projects started in big consulting firms will eventually lead to accepted models for estimating the monetary value of reusable knowledge and that the CBR community should be able to borrow or inherit from such models.

But till today, there is – in spite of activities within accounting standards committees - no result of this kind visible and therefore I am beginning to doubt whether “wait and see and adapt what the KM community has agreed upon” will work for solving our problem – at least not on a short-term basis.

Maybe these KM projects are still in a stage where the upper management thinks that these projects are of strategic importance and this means “don't ask about the ROI yet” - but CBR has never been in such a position and therefore has to learn rather quickly how to deal successfully with ROI questions.

So this paper is a pleading for starting to think about and for developing approaches that show how the knowledge contained in case bases can be given a reasonable monetary value.

4 What can we Propose Regarding the Value of CBR systems?

4.1 The Classical Approach

The classical approach essentially says that the costs of an application can be determined in the following way:

- Calculate the efforts for building the CBR system and putting it into use
- Add the efforts for using and maintaining the CBR system for about 3 to 5 years

Its benefit can be determined in the following way:

- Calculate how much time and money is saved by re-using a suitable case retrieved from the case base
- Multiply this result by the number of expected case base usages and the probability to find a suitable case in the case base for the same period of time (about 3 to 5 years) as above.

This classical method is appropriate in situations where there are very many users around and not so very many cases so that every case has a rather high re-use rate – at least on the average.

This method is less appropriate in situations where e.g. the number of users is limited in the beginning and it is unclear how quickly the number of users and with them the re-use rate will grow.

4.2 A non-classical Approach

My problem with the classical approach is that the acquisition of a case makes money just fade away – which is not the case when I buy a computer or build a house. People argue that I can touch a computer or a house and that I can try to sell it if I don't want to use it any longer – and that this is not true for “virtual goods”.

This is exactly the point where I don't agree. My view is that nobody would invest money in documenting a solution if (s)he would not be rather sure that this investment will pay off – and therefore the investment just changes its form to a knowledge asset but does not fade away.

Knowledge assets have an actual value that can even surmount the initial effort for their acquisition, especially when their re-use rate is high and their contents is attractive for other firms using the same equipment and having the same sort of problems to solve.

I admit that we are far away from a case base market because most firms explicitly don't want to share their knowledge with their competitors. But not only competitors might be interested, also the developers or manufacturers of the faulty equipment could profit from the experience contained in the cases, user interfaces or manuals could be improved with this material etc. and all such improvements generate value from the contents of a case base.

Therefore I propose an alternative case evaluation model along the following lines:

- The initial value of an acquired case is equal to the effort invested in its acquisition.
- This value rises when the case is frequently re-used – the higher the re-use rate the higher the value of the case.
- This value goes down when the case is re-used very rarely or not at all.

This way of evaluating cases has the consequence that a case base administrator is responsible for deciding which cases are worth being documented with how much effort to invest. Furthermore (s)he should explicitly aim at gaining more users and improving the CBR application in order to preserve and rise the value of the case base.

For case bases that could be sold to other parties an additional component comes into play:

- The actual value of a case base consists of its internal value plus its market value – even if nobody thinks about sharing the case base because it represents a competitive advantage.

5 Discussion

In this paper I did not try to give exact numbers or percentages for the value of a case, its overall lifetime, increase in value or depreciation because such numbers depend on the complexity of the case, the lifetime of the equipment associated with the case, the technical ways to use it and many other decisions made mainly during the development phase. But all these parameters can be given a quantification for a specific application.

It is clear that all knowledge assets discussed in this paper are virtual and not always and at any time convertible into real money. But I think we should start with taking our knowledge assets serious and this will bring us in a better position when one day classical ROI approaches are able to deal with “virtual assets” like case knowledge.

Reference

[Bach et al. 99] Bach, V.; Vogler, P.; Österle, H.: Business Knowledge Management, Springer-Verlag Berlin 1999.

A Lifecycle Process for Experience Databases

Torgeir Dingsøyr¹

¹ Dept. of Computer and Information Science, Norwegian University of Science and Technology, 7491 Trondheim, Norway
dingsoyr@idi.ntnu.no

Abstract. A process model for developing, using and maintaining a database for reusing experience from software engineering is suggested. Special emphasis is placed on introducing the experience database in a company.

1 Introduction

Software engineering is a complex task where requirements from users are translated into a computer program. This involves tasks at different abstraction levels, from project planning and resource allocation, to algorithm design, testing and inspection of code.

A major problem is how to assure that the final program is of high quality. Various attempts have been made to improve the quality of software. Some of the attempts are formal methods, more reuse of software like in object-oriented technology, and also modelling the whole process of developing software [7].

The Norwegian project Software Process Improvement for better Quality (SPIQ) has as its goal to help the Norwegian software industry to improve the quality of software products [5]. One of the methods that has been studied is to improve learning in software developing organisations by reusing earlier experience. Experience can exist in the form of project plans that can be relevant, process models, checklists, quality manuals or tips on how common problems has been handled. A question is then how you collect and refine this set of experience. Within software engineering, a method called Experience Factory has been used at NASA [4].

The purpose of this paper is to describe the processes involved in introducing a system for reusing experience in a company. The technology which is used for developing this system might be other than a Case-based reasoning (CBR) [3] system. Here we suggest a set of processes a company should think of when introducing such a system. It is a model which is intended for discussion, and is not validated in other means than through discussions with participants from the industry within the SPIQ project. We will use the word *experience database* as a term describing a system which supports reuse of earlier experience in many forms.

2 A Process for Experience Reuse

To improve the quality in software development, the SPIQ project is developing a method which follows Demings Plan-Do-Check-Act cycle from Total Quality Management [6]. We advise companies to follow this double loop at strategic and at project level. That is:

- *plan* long term improvement goals
- set up several improvement projects (*do*)
- *check* the result of these projects
- perform changes to the whole organisation (*act*).

A change might be to decide to use object-oriented technology after evaluations in several project types. The improvement projects that are performed follow the same cycle in that they have to be:

- *planned*
- executed (*do*)
- the result should be *checked* against the plan
- results that are interesting for other parts of the organisation should be reported to other levels (*act*).

The experience gathered while implementing improvement efforts should be made available for future use. One way to achieve this is to develop an experience database. We will here discuss what companies should be aware of when creating such a database:

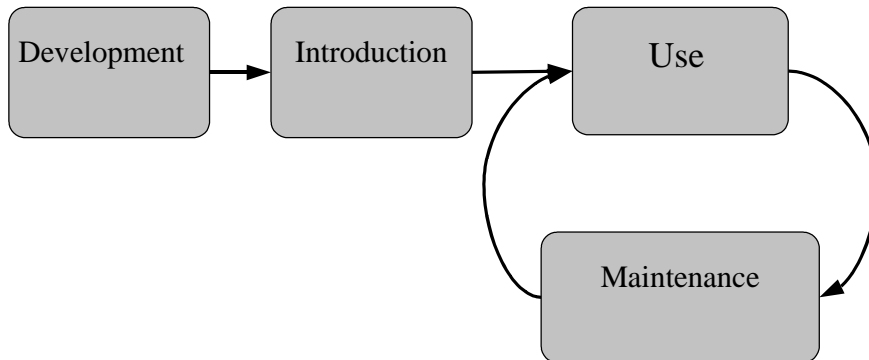


Fig. 1. Phases that a system for experience reuse goes through.

An experience database will have to be developed and introduced in an organisation, before it is actually used. It will also have to be maintained, as indicated in Fig. 1. From the SPIQ project, we know that the introduction can be particularly difficult [6].

Development

Before starting the actual development, it is important to know the intended use of the experience database. Who will use it and for what purpose? We list examples in the table below:

Content	Description
Guidelines	For improving software quality.
Reusable models	For development processes, quality, estimation and risk analysis.
Metrics	Definitions of product and project metrics related to quality.
Examples	
Scenarios	
Frequently asked questions	
Description of the organisation	

Introduction

Employees in companies are often enthusiastic when an experience database is introduced, but experience from the SPIQ project show that the databases are often not used as expected. To ensure that the database is used, some incentives might be:

- Demonstrate the usage of the tool: show the users how they can save time during development, and how it can give them a better overview of their own situation.
- Make the system dynamic: show that there are new information to investigate.
- Give feedback to users and market the database often.

Use

We here try to create an overview of different possible users in a software developing organisation:

User	Usage
Local developers	Retrieve and retain models.
Other developers	Learn from other's experience.
Project leaders	Plan, estimate and adapt projects.
Quality/process managers	Validate, improve and generalise contents.
Everyone	Improve by learning from others.

Maintenance

We can set up two kinds of maintenance of an experience database – technical development and maintenance of the information that is stored. The first category includes issues like converting old files from older to new versions of Word. The second is to ensure that information is kept up to date, for example by assigning a date-stamp on all information when it is considered not to be relevant anymore, and having an editor that can assure the quality of the information.

3 Related Work

The Experience Factory is a way to structure an organisation to support the reuse of experience which has been used at NASA [4]. This work has been continued and further specified in the Esprit PERFECT project [10]. Attempts have been made to combine work on software quality and Case-based reasoning [2], and also to use Case-based reasoning as a method to estimate the size of new projects based on earlier experience [1].

Within SPIQ, the telecom company Telenor 4tel has used an experience database for estimation [9]. Another company, ISI, is focusing more on organisational learning through an experience database [8].

4 Conclusion

We have described a lifecycle process for development and use of experience databases in software developing organisations, and have focused particularly on introduction of an experience database in an organisation as this has been a critical step for companies who have participated in the SPIQ project.

References

1. Aarts, R. J.: A CBR Architecture for Project Knowledge Management, In Smyth, B., Cunningham, P.: *Advances in Case-Based Reasoning : 4th European Workshop. Lecture Notes in Computer Science*, Vol. 1488. Springer-Verlag, Berlin Heidelberg New York (1998) 414-424
2. Althof, K.D., Birk, A., Wangenheim, C.G., Tautz, C.: CBR for Experimental Software Engineering. In M. Lentz, B. Bartsch-Spörl, H. D. Burkhard, S. Wess: *Case-Based Reasoning Technology - From Foundations to Application. Lecture Notes in Computer Science*, Vol. 1400. Springer-Verlag, Berlin Heidelberg New York (1998) 235-254
3. Aamodt, A., Plaza E: *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications* 1 (1994) 39-59

4. Basili, V., Caldiera, G., Rombach, H. D: The Experience Factory, *Encyclopedia of Software Engineering* (1994) 469-476
5. Conradi, R: SPIQ: A Revised Agenda for Software Process Support. In Montangero, C. (Ed.): *Software Process Technology - 5th European Workshop (EWSPT'96)*, *Lecture Notes in Computer Science*, Vol. 1149. Springer-Verlag, Berlin Heidelberg New York (1996) 35-41
6. Deming, W. E: *Quality, productivity, and competitive position*. Massachusetts Institute of Technology Center for Advanced Engineering Study, Cambridge, Mass (1982)
7. Glass, R. L.: The Realities of Software Technology Payoffs, *Communications of the ACM* 2 (1999) 74-79
8. Halvorsen, K., Nguyen, M.: A Successful software knowledge base. Panel introduction at to appear at *Software Engineering and Knowledge Engineering conference, SEKE'99*.
9. Jørgensen, M., Conradi, R., Sjøberg, D: Reuse of software development experience at Telenor Telecom Software. In *Proc. European Software Process Improvement Conference (EuroSPI'98)* Gothenburg, Sweden (1998)
10. PERFECT Consortium: *PIA Experience Factory, The PEF Model*, ESPRIT Project 9090, D-BL-PEF-2-PERFECT9090 (1996)

Case-Based Workflow

Thomas V. Fischer

Institut für Textil- und Verfahrenstechnik Denkendorf; Körschtalstr. 26, 73770 Denkendorf,
Germany
thomas.fischer-jr@itvd.uni-stuttgart.de

Abstract. A major part of the new product development process of a fabric manufacturer can be modeled as a workflow. A html-based workflow management system controls this workflow. It is based on a SQL database and an Active Server Pages application. The database serves as case base for a CBR system that supports the user in performing the task and in selecting and scheduling the next steps. Cases are retrieved with respect to content similarity based on the project data itself and structure similarity based on the workflow. The CBR system is under construction and will be realized through software agents that serve as small CBR systems with the same case base.

1 Introduction

Within the European research project VIRTEX¹ - Virtual organisation of the Textile and Clothing Supply Chain for co-operative innovation, quality and environment management, different integration methods and strategies for co-operative product development of functional textiles are explored. In order to integrate and support the product development processes of different partners in the supply chain, it is first necessary to understand the procedures within one company.

One of the industrial partners is a vertically integrated (spinning, weaving, dyeing and finishing) producer of fabrics for apparel, work wear and protective wear. The product development department (PD) handles all development and improvement projects, including customer or market needs and internal projects. Their main problems are first to organise and monitor these projects and second the knowledge and information recovery from the projects performed in the past. While the first problem could be solved with a workflow management system, the second is a typical CBR problem.

A business process analysis (BPA) showed different perspectives of innovation (e.g. completely new product, minor modifications) and different processes for these different perspectives, but the partial process for approval and cost calculation of the projects turned out to be identical for all projects. Hence, this process was chosen to be supported by a workflow management system that is able to act as a case base for a CBR system.

¹ VIRTEX, Brite EuRam 96 3470, www.itvd.uni-stuttgart.de/man/virtex

2 The Workflow

The process from a request for a new product to the approval to start with the trials is a typical cross process. It consists of up to 15 tasks to be performed by 5 different departments of the company. The tasks include the specification of the development request with respect to market (volume, price etc.) and product (type of product, technical specification), approval steps, technical feasibility studies, project cost calculations, and project planning (scheduling, deadline).

In compliance with the workflow reference model published by the workflow management coalition [1], these tasks are called steps. A step consists of three parts:

1. the task itself as described above (deadlines, costs)
2. the resulting event (e.g. ok, cancel the project)
3. the assignment of one or more next step(s), based on the resulting event

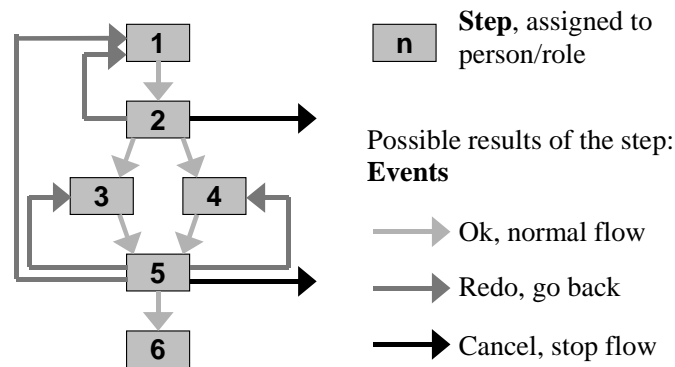


Fig. 1. A sample structure that contains all possible elements and clarifies the flow. At the end of each step, the user must select an event and one or more next steps, e.g. step five can be followed by step 6 (OK), by step 1 (Redo), or by step 3 and/or 4 (Redo). Note that due to the structure the selection of 1 and 3 after step 5 (Redo) is prohibited since they are followers.

A workflow management system (WFMS) provides automation of the business process by management of the sequence of steps². Such a system was developed within the research project. Because of the prototypical use within the research project, three main requirements in addition to the standard WFMS features arose:

- Flexibility: the system must be easy to install, modify and update.
- Internet: the system should be accessible via internet³ in order to allow remote access for sales agents. In addition, the same technology could be used for an inter-organizational WFMS for co-operative product development by more than one company.

² the exact definition in [1] is: A system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow.

³ see [2] for an overview of the possibilities of workflow and internet

- CBR: as already mentioned, the system should enable and support the design of case-based technologies.

3 The Workflow Management System

Since a market analysis of existing products [3] did not suggest affordable products able to cope with these requirements, we decided to develop a new tool. It consists in principle of two parts. All project data and all workflow management data is modelled as a relational database and stored at a SQL Server. The second part is an Active Server Pages (ASP) application running on an Internet Information Server (IIS) under Windows NT. ASP communicates with the database via Open Database Connectivity (ODBC). The ASP application produces HTML pages that are sent to the clients via TCP/IP. The user views the pages with a standard browser and sends data back to the ASP application. Here, the data is processed, controlled and sent to the database. In addition to the HTML pages presented to the client (which is in fact a pull technology, since the user has to open the browser and view the pages), ASP can send emails to the client using a standard Single Mail Transfer Protocol (SMTP) Server (push technology).

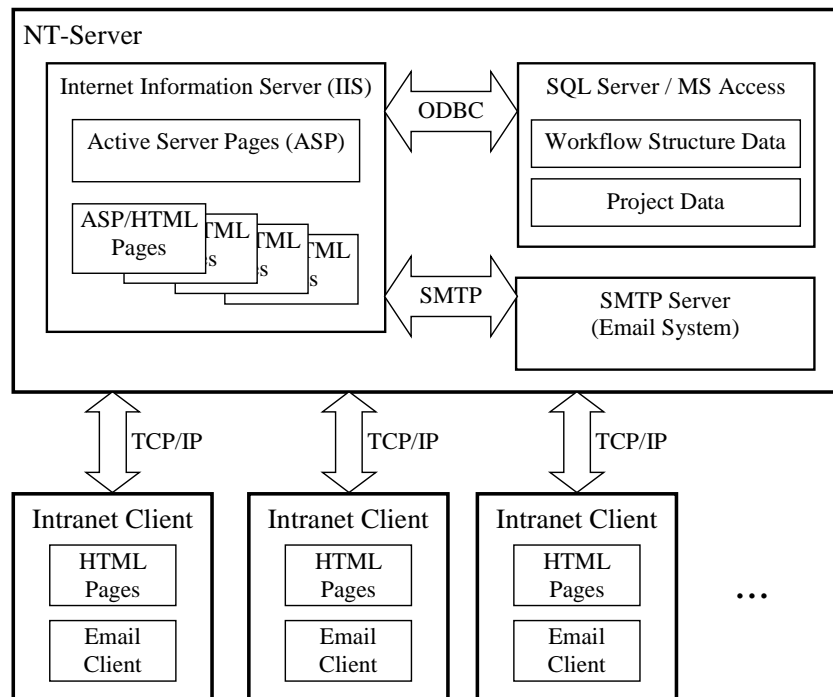


Fig. 2. The architecture of the developed workflow management system

The relational data model of the workflow engine makes the system very flexible. It is possible to design a new type of workflow by editing the tables containing the steps and the order of the steps (structure). All control- and feasibility functions are dynamic and do not need any update. The same holds for the management of users and their assignment to roles and functions within the workflow.

The project data database stores the complete history of all projects and allows tracking and monitoring of projects in progress and already finished projects. This is a big benefit compared to the old paper based process.

4 Case-Based Support

The described workflow management system fulfills two functions. First, it serves as a tool to control, distribute and monitor the request for new product development projects. Second, it stores all these projects in a database. This can be seen as a well defined business process to build up a case base for a CBR system. What are the problems such a CBR system can solve?

Possible tasks supported by CBR

Three main tasks have been detected:

- It can provide the user with an estimation of the success of the project (analytic task: classification, according to the classification given in [4] and [5]).
- It can support the task itself by reusing reports, calculations (analytic task: decision support).
- It can support the scheduling and estimation of the deadlines (synthetic task: planning).

The structure of the workflow remains the same within the given possibilities of branching and returning to previous steps. The CBR system only makes suggestions how to complete the tasks and navigate within the given structure. This is the main difference to the learning WFMS system Flexware⁴. There, the workflow is modeled only roughly a priori. The detailed structure is determined by reusing similar cases and their detailed workflow structure.

It is important to note that the CBR support is dynamic, that means the system can ideally support each user at each step of the workflow. This is a major difference to typical CBR systems that support the user only once. Especially the retain phase of the CBR cycle⁵ becomes more complex, since the proposals of the system cannot be evaluated before the whole project is finished. Hence, the system provides suggestions at different phases of the workflow, but the new case is added to the case base at the end of the flow when the project was finished and evaluated.

⁴ see [6] and [7]

⁵ the author refers to the CBR cycle first presented in [8] and its four phases: Retrieve, Reuse, Revise and Retain.

Similarity Concepts

The different tasks require different, appropriate similarity concepts. A basic distinction can be made: A case, which means a finished project, has two main classes of properties:

1. The project data itself: the initial specification consisting of technical and market information, the data inserted during the workflow: costs, deadlines, evaluations and finally the result of the project (rejected or approved). This project data based similarity is called content similarity.
2. The workflow data: the sequence of the steps performed, the resulting events (OK or back) and the throughput times of the different steps. Similarity based on workflow data is called structure similarity.

The arrangement between these two classes of similarity is a complex problem and depends heavily on the user preferences, i.e. what kind of support he wants to get. The tuning needs a lot of experience and lots of cases.

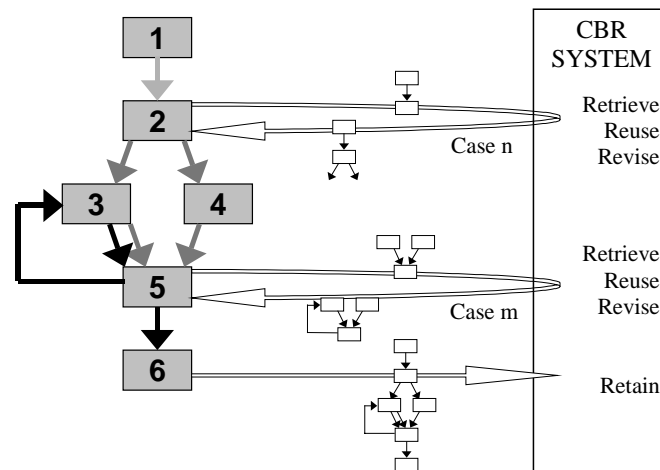


Fig. 3. An example explains the communication with the CBR system: At step 2, the system suggests to branch to step 3 and 4, based on the most similar case n (Retrieve, Reuse and Revise) to go back to step 3. After step 5, the system suggests to go back to step 3, this time based on the best matching case m. After step 6, when the workflow is finished and evaluated, the new case may be added to the case base (Retain or learning phase).

The content similarity is based on the project data. The values of the attributes are filled during the flow, starting with the project specification. In total, there are 40-50 attributes, but the ones consisting of free text (e.g. the remarks possible at the end of each step) cannot be used. Hence, there are 15 numerical values, about 7 based on possible value sets (lists), and 5 dates, which can be seen as numerical attributes since in fact only the difference of dates (number of days) is of interest. The similarity measure is based on a weighted partial match concept ⁶.

⁶ as proposed in [4]

At the moment, the concept for the structure similarity has not been finalized yet. Two possible approaches are evaluated. The first is based on a method for similarity measures for structured representations⁷. The basic idea is to calculate the distance between structure A and structure B based on the minimal transformation costs necessary to transform structure A into structure B.

The second approach is to introduce attributes that describe the decisions made at 'critical' situations (e.g. branching after step 2?). In addition with the throughput times (which are difficult to include in the first approach), these attributes will offer a protocol of the flow so far. Hence, again the concept of weighted partial match could be applied. While the first approach is formally correct, the second allows the use of additional knowledge (e.g. the definition of critical points) and is easier to implement.

Implementation

It is obvious that there is no need to support the workflow of every project at every step. In order to provide support only when necessary, it is planned to design software agents that will offer support only in critical phases of critical projects. These intelligent agents⁸ will offer support automatically. The challenge is that each of these case-based agents represents a small CBR system of its own which needs to be integrated. At the moment, the workflow system is in use and the CBR system is under construction.

References

1. Workflow Management Coalition: The Workflow Reference Model. Document Number TC00-1003, Issue 1.1 (1995)
2. Workflow Management Coalition: Workflow and Internet: Catalysts for Radical Change. WfMC White Paper (1998)
3. Weiss, M.: WFMS Systems. Internal report, ITV Denkendorf (1998)
4. Wess, S.: Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik. Infix Verlag (1996)
5. Althoff, K.-D., Bartsch-Spörl, B.: Decision support for case-based applications. In *Wirtschaftsinformatik* 38 (1996), 8-16
6. Aamodt, A., Plaza, E.: Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. In *AI Communications* 7 (1994) 1, 39-59
7. Wargitsch, C.: Ein Organizational-Memory-basierter Ansatz für ein lernendes Workflow-Management-System. FORWISS Report 1997-004 (1997)
8. Wargitsch, C., Wewers, T.: FLEXWARE: Fallorientiertes Konfigurieren von komplexen Workflows - Konzepte und Implementierung. In Müller, M. et. al. (Hrsg.): Beiträge zum 11. Workshop 'Planen und Konfigurieren' im Rahmen der XPS 97. Erlangen (1997), 45-55
9. Bunke, H., Messmer, B.T.: Similarity Measures for Structured Representations. In *EWCBR 93 Proceedings*, Springer (1994), 106-118
10. Müller, J. P.: *The Design of Intelligent Agents*. Springer (1996)
11. Brenner, W.; Zarnekow, R.; Wittig, H.: *Intelligent Software Agents*. Springer (1998)

⁷ as first presented in [9]

⁸ see [10] and [11] for a description of software agents

Using Scenarios to Envisage the Impact of CBR on Decision-Making Processes¹

A.D. Griffiths, M.D. Harrison and A.M. Dearden,

Human-Computer Interaction Group, Department of Computer Science,
University of York, York YO10 5DD, UK

email: `Tony.Griffiths` or `Michael.Harrison` @ `cs.york.ac.uk`

www: <http://www.cs.york.ac.uk/hci>

Abstract. This paper describes our experiences of using scenarios to design a ‘knowledge mediating’ CBR system. We show how we built *negotiated descriptions* of decisions made by the business, how these descriptions emerged through processes of *negotiation* and *refinement* and how we envisaged ways in which work processes would be transformed by our new system. We argue that envisaging the changing context of work is a vital step in the successful integration of decision technology.

1 Introduction

Successfully integrating decision support software into the decision-making processes of a business requires envisaging the ways in which those processes will be transformed by the new technology. This paper demonstrates this viewpoint by describing a *scenario-based* approach that we have used to envisage the impact of a prototype CBR system. Our approach involved the iterative refinement of scenarios, a method that is particularly suitable in domains where eliciting descriptions of work is difficult.

Our research concerns CBR as a technology for knowledge mediation. ‘Knowledge mediation’ includes the processes by which knowledge and information is transformed and translated as it is communicated between different expertise groups within an organisation. We are exploring the hypothesis that technological solutions based on CBR may alleviate the problems of communication that arise within an organisation through the different perspectives provided by the background, language and training of different groups of personnel. Our case study concerns knowledge mediation between the economic perspective of senior management and the perspective of engineering personnel at Northern Electric Distribution Ltd (NEDL). NEDL is an asset management company in the recently privatised Northern Electric group. Our problem is to support senior managers in making business planning decisions. The particular decisions we are investigating are concerned with maintenance policies and asset replacement, where decision-making must balance the requirements of parsimony and

¹ We acknowledge with gratitude the support of the UK Engineering and Physical Sciences Research Council (Grant GR/K84752) and Northern Electric Distribution Ltd.

the continual achievement of high standards of reliability. This is a knowledge mediation problem since decisions depend both upon an economic viewpoint (e.g. the priorities of different business drivers, the financial consequence of different plans) and an engineering viewpoint (e.g. the impact of the plans on reliability of service).

At present, time constraints and the availability of information mean that business planning decisions are made using only a few of the information gathering and decision-modelling activities that *could* be carried out. Specifically, we have observed that there are many rich sources of information in the business that are not exploited by business planners except through lengthy processes of interaction and negotiation with the engineering personnel. This is because the business planners do not generally have the familiarity with these systems needed to extract relevant information and they do not necessarily have the domain knowledge needed to interpret some of these sources of information, whereas these are both available to the engineers. The aim of the knowledge mediation tool is to represent this knowledge used by the engineers and provide, through the functions of a CBR system, the means for this knowledge to be reused within the community of practice of the business planners. This tool has been implemented in a prototype form and is currently being evaluated. A fuller description of the system can be found in [Griffiths et al., 1999].

The use of this technology would entail a number of desirable changes to the activity of business planning – relevant information would be available to business planners at reduced cost in time and resources, the scope of the analytical investigations supporting business planning decisions would be broadened and uncertainty about future outcomes for the business could be reduced through the provision of extra information. Thus, the task of designing the knowledge mediation system was also a task of envisaging transformations in business planning and the design process must evaluate not only the new computer technology but also the extent to which the technology delivers the desired changes in work practices.

2 A method of scenario capture and technology design

The process by which we envisaged how business planning would be transformed by our ‘knowledge mediation’ tool consisted of three steps, described below.

Describing Business Decisions

In order to understand the organisational role of the new system, it was first necessary to identify those classes of decision which the tool would support, and to describe those decisions from the point of view of the ‘business planner’ who would use the system. A central element of the approach described here is that the scope of the system was represented by a set of representative *decision scenarios*. This came about as follows. We were unable to observe the process of business planning directly for two specific reasons. Firstly, business

planning work is ‘temporally discontinuous’; it does not occur as single, contiguous activity on the part of any one person in the business. Instead, the work is distributed over time and between agents and for this reason becomes much harder to observe. Secondly, the personnel responsible for business planning are the most senior managers in the business. We found that our access to our primary informants was therefore limited by time pressures and by social barriers surrounding these personnel. Recently developed techniques for studying work such as contextual enquiry [Beyer & Holtzblatt, 1998], which depend on *observing* work in progress and ‘in context’, were unsuitable and instead we *constructed* descriptions of business planning work from a series of interviews. We found that our informants were happier to recall specific, historic decision-making episodes than to give a generalised, abstract account of ‘how decisions are made’ and therefore adopted a ‘scenario-based’ approach [Potts et al., 1994] [Carroll, 1995] [Febowitz & Greenspan, 1998] to describing work.

‘Scenarios’ are commonly described as *narratives*; they are stories that might be told about the organisation. A scenario describes a decision being made, or a work process being carried out, *in situ*. It describes not only the decision itself but also the organisational context in which the decision is made. An example of our initial set of scenarios is given in Appendix A, below. Each scenario describes a decision from the point of view of the business planner and also includes contextual information such as the ‘drivers’ causing the decision to be made, the alternatives considered, the criteria for evaluating the decision and the constraints under which the decision is made. Unlike some established notions of ‘scenario’, e.g. [Carroll, 1995], [Febowitz & Greenspan, 1998], our scenarios did not include the series of actions that were followed in resolving the decision. This is because the method followed here deliberately separates ‘what decision’ from ‘how’ the decision is made. Through this we focus on alternative means that will be provided by the new ‘knowledge mediation tool’ of addressing the business decisions represented by our set of scenarios.

Scenario descriptions were developed through a cycle of interviewing, writing down the scenarios and then reviewing the scenarios with our interviewees. The initial set of ‘historical’ decision scenarios was then augmented by a number of ‘invented’ scenarios. Invented scenarios represented *novel* decisions that might arise because of the changing context of the business or decisions that the business would like to engage with but are not addressed by existing processes. Clearly, where scenarios are invented, extra care must be taken to discuss and validate these with the intended users of the system and with other stakeholders in the development process. At the end of this step we had negotiated a set of descriptions of business planning decisions, which together served to establish the scope of the work to be supported by the knowledge mediation system.

Creating Work Scripts

The next step is to create *scripts* describing sequences of actions that would be sufficient for solving each of the business scenarios recorded in the previous step. An example of a ‘work script’ is given in Appendix B. These scripts played two

roles in our development process, one that is generic for the envisagement of decision support technology and one that is specific to our knowledge mediation problem.

Firstly, scripts record *existing* methods for decision-making; they are realistic descriptions of ‘how’ our decision scenarios might be addressed in the existing context of the business. Scripts therefore provide a baseline against which the transformations of decision-making afforded by the new system may be evaluated. This ability to evaluate the changes to decision-making provided by a proposed piece of technology is a vital part of our approach to integrating new decision technology into work processes.

Secondly, in the context of our knowledge mediation system, scripts match the business perspective of the scenario descriptions to the perspective of the engineering personnel who, in the existing work regime, are generally enlisted as analysts to provide much of the information needed by the business planning process. The scripts represent the activities of these engineers and contain the expertise which they might bring to bear to provide information required to resolve the business scenarios. The scripts therefore provide some basic translations between the points of view of the two communities of practice.

Scripts were reconstructed from recollections of previous decision-making episodes, by studying the artefacts produced and referenced by decision makers and by ‘role-playing’ various scenarios. As with the business scenarios, described above, these work scripts were validated and refined through a number of cycles of reflection and discussion.

Envisage the Proposed System

The third step was to author a series of *storyboards* envisaging how the process of business planning might be transformed by the new knowledge mediation tool. Each set of storyboards shows how one of the problems expressed as a business scenario might be solved by an interaction with the ‘envisaged’ tool. The storyboards were simply drawn out using a drawing package (we actually used Microsoft Powerpoint; an example is shown in Appendix C) and the process is one of imagining an application that would be sufficient to solve the problem being tackled.

These storyboards allow the transformation of the work of the decision maker and also the new decision support tool to be envisaged. On the one hand, these storyboards express alternative ‘methods’ for solving the problems expressed in the business scenarios. The storyboards therefore express in a concrete form the transformation of decision work that will be afforded by the new system, which allows these proposals for the transformation of work to be communicated to potential users and evaluated by all parties concerned. In the context of our research, they provide for the prior evaluation of our tool as a knowledge mediation artefact. It is possible, through the storyboards, to explore the knowledge and skills that are required of the user and in this way to consider whether the system successfully provides knowledge mediation. On the other hand, the storyboards also express emerging design ideas for the new system. Following the

arc of each narrative continually suggests features of the system, representations and operations that should be included in the user interface. In the case of our prototype system we found ourselves asking questions such as:

- What inputs are required to form an query to the case memory?
- How will retrieved cases be presented to the user of the knowledge mediation system?
- What facilities will be provided to the user of the knowledge mediation system in order to manipulate retrieved information?

Creating storyboards is an exploratory process and the storyboards depict a fluid system which is constantly evolving as each storyboard sequence is developed. Further iterations through the storyboards become necessary to collate the emergent ideas in a single design.

3 Discussion

The work we have described draws on existing concepts of ‘scenario-based design’. In this paper we have reflected on our experience of these approaches to design in the context of our research on ‘knowledge mediating’ CBR systems. The prototype knowledge mediation tool built by the project was based on a set of seven or eight of these scenario descriptions. The process described took place over a period of nine to twelve months, but this period was considerably lengthened by the exploratory nature of our work. However, we believe also that a lengthy period of investigation and design may often be necessary for the successful introduction of novel technology into the workplace. Scenarios have proved to be a powerful means of capturing ‘work as practised’ and envisaging the way in which that work may be transformed by decision support technology. This is a vital step in evaluating the transformations of work processes that will be brought about by new technology and in the successful integration of decision technology such CBR systems.

Scenarios have often been described as a useful tool for overcoming the implicit nature of workers’ knowledge about their own work. That is, it is known that many forms of work, even the ‘knowledge work’ carried out by engineers and business managers, draw upon tacit and implicit knowledge that is not easily articulated. The use of scenarios in elicitation is aimed to overcome some of these problems; their narrative form stimulates recall by harnessing our natural enthusiasm for story-telling and their informal nature provides a representation which can be shared by the user of the proposed system and by the computing professional responsible for its development. Our experiences supported these observations, but we were also aware that, even using scenarios as a vehicle for elicitation, certain kinds of information were easier to obtain than others. In particular, our informants were readily able to provide information about the *outcome* of decision-making and the rationale for those decisions, but details of *process*, such as the sequence of work, intermediate designs and hypotheses, and

the specific roles played by different participants, were more elusive. Difficulties in recall are presumably due to the way in which individuals ‘compile’ and process past memories. However, difficulties may also be due to the interaction between the individual and the ‘official’ corporate memory of the same events expressed in written reports describing the episode. McDonnell describes how these reports typically present a rhetorical argument in support of the chosen outcome rather than documenting the actual decision-making process [McDonnell, 1997], thereby obscuring those processes from memory.

In this context, we found that processes of *negotiation* and *refinement* were vital to making the approach work. We found that it was necessary and possible to go through a repeated process of ‘drafting’ scenario descriptions and reviewing them with the ‘customers’ of the design process until acceptable descriptions were agreed upon. These iterations of the interview process allowed our own misconceptions to be detected, gave opportunity for us to focus on areas where more information was needed and also stimulated our informants to extend or revise the descriptions as the results of earlier interviews were reflected to them. Our scenarios provided a means by which we, as designers, and the senior managers, as customers, could reach a *negotiated* view of the nature of their decisions and the processes that would be required to make them. Scenarios were valuable in providing a format which could be discussed and modified by all concerned, and helped us overcome the corporate nature of business decision making by incorporating the subjective accounts of the many different stakeholders into a single narrative. Through these processes of negotiation, we found that our scenarios were refined along two dimensions. Our scenarios and scripts were continually refined to increasing levels of *accuracy* as representations of the work processes of the host business. In addition, the progressive elaboration of the storyboards led to increasing levels of *commitment* to details of a design for our proposed ‘knowledge mediation system’.

In addition to the general benefits of the method, the use of scenarios in the development of a CBR system has an interesting corollary in that scenario-based design is essentially an instance-based approach. The scenarios elicited in the design of the system might in many contexts naturally become cases to incorporate into an initial case-base for the system. The method we are describing served as both requirements elicitation and knowledge acquisition for our case-based knowledge mediation tool.

References

- [Beyer & Holtzblatt, 1998] Beyer, H. & Holtzblatt, K. (1998). *Contextual Design. Defining Customer-Centred Systems*. Morgan Kaufmann.
- [Carroll, 1995] Carroll, J. M., editor (1995). *Scenario-Based Design: Envisioning Work and Technology in System Development*. John Wiley & Sons.
- [Febowitz & Greenspan, 1998] Febowitz, M. & Greenspan, S. (1998). Scenario-based analysis of COTS acquisition impacts. *Requirements Engineering*, 3(3/4):182–201.
- [Griffiths et al., 1999] Griffiths, A. D., Harrison, M. D., & Dearden, A. M. (1999). Case-based reasoning systems for knowledge mediation. To appear in *Proc. INTER-ACT99: Seventh IFIP conference on Human-Computer Interaction*.

- [McDonnell, 1997] McDonnell, J. T. (1997). Descriptive models for interpreting design. *Design Studies*, 18(4):457-473.
- [Potts et al., 1994] Potts, C., Takahashi, K., & Anton, A. (1994). Inquiry-based scenario analysis of system requirements. *IEEE Software*, March 1994: 21-32.

A Example Scenario

Scenario (A). Strategic Purchasing Decision - Insulating Oil

A new brand of insulating oil has appeared on the market which is more resistant to di-electric breakdown than existing brands and therefore offers cost savings to the business through reduced frequencies for the overhaul of oil-filled plant. However, the new oil is also substantially more expensive than existing brands. Do the benefits of decreased maintenance justify the increased expenditure, or should the business continue to purchase the current choice of insulating oil?

Comments: This scenario is driven not by changes in objectives or performance but by a proposal, made possible by awareness within the company of some technological change, and supported by someone within the organisation who believes in the proposal. The decision required is whether the expenditure is justified in terms of the objectives of the business. Aspects of decision include assessing the financial benefit delivered to the business and determining whether the improvements delivered are justified by a 'gap' between the present and intended performance of the business.

B Example Work Script

Work Script (A).1 : Oil purchasing decision procedure

A new brand of insulating oil has appeared on the market [...]

Script:

1. The asset data-base contains a representation of all scheduled maintenance actions relating to each piece of plant. Taking the forthcoming financial year 99/00 as a representative period of time, the data-base is queried for a list of all scheduled actions satisfying the following criteria;
 - The event is scheduled for financial year 99/00.
 - The maintenance action is directed towards an asset type which makes use of the relevant class of insulating oil.
 - The maintenance action is one which could be deferred if the new insulating oil is used; i.e. the overhaul is not ‘packaged’ with other kinds of maintenance which must themselves be carried out on the shorter time scale.
2. For each of these maintenance actions, access the related database tables to determine;
 - The cost to the business of each maintenance action.
 - The oil capacity of each related piece of plant.
3. Calculate the sum of the cost of this set of maintenance actions and the sum of the quantity of oil involved.
4. Calculate the projected cost of the affected parts of the maintenance schedule according to the following formula;

$$\begin{aligned} & (\text{Cost}(\textit{deferred}) + (\delta(\text{Cost}(\textit{oil})) \times \text{Oil_Quantity}(\textit{deferred}))) \\ & \times \frac{\text{Period}(\textit{current})}{\text{Period}(\textit{proposed})} \end{aligned}$$

Where *deferred* refers to the set of maintenance actions selected by the criteria stated in Step 1.

C Example Storyboard

Each set of storyboards is a 'slideshow' illustrating an interaction with the 'imagined' application.

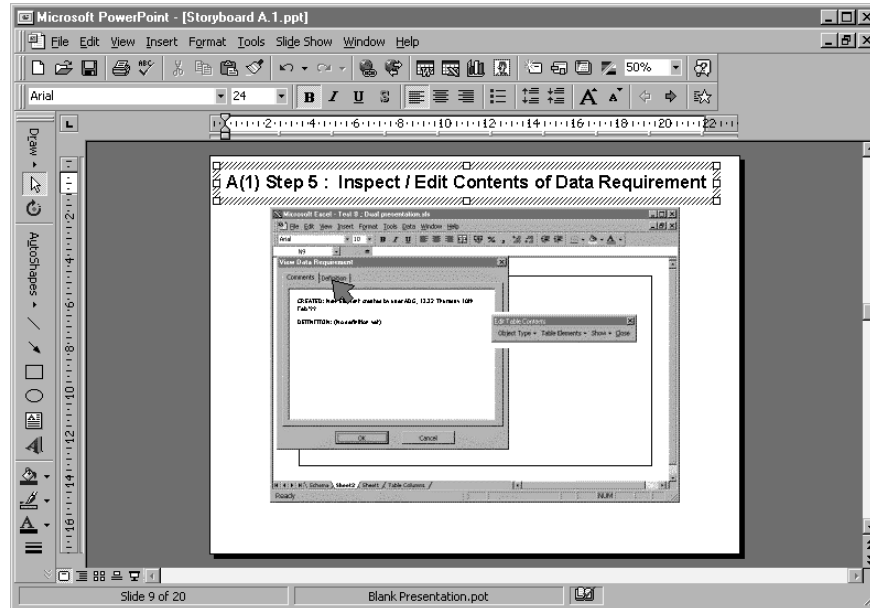


Fig. 1. Example Storyboard

Cases with a Life-Cycle

Mirjam Minor and Alexandre Hanft

AI Lab, Dept. of Computer Science, Humboldt University, D-10099 Berlin,
[minor,hanft]@informatik.hu-berlin.de

Abstract. In this article, a case-based approach for managing cases with a life-cycle is presented. Retrieval and maintenance processes are interlaced. As an application area, a case-based system accompanies and supports software engineers in their work of specifying test cases. The steadily growing case base requires a multi-layered similarity function and a sophisticated user guidance.

Keywords: Case-based reasoning, knowledge acquisition, maintenance.

1 Introduction

In several domains, cases are still evolving when they are stored and used in a case base. Different authors modify a case according to their personal knowledge. Sometimes new knowledge about a case emerges when the case-based system is already in use. Normally, such new experience is stored as a new case.

In the approach presented in this paper, the ripening of cases during the usage of the system is allowed and necessary. The application is situated in the domain of functional software testing. Under the same identification number, test cases pass through a life-cycle of several revisions from fragments to completely specified cases. So, the retrieval is embedded in numerous case authoring processes. Maintenance processes editing the background knowledge are frequently performed during the usage of the system.

A short overview of the application area and the case format is given in Section 2. How the frequent case authoring and maintenance processes are combined with a robust retrieval, is shown in Section 3. Section 4 describes, how to ensure a certain continuity of the retrieval processes by a multilayered similarity function. In Section 5, the concepts of the interaction with the users are presented, and Section 6 contains a short discussion of the required effort and the benefits of the approach.

2 Generating Test Cases With CBR

The project presented in this application resides in the area of generating test specifications for a software system. Producing software with an ISO 900x certificate requires several testing processes. In practice, the most important kind of test is functionally testing the implementation against the specification of

the software. Therefore, one has to develop a representative set of test cases demonstrating that the software works as specified.

Our case-based system supports the software developers who generate test specifications [5].

A case consists of an unique case number, textual descriptions and some additional attribute-value pairs. So, we have a case structure weakly reminding of database objects with several textual fields plus some scalar or multivalued attributes. The number of sections, their labels, and their content types are fixed, but their length is arbitrary. The set of attribute names is defined in the knowledge base, i.e. the number and names of attributes are not part of the fixed case format and can be changed in an easy way. Multivalued attributes can be specified.

Often, the contents of several cases overlap by some sentences or even sections. For example, it is described in different test cases how the system handles values of a variable higher than an upper limit, lower than a lower limit, and within the limits.

Example of a case:

```
<CASE_NUMBER>
122
</CASE_NUMBER>
<RETRIEVAL_ATTRIBUTES>
OBJECT = basic processing
CATEGORY = simulation
</RETRIEVAL_ATTRIBUTES>
<INFORMATION_ATTRIBUTES>
GENERATED_AT = 24.12.1998
GENERATED_BY = smith
</INFORMATION_ATTRIBUTES>
<DESCRIPTION>
The online data model manages measuring values accepting a
limit in the field "upper alarm limit".
</DESCRIPTION>
<INSTRUCTION>
In the test station, a value higher than the limit is set.
</INSTRUCTION>
<EXPECTED_RESULT>
The value is processed as described in chapter 5: entry stat.
limit ZA Go warning.
</EXPECTED_RESULT>
```

The system collects test cases in different degrees of ripeness in the case base. The cases pass through a *life-cycle*: Vague ideas as case fragments can be extended several times until they become complete test cases.

During the software development process, often a software engineer knows *what* will have to be tested later (DESCRIPTION) but has not yet specified *how*

to do this (INSTRUCTION). Then a fragmentaric case is produced with only some filled sections. In the process of writing the fragmentaric case, the user saves it several times. After this, the case may rest for several weeks until the author or another software engineer fills out some more parts, for example, when the concerned module has been finished and shall be tested systematically.

At the same time, the case base contains finished cases as well as fragmentaric cases that will be expanded. Permanently, maintenance processes are stimulated by development steps like the completion of a module. The case base is steadily growing.

Using a case-based system for test specification has the following benefits: Testing knowledge gained once in the development process can be conserved and accessed in the case base. This will shorten the actual testing phase after the implementation. Furthermore, a retrieval result can help avoiding duplicates or finding good formulations and scenarios for an actual edited case. Specifying test cases becomes more easy when good patterns from the actual or previous projects are available.

3 Interlacement of Retrieval and Case Authoring

The scenario described above necessitates that several persons can use the system for retrieval and maintenance all in a tumble. Writing new cases and editing incomplete cases is a crucial part of the usage of the system, because a showable set of test cases is one aim of testing. Instead of the usual term *retain*, we will use *case authoring* for describing the process of designing and editing test cases. The system supports case authoring processes as comfortably as retrieval processes.

We distinguish the case data used for retrieval processes from the highly up-to-date case data for case authoring processes. Between two complete updates, newly generated test cases are stored separately. New cases are only visible in the view of case authoring. When an already existing case is modified, a new revision is stored only accessible for the case authoring view, too. So, all recently edited cases are shown in two different views to retrieval and editing processes. Other revisions saved meanwhile are only stored in a history management which is not used yet. The unification of the two views happens always when an update is started explicitly or at regular intervals automatically.

Because of their high performance, retrieval processes can be queued easily. And by reason of the two different views on the case data, users asking a query are not bothered by maintenance processes performed by colleagues and vice versa. The only conflict can appear through a simultaneous write access to the same case. To solve such conflicts, time tags are asserted to cases when opening them for writing. Saving a case which has been changed by another person meanwhile produces an error message and asks the user for solving the conflict by either saving the case as a new case or overwriting the changes of the colleague. Should we see that a more sophisticated conflict management is necessary, a database or a revision control system could be embedded.

An example of a scenario is that a user edits several cases while other users want to have a look on some cases for performing the tests described within. The frequent changes of cases actually being edited are not mapped to the retrieval process for reasons of performance and continuity of the retrieval results. After finishing the phase of editing, possibly the case autor starts an update. Now, the newest revisions are visible in the retrieval results, too.

For avoiding confusion, the cases have persistent case numbers within their whole life-cycle. Author and time of the last update are shown when a case is browsed.

4 Three Layer Similarity

The purpose of our system to accompany a software engineer during all phases of specifying tests leads to a modification of the usual concept of similarity. In some publications, similarity has been replaced by usefulness [1]. Adopting this, useful cases for writing test cases are those with similar sections to a query. But both query and case, can be patchy and contain empty sections. The role of the empty sections disturbs the unity of similarity and usefulness (see below).

First, we will describe what kind of cases the system should find to what kind of a query. After this, we will explain how to encode this in a similarity function.

When a query has filled only some of the sections, completely specified cases shall be found as well as fragments of cases in an earlier development state. Otherwise, when a query has a lot of filled sections, also cases at the beginning of their life-cycle should be considered. So, a case can be very useful to a query even if a few sections are empty in one or in both.

For reaching these requirements, we chose a three layer similarity function. *Partial similarity* values between corresponding sections filled in both, the query and the case, are computed by means of *local similarity* values a priori gained from atomic similarities in the real world. As most of the sections of a case or a query contain textual descriptions, a method of textual case-based reasoning developed in the CBR-Answers project [3] has been applied for the partial similarity function. Broadly spoken, that means a representation of the texts by means of dictionaries of terms and comparison of the texts by means of dictionaries of synonyms and further local similarity relationships between terms. A *global similarity* function comparing a query with a case combines the partial similarity values:

$$\begin{aligned} sim(query, case) = & \frac{1}{\alpha} sim(S_{query}^1, S_{case}^1) + \\ & \frac{1}{\alpha} sim(S_{query}^2, S_{case}^2) + \\ & \dots + \\ & \frac{1}{\alpha} sim(S_{query}^k, S_{case}^k). \end{aligned}$$

If a section S_i is empty in the query or in the case, then $sim(S_{query}^i, S_{case}^i)$ is set to 0. For reaching similarity values in the range between 0 and 1, the

similarity function is scaled by $\frac{1}{\alpha}$ where α is the number of representative terms in the sections of the query that have a filled partner section in the case. The scaling can lead even to a lowering of the similarity value of a case when a query is extended and asked again. With respect to the overlaps of cases, this effect could be useful (see also Section 2).

This multilevel similarity function is quite robust to updates of parts of the cases. Compared with the same question, the similarity value of a case is not too erratic when the case is being expanded in several steps.

5 User Guidance

The graphical user interface guides the user through starting and evaluating a retrieval process, through the case authoring, through the maintenance of term dictionaries, and the maintenance of local similarity values. Those four general use cases of the system are layouted as four index cards. Skipping between the four cards is possible and supports the interlacement of different processes by one person. The single steps of the use cases as indivisible transactions each need a communication with the server. When writing a case, for example, a user may add a new term to the term dictionary. After switching to the maintenance card for the term dictionary and adding the term, the user can go back to the actual edited case and continue.

Implementing the graphical user interface as a Java applet with a connection to a CBR server, provides multi-user access and supports different development platforms of the users.

Basic term and local similarity dictionaries have been developed in advance. Before the users can fill the case base and expand the dictionaries, other sources have to be found for extracting phrases and terms. We took a computer dictionary from the internet as source as well as technical documents from the domain and test cases of other projects. Local similarity values are gained from a thesaurus.

All users of the system are allowed to do everything. Because of the system is running within a firewall, we don't need to respect any security aspects and to specify user profiles.

6 Conclusion

We have presented a case-based system accompanying and supporting software engineers in their work of specifying test cases. The system can collect and retrieve test cases in different degrees of ripeness, i.e. it manages frequent updates of cases within their life-cycles. Vague ideas for tests are edited several times until they become fully specified test cases.

In advance, the system provides a basic background knowledge in the form of dictionaries of terms and local similarity relationships. Applying the system to another software development project, it would consume a few hours to adapt the

basic dictionaries. During the specification of test cases, some additional effort is need to add actual terms and local similarity relations to the dictionaries. But it is also possible to insert a notice as a new case fastly on-the-fly and to enhance the dictionaries later. Meanwhile, the case is only mapped to the terms already known in a dictionary and may reach a lower similarity value to a query. So, because of the steadily growing case base, maintenance processes have to be performed simultaneously to all phases of using the system.

The effort of knowledge acquisition and maintenance of the system lies in between the high effort need to build an expert system and the one of writing test cases with a revision control or a database system. Available sources for gaining knowledge have been used to reduce manual knowledge acquisition. However, writing the test cases is much more time-consuming and has to be done in each case.

The system provides the possibility to distribute the work of designing and specifying test cases over the whole development process in an easy way. This influences the order of practical work and reduces the overall effort of testing. Whether the quality of the resulting test specifications is arising, can not be determined yet.

References

1. H.-D. Burkhard. Extending some Concepts of CBR – Foundations of Case Retrieval Nets. In *Case-Based Reasoning Technology — From Foundations to Applications* [2], pages 17–50.
2. M. Lenz, H.-D. Burkhard, B. Bartsch-Spörl, and S. Weiß. *Case-Based Reasoning Technology — From Foundations to Applications*. LNAI 1400. Springer Verlag, Berlin, 1998.
3. M. Lenz, A. Hübner, and M. Kunze. Textual CBR. In *Case-Based Reasoning Technology — From Foundations to Applications* [2].
4. E. Melis, editor. *7th German Workshop on CBR*, Saarbrücken, 1999. Universität Saarbrücken.
5. M. Minor. Managing Test Specifications with Case-Based Reasoning. In Melis [4], pages 132–139.