# A Survey of Case Studies of the Use of Knowledge Management in Software Engineering

Torgeir Dingsøyr* and Reidar Conradi**

*Sintef Telecom and Informatics
NO-7465 Trondheim, Norway
Torgeir.Dingsoyr@sintef.no

**Department of Computer and Information Science
Norwegian University of Science and Technology
NO-7491 Trondheim, Norway
Reidar.Conradi@idi.ntnu.no

e-mail of contact person: Torgeir.Dingsoyr@sintef.no

**Abstract.** This article examines the literature on case studies of knowledge management systems in use in organisations that develop software. We investigate knowledge management approaches in eight case studies, and what the reported benefits are. Surprisingly, very few organisations claim to have lowered software production costs or increased the quality of the software. But many claim to have improved the work situation for software developers and managers.

## List of Figures and Tables:

# 1. Introduction

This article is a survey of case studies of knowledge management systems in use in companies that develop computer software. We find many descriptions of such knowledge management systems in the research literature, but most of them deal with technical issues, and few are dealing with how these systems actually work in the organisations where they are deployed. This is an attempt to systematically present published case studies of knowledge management systems that can be found in the research literature, and to analyse 1) What systems are in use, and 2) What is impact of such systems on work in an organisation?

We have written this article for people who are either skilled in knowledge management, and are eager to know how this is interpreted and used in software engineering, or for people in the software engineering-field, who are interested in knowing more about what knowledge management has had to offer them. This article is partially based on [1].

First, we will briefly motivate the use of knowledge management systems in software development by discussing the use of software, common problems in development and suggested improvement actions. We then go on to define what we mean by “knowledge” and “knowledge management”, before we state more precise research questions for this survey. Next, we present different technology innovations for knowledge management in software engineering as context, and then present and discuss eight case studies found in the literature.

## 1.1 Software Development; Problems and Remedies

To develop and maintain software is often referred to as ”software engineering”. One definition is that software engineering ”is concerned with theories, methods and tools which are needed to develop software ... for computers”, and it differs from engineering in other disciplines because it is ”not constrained by materials governed by physical laws or by manufacturing processes” [2] (quoted in [3]).

### *1.1.1 Problems with software development*

Software development can often be challenging. There are many examples of software projects that have failed. The much-cited Standish report on software projects [4] “shows a staggering 31.1% of projects will be cancelled before they ever get completed. Further results indicate

52.7% of projects will cost 189% of their original estimates. The cost of these failures and overruns are just the tip of the proverbial iceberg. The lost opportunity costs are not measurable, but could easily be in the trillions of dollars...” The view that the software systems we use today are not very mature is also supported by the American ”President’s Information Technology Advisory Committee”, that writes: “The Nations needs robust systems, but the software our systems depend on is often fragile. Software fragility is its tendency not to work properly – or at all. Fragility is manifested as unreliability, lack of security, performance lapses, errors and difficulty in upgrading” [5].

So why does there seem to be so many problems related to software development projects? Software is an immaterial product, and it can be difficult to get an overview of a total program system, which can be millions of lines of code, to identify all possible error sources. Also, a very small defect might have a lot of influence in safety-critical systems, like the European Space Agency’s Ariane 5 satellite launcher, that ended in a failure in 1996. About 40 seconds after initiation, the launcher “veered off its flight path, broke up and exploded” according to the report by the inquiry board [6]. The error was “caused by an internal variable related to the horizontal velocity of the launcher exceeding a limit which existed in the software”. Thus, just a few lines of code that was lacking, had severe consequences – a loss of around 500 million pounds.

Other problems can be that the communication between the end-users and the software developers is lacking, or that project management is difficult in an environment where a small bug can take a very long time to correct, and where it is often difficult to determine how much work is left to do on a software module.

Numerous examples of problems in software development projects can be found in popular books like *Crash – Learning for the World’s worst Computer Disasters* [7] and *Software Runaways* [8].

After listing all these problems that exist in software, you may ask: are all software systems that bad? That is not so, there are a lot of software projects that deliver software that is highly usable and working. Robert Glass has argued that the software failures are the exception rather than the trend [9] – “we tend to focus on the unusual things that go wrong because they’re more interesting or important than the run-of-the-mill things that go right”. We should not use a word like “crisis” to describe the software development field when we know of so many well-working

systems. The main reason for this argument is that problems in software is used to motivate a lot of research; which should be able to stand on it's own feet.

We acknowledge that there have been more writings about the failures than the successes in software engineering projects, and that the situation might not be as bad as it looks. But as the reports we have cited earlier shows, there are at least quite a lot of projects that could improve, although it is not right to use a word like "crisis".

### *1.1.2 Suggested Remedies*

There has been a lot of discussion in the software engineering community about finding a "silver bullet" to end the problems, or at least reduce the impact of them. Several solutions have been tried to improve the way software is developed, like changes in the way software is produced, the "process", introduction of new programming languages, and supporting tools to assist in development. The goal is usually to increase productivity and quality of the developed software. The outcome of several of these improvement initiatives was summed up in an article in Communications of the ACM [10]. Claims of "order of magnitude" improvements were evaluated, on different "technologies", such as:

- Structured techniques – using structured analysis, design and programming.
- Fourth generation programming languages (4GL).
- Computer Aided Software Engineering – tools to support software engineering, mainly in analysis and design.
- Formal methods – formal specification and verification of software.
- Cleanroom methodologies – a method for removing defects from software.
- Process models – descriptions of appropriate processes in software engineering.
- Object-oriented technology – to find "objects" in the problem to be solved, and use those in generating software solutions.

Many of the technologies show promising results, but there are relatively few scientific articles that evaluate how the different methods work. Also, in some studies that claim improvement, the improvement technology is confused with other changes, like changes in the programming language. So there is still a need for more research on how these technologies really work.

## 1.2 What is Knowledge Management?

Recently, much focus has been placed on "managing knowledge" better in what we can call knowledge-intensive companies. This has been applied in many other domains than software

development, but we focus mainly on what has been achieved there, although we draw on general knowledge management theory to discuss what has happened in this domain. But first, we discuss what we mean by “knowledge” before going on by discussing “knowledge management”.

### *1.2.1 What is knowledge?*

The term “knowledge” is defined in the Oxford Dictionary and Thesaurus [11] as: “awareness or familiarity gained by experience (of a person, fact, or thing)”, “persons range of information”, “specific information; facts or intelligence about something”, or “a theoretical or practical understanding of a subject”. A more philosophical (and positivist) view of knowledge is to see it as “justified true belief”. We often divide knowledge into two types, *tacit* and *explicit* knowledge [12]. By tacit knowledge we mean knowledge that a human is not able to express explicitly, but is guiding the behaviour of the human. For example how to ride a bike is something that is difficult to express, which you have to learn by trial and failure. Another example of tacit knowledge is the struggle of Japanese engineers to make a machine that bakes bread. According to Nonaka and Takeuchi [13], there were several trials to construct such a machine, but the bread simply did not taste as well as bread made by human bakers. The company NEC decided to send people to a local baker to see how the process of making bread was carried out. The researchers returned with new insight on the kneading process, and later were able to replicate this in their machine. This is an example of tacit knowledge that is difficult to transfer by other means than looking at someone who are actually baking bread.

*Explicit knowledge* is knowledge that we can represent, for example in reports, books, talks, or other formal or informal communication. So when we later talk about computer systems for knowledge management, it is only the explicit knowledge that can be managed in these kinds of systems; the *tacit* knowledge remains in the people! Some claim that tacit knowledge can be converted to explicit through *externalisation* [13], and from explicit to tacit through *internalisation.* We also find conversions from tacit to tacit – *socialisation*, and explicit to explicit – *combination*.

Some terms related to knowledge, are *experience* and *information*. In normal English, experience means “actual observation of or practical acquaintance with facts or events”, or ”knowledge or skill resulting from this” [11]. Most people see experience as a type of knowledge that you have gained from practise. Information is seen as ”something told; knowledge”, ”items of knowledge; news”. In normal English, it is difficult to distinguish the terms information and knowledge. Within artificial intelligence, information is often referred to as ”data with meaning”. The characters ”4m” does not say much in itself, but if we know that ”m” stands for ”meters”, it can

be useful information. Knowledge is then often defined as information that is used (in an artificial intelligence-sense: in a computer system). For an interesting discussion about the terms data, information and knowledge in artificial intelligence, see [14].
This use of the term knowledge in artificial intelligence is however greatly disputed by Dreyfus [15], who claims that knowledge requires other processes than those in a computer system.
To sum up this discussion, it is clearly out of scope to land the discussion on knowledge in this article, but we will use a pragmatic definition of knowledge, what Taylor [16] who has been working with "information use environments" would call "instrumental information" – information that is used so that individuals know how to do something, or "factual information" – information that is used to determine facts. We will refer to this type of "operational information" as explicit knowledge, and we will also use the term tacit knowledge.

### *1.2.2 What is Knowledge Management?*

There are many interpretations of what knowledge management is, and many terms that describe computer systems to support managing knowledge in companies. In 1974, the book "The Corporate Memory" was published [17], arguing on the benefit of collecting information from different sources in a company and making it "searchable". At this time, the information was gathered on paper, and "search" would mean to submit a form to a department who would manually search through their files. The term corporate memory is still in use, but now meaning a computerised database for storing documents from many people in a company. The term "corporate brain" is also used to describe such a database. Another related term is "organisational memory", which does not really have a clear definition, but "intuitively, organisations should be able to retrieve traces of their past activities, but the form of this memory is unclear in research literature. Early efforts assume one could consider memory as though it were a single, monolithic repository of some sort for the entire organisation" [18]. Many see this term as meaning both a process of collecting and using information as well as a repository.
In Software Engineering, to reuse life cycle experience, processes and products for software development is often referred to as having an "Experience Factory" [19]. In this framework, experience is collected from software development projects, and are packaged and stored in an *experience base*. By packing, we mean generalising, tailoring and formalising experience so that it is easy to reuse. This will be further elaborated in the next subsection.
So what do we mean by knowledge management? We think that this term includes issues from all the terms discussed. Some goals of knowledge management can be [20]: "To make the enterprise act as intelligently as possible to secure its viability and overall success". Thomas Davenport has defined it as "a method that simplifies the process of sharing, distributing,

creating, capturing and understanding of a company's knowledge" [21]. If we look a bit more into knowledge management, we find that some important aspects are [22]:

- Survey, develop, maintain and secure the intellectual and knowledge resources of the enterprise.
- Determine the knowledge and expertise required to perform work tasks, organise it, make the requisite knowledge available, "package it", and distribute it to the relevant points of action.
- Provide (...) knowledge architecture so that the enterprise's facilities, procedures, guidelines, standards, examples, and practices facilitate and support active Knowledge management as part of the organisation's practices and culture.

This seems to be pretty in line with what people from two software companies see as knowledge management. We interviewed 13 managers and developers about what they meant by "knowledge management" and got answers like "manage, plan, deploy, collect and spread knowledge in an organisation, and do it in a planned manner", and "to create, store, survey, use and revise knowledge".

We can divide between two different usages, or strategies for knowledge management [23]:

- Codification – to systematise and store information that represents the knowledge of the company, and make this available for the people in the company.
- Personalisation – to support the flow of information in a company by storing information about knowledge sources, like a "yellow pages" of who knows what in a company.

We should add here that the codification strategy does not fit all types of knowledge. In situations where knowledge is very context-dependent, and where the context is difficult to transfer, it can be directly dangerous to reuse knowledge without analysing it critically. For some more examples of problems with this strategy, see: [24].

Another strategy than the two mentioned above could be to support the growth of knowledge – the creation of new knowledge by arranging for innovation through special learning environments or expert networks, but that is beyond the scope of this article.

When we go on to discuss computer systems that support knowledge management, we will restrict the scope to systems supporting the first two strategies.

### *The Experience Factory*

One way to manage knowledge is by giving the responsibility for capturing and reusing experience to a separate part of the development organisation. This is the idea behind the "Experience Factory"; a technical and social knowledge management infrastructure to reuse life cycle experience, processes and products, which has been very much referred to in the software

engineering field [19]. Experience is collected from software development projects, and are packaged and stored in an *experience base*.

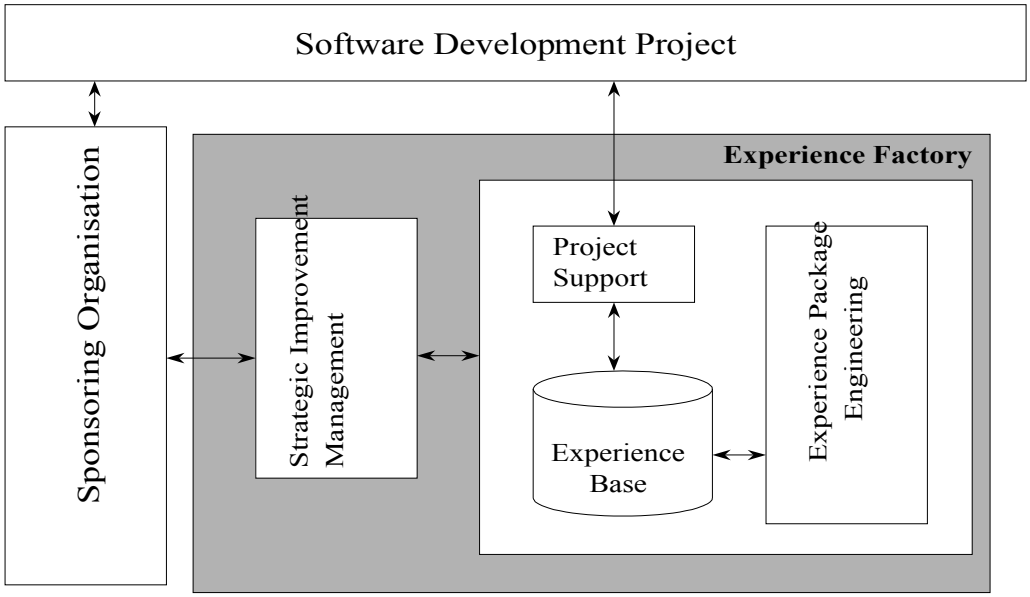


*Figure I: The Experience Factory (taken from* **[25]***).*

The Experience Factory is a part of the *Quality improvement paradigm* [26], which is inspired by work in Total Quality Management. It involves a feedback-loop for improvement initiatives which involve: 1) characterise the environment, 2) set goals, 3) choose process, 4) execute, 5) analyse, and 6) package. So, what we learn from these improvement cycles should be made available for the organisation.

Examples of experience packages are:

- Product Packages - information about the life cycle of a product, information on how to reuse it and lessons learned from reuse.
- Process Packages - information on how to execute a life cycle process, and how to reuse it.
- Relationship Packages - used for analysis and forecasts. Can be cost and defect models, resource models.
- Tool Packages - instructions for use of a tool and experience with it.
- Management Packages - reference information for project managers.

Data Packages - data relevant for a software project or it's activities. Can be project databases or quality records.

perience Factory organisation will then help new software developing projects with earlier experience, and can suggest improvements in processes based on collected experience (we call this "strategic improvement nagement" in Figure I). The interaction between the Experience Factory, the sponsoring organisation and the ware development projects is shown in Figure I.

deas were further elaborated in the Perfect project [25]. Here, we find advise on how to "implement" an erience Factory in an organisation: which steps to take, from "characterising the business situation" and "setting ls", to making an "implementation proposal" and "establish an Experience Factory". It also gives advice on ch roles different people in the organisation can have in this work.

r addition to the original ideas in Experience Factory, is in a paper from Daimler Chrysler [27], where some es that are taken for granted in the original Experience Factory work are clarified:

vement activities in a QIP perspective, is a long-term activity.

jects, process improvement and learning will require additional effort.

transfer between projects requires some similarity between projects.

n Experience Factory would probably be implemented in a different way today, than when the ideas emerged. eb-technology is something that was not developed when this work started.

### *A Model for Knowledge Management*

Now we will present a model for knowledge management “systems” or “programs” that exist in companies. We will use this model, which is shown in Figure II, when discussing case studies later:

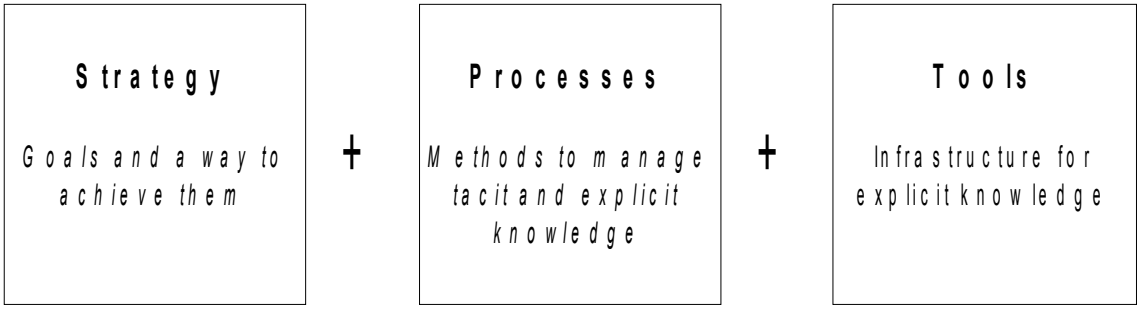


*Figure II: A Model of the Components of a Knowledge Management "System" or "Program".*

We can say that a knowledge management "program" or "system" in a company can consist of three parts; first an overall *strategy* for knowledge management, that is, what are the company goals, and how does it proceed to achieve them. Usually, the goals within software engineering companies are to develop software with less cost, or with a higher quality. But it can also be to make the work of software engineers easier.

By *processes* we mean company activities in order to facilitate knowledge management. This will usually be methods for collecting and distributing knowledge, and can be activities of a separate part of the organisation (such as an Experience Factory), project managers and software developers.

A *tool* to support knowledge management is a software system where operational information, or "knowledge", can be found by different practitioner groups of a software company (like developers, project managers, quality management), usually on an Intranet. The knowledge can be represented in databases, web-pages or files. However, the maintenance effort would be larger with the two last options. Another way to represent knowledge in such a system to make it easy to find relevant information later is to use Case Based Reasoning, like in the COIN EF system in use at the Fraunhofer IESE [28]. We see knowledge as something dynamic, that might be changing over time, so a knowledge management tool, must offer possibilities for revising and discarding knowledge, as well as supplying new knowledge into the system. For a broader view of what the artificial intelligence community view as knowledge management, see [29].

## 1.3 Our Research Question: What are the Approaches and what were the Effects?

This article examines the literature on knowledge management initiatives in the software engineering domain. We examine case studies reported from different organisations, to see 1) What kind of knowledge management approaches that have been used, and 2) What the results of these actions were.

We want to see if the literature on knowledge management in software engineering can support claims such as: increasing the focus on (re)use of experience will improve the situation of both organisations developing software, and improve the work situation for employees. More precisely, we ask: Does the introduction of a knowledge management system:

1. Improve the quality of software?
2. Lower the cost of developing software?
3. Improve the work situation of employees in an organisation?

Now, we first give an overview of research methods, to be able to analyse the claims about benefits of knowledge management that we find in the case studies. Then, we present the research method used here.

# 2 Research Methods

Here, we first present research methods in general, which will be used in the discussion later, and then discuss positive and negative aspects of using literature search to write this survey article.

## 2.1 Research Methods in General

There are several ways of classifying research methods. One is to look at which data sources that are available, and examine if they are primary or secondary. Studies with primary data sources are studies that collect data through surveys, observations or experiments. Secondary data sources are sources for data collected by others, such as conferences and scientific journals.

We could also group research methods according to the subject of study; in software engineering it can be either a process to produce software or a software product.

In an article on research methods in software engineering [30] we find three types of research methods: observational, historical and controlled. We now describe observational methods that are suitable for investigating the phenomenon we are interested in:

By observational we mean collecting information about the subject of our study in a situation where we do not have strict control over the environment. We have to decide what type of information to collect, and a proper way to collect it. Data collection methods may include questionnaires, observation, written reports, logs, etc. Some types of observational studies are *project monitoring*, which is simply to collect data that occurs, for example, during a project being performed. We do not interfere with the project, and do not ask for other information than what the project itself normally produces. If researchers are involved in deciding what information should be collected, we call it a *case study*. If there is not strong disjunction between the subjects of the experiment and the researchers, we call it an *assertion*. This type of study would increase the possibilities of biased results. If we collect data from several projects, we call it a *field study.*

## 2.2 The Research Method Applied Here

The research method used for this article is literature search. We selected a set of papers that were found through searches in databases such as Inspec, Science Citation Index, ACM Digital Library and the IEEE Computer Digital Library. We also searched through proceedings from the last five years of conferences like the International Conference on Software Engineering, The Software Engineering and Knowledge Engineering conference, the International Conference on Product Focused Software Process Improvement and the International Conference on Case-Based Reasoning manually. Some papers were found after suggestions from others, or from references from other papers. We used keywords as "knowledge management", "corporate memory" and "Experience Factory" together with keywords like "software engineering" and "software process improvement". Also, we used a list of 20 knowledge management tools [31], like "grapeVINE", "KnowMan", and "SemioMap" to see if we could find articles reporting experience with those tools in software engineering.

Limitations of this strategy is that we rely on the same understanding of keywords – if there are other papers describing the same topic but using a different vocabulary we would not find them. Another limitation is that it is very common to publish success stories, and not so common to publish results that would either compromise a method, a firm or an organisation.

Of course, all the papers were written for some purpose, which does not necessarily correspond with the purpose we have for analysis. Therefore, the papers may contain incomplete information, or the information might be reported using other terminology than we expect.

Alternatives to choosing literature search would be to conduct formal experiments, or to do a case study of tools in an organisation. The reason for choosing a literature search for investigating our research questions is that there exists literature on the field, and that the results would be less general if we made an experiment or did another case study. There is, however, a lack of articles presenting the results from several case studies, as we will do here.

# 3 Knowledge Management in Software Engineering

As mentioned, a lot of research has been reported about knowledge management in software engineering. When we searched for literature, we found that we could divide work in two major groups: technical development for effective knowledge management, and research that examines the effect of knowledge management on an organisation. We first briefly go through the literature on the first field, and then present the second more thoroughly.

## 3.1 Knowledge Management Technology in Software Engineering

Many tools have been designed to support knowledge management in software development, for example the Experience Management System [32]. Many have used Case-Based Reasoning (CBR), see [33], for retaining and retrieving experience, like [34] who report on the benefits in using this technology to support experimental software engineering more generally, and [35] who are concerned with CBR for building learning software organisations.

Several technologies for experience reuse are evaluated in [36], where the conclusion is that CBR is suitable for reusing experience from software engineering. In [37] we find a number of technical requirements for an experience database. Other work on technology can be divided into work on knowledge acquisition [38] and knowledge reuse [39]. Some work also covers the whole process [40].

Yet other work has been done on using ideas from Experience Factory in the construction of CBR systems, [41], for process improvement in developing educational software [42]. Other technical approaches than CBR have been suggested [43]. Additional work has been done on models for introduction of technical systems for experience reuse in an organisation [44].

We also find descriptions of knowledge management systems in the literature, like one in use in Computas [45] and at Hewlett Packard India [46, 47]. Further, we find descriptions of

knowledge management systems in four companies in Norway [48], together with a discussion on success factors in implementing such systems in organisations.

The University of Nebraska-Lincoln has developed BORE, a research prototype system for knowledge management support in software development [49-51]: This is a tool which contains information in cases about some problem solving experience, and in descriptions of resources like tools, projects, people and development methods. These descriptions are used to find which solutions are relevant when software developers are faced with a new problem.

Another prototype system, is CODE – a general-purpose knowledge management system - which serves as a medium for knowledge capture and transfer, as well as editing or "packaging" knowledge to make it easily available [52].

Yet another technical implementation of a knowledge management system for software engineering is developed at the University of Kaiserslautern [53]. Here, a comprehensive reuse repository has been developed, with possibilities for advanced search and retrieval mechanisms.

## 3. 2 Case Studies of Knowledge Management in Software Engineering

If we look at work on actual use of knowledge management in an organisation, we find much less in the literature. We here report eight case studies, and examine what claims are made about knowledge management in each of them, and describe in what organisational setting each of the case studies were performed. We also place the studies in a category of scientific methods, which were outlined in section 2.1.

### *3.2.1 The NASA Software Engineering Laboratory*

The first implementation of an Experience Factory was at the NASA Software Engineering Laboratory, which is reported in [54]. The Experience Factory is used as described in [19]. Experience in forms of cost data, process data as project methodology information and information on tools and technology used, as well as product data such as change and error information and results on static analysis on delivered code was collected, and used to develop predictive models and to refine the software processes that is used.

The results of this activity is reported as defect rates that went dramatically down (75% from 1987-91, and 37% from 1991-95); the cost of producing software went down by 55% from 1987-91 and 42% from 1991-95. Reuse was improved by 300% from 1987-91 and 8% from 1991-95. Finally, functionality was increased five-fold from 1976-92.

The organisation produces software for NASA only. Thus, it is difficult to compare this organisation with normal, more competitive companies. The article reports lessons learned through 15 years of operation.

### *3.2.2 Daimler Chrysler*

Daimler Chrysler has implemented three experience factories in different environments within a two-year period, in co-operation with the University of Ulm, Germany [55]. The environments were: 1) A department responsible for developing software for the aerospace area with real-time constraints. 2) A department which develops small embedded systems for cars, with special focus on keeping software portable across different micro controllers, and making sure that planned functionality was actually implemented. 3) An administrative software unit that manages internal business processes such as car sales. This unit operates only on requirements, and the software production itself is outsourced.

Other work on experience reuse from Daimler Chrysler can be found in [56, 57]. Three case studies on experience transfer in the company can be found in [58].

The study from Daimler Chrysler takes the form of a "lessons learned" report, and reports the following findings from the three environments (amongst others) [27]:

- There are many sources of reusable experience, and measurement is just one of them.
- There were difficulties in finding how "packaged" users wanted the experience to be.
- Handling qualitative data was a bottleneck.
- Building predictive models from quantitative data was difficult when context information was missing.

We also find a discussion on benefits and problems of introducing an Experience Factory in a top-down and bottom-up manner.

### *3.2.3 Telenor Telecom Software*

In an effort to reuse software development experience, Telenor Telecom Software, a company with 400 software developers in five geographical locations, decided to improve the estimation of software development effort, as well as risk management [59]. To achieve this, they set up:

- An experience reuse process, with new and modified role descriptions.
- An experience database tool, available on the Intranet.
- Resources allocated for experience reuse and for experience database administration.

The experience database was available as an "expert system" which would ask you questions on the nature of a new project, and recommend an estimation model, based on data from earlier projects in the company. It would also give you information on company experts on estimation. This database was

linked to a risk management module, which included risk factors found from interviewing experienced project managers. This module consisted of a set of "best practise" processes, a tool to identify, assess and store risk factors, and a tool to visualise risk exposure over time. In addition to this, new roles for "experience database administrators" were set up – responsible for technical and editorial contents, as well as several roles for "process analysts", responsible for analysing information from processes such as the estimation process, project management process and the testing process.

Although the authors of the article acknowledge that the study was made too early after the initiative was introduced too draw firm conclusions, and that it was difficult to isolate the impact of their own work from other improvement initiatives in the company, they find several indications of improvement:

- The estimation accuracy improved, and estimation models were more widespread in use.
- The focus on experience based risk management increased in the projects.

The organisation accepted the need to collect and share experience.

dy takes the form of a lessons learned report.

#### *3.2.4 Ericsson Software Technology*

Ericsson Software Technology in Sweden have experimented with transfer of experience on a site that develops a wide range of software applications, having around 1600 employees who work in business units of 20 to 30 people. They develop software for telephone switches, base stations and mobile phone management systems. The company has formal communication channels such as meetings, e-mail and written reports, but wanted to establish a corporate culture that facilitate more oral communication of experience [60]. Two organisational roles were invented: "Experience brokers" keep track of what other people in the company know, and match people who can have a benefit from talking to each other. "Experience communicators" help other people solve problems, by teaching them how to solve the problems on their own. The study reports that employees are more motivated when they know that there is a system for transferring experience that works.

The scientific method used in this article is a "lesson learned" report.

#### *3.2.5 An Australian Telecom Company*

Another paper [61] reports on the introduction of an Experience Factory in an Australian telecommunications company. The study was done by the company in co-operation with the Center for Advanced Empirical Software Research at the University of New South Wales, Australia. The goal was to improve the speed and quality of software development, and to enhance experience transfer of process knowledge between projects. This was sought to be done by collecting information that was already documented in the company, and to make it available

and searchable, a kind of a “bottom up” way to start a knowledge management program. The article then reports the usage of this experience base over time, and classifies the searches that were made. A survey amongst the users was conducted, and the “acceptance and judgement of the product was good”. The experience database is also reported to break down barriers between project environments, but this is not supported by quantitative data. Although no information is given on the research method used, it seems that the researchers involved defined the metrics to collect and we can then say that this is a case study. In a later paper, this introduction is described as a “failure” [61]. Although an informal survey amongst users said the “acceptance and judgement of the product (possibility to search an experience base) was good”, the project was abandoned by management. Some reasons for this is discussed in the paper: 1) The researchers felt that there was a lack of ongoing management support for this initiative. 2) The goals and payback-criteria for the project were not clearly defined. 3) The researchers think that a more formal approach should have been used to construct an experience-repository, because the users were physically co-located, and the number of people relatively small. The scientific method here is assertion.

#### *3.2.6 ICL High Performance Systems*

ICL High Performance Systems in the UK has developed an ”Engineering Process Improvement Framework”, which includes a repository for knowledge sharing [62]. The *engineering knowledge base* contains information divided into three categories [63]:

- Projects and processes – descriptions of processes.
- Topic-based instructional material to introduce new concepts.
- General background and further information.

The main objective for introducing this improvement program, was to “improve the predictability of costs and delivery dates of systems and solutions”. The authors claim that there is a “perception by project members that the framework has facilitated the transfer to the new mode of working but this perception is only backed up by anecdotal evidence”. The main benefit has been to “reduce risks to achieving project deliverables within agreed budget, on time, and with the required quality”. Several “lessons learned” are reported, like the importance of management commitment when introducing such a framework, and that the developers should be involved in designing the framework. The scientific method is a lessons learned report.

#### *3.2.7 ICL Finland*

ICL in Finland has also made a knowledge management system. The Finnish part of the company employ more than 800 people working with software development, in applications and services,

and on Internet technology for business applications (like electronic commerce) [64]. ICL classifies their knowledge resources in three groups:

- External knowledge: which includes technical Internet pages, related to customers, software suppliers, tools, technical partners, journals and research centres.
- Structured internal knowledge: includes databases for sales and marketing information and employee competence, as well as examples of frequently used documents, templates, software components, best practise information and research reports.
- Informal internal knowledge: includes electronic discussion forums, news and "project folders". The project folders contain overviews of the projects, news and important announcements, technical documents and reusable components (for a complete list, refer to the paper cited above).

ICL did a survey about use of this "Extranet system" amongst participants in a large project with a peak manning of 50 people, and with an estimated effort of 7800 man-days. The survey was done with a questionnaire, but it is unclear what kind of questionnaire was used, and how many people were interviewed. Based on the survey and interviews, ICL found the following: Most of the project members say "use of the Extranet has supported the work on the project and saved time", it is also "easier to find documents and other information". Further claims are that the "use of project management and software engineering methods has been easier via the Intranet", and document templates are especially appreciated. Learning new project members about project work is also said to be easier, one interviewee estimates that project managers and other project members "save about 30 percent in time, when making a new project member familiar with the system under development". In all, the benefits of the Extranet has been highest in "technical planning, implementation and unit testing". The most important benefit is described as the "better visibility through knowing what kind of projects are going on and have been completed at ICL, and the own unit". The scientific method used is a lessons learned report.

#### *3.2.8 sd&m*

The German Software company sd&m, focuses on designing and implementing large business information systems tailored to customer needs, and used to have problems with rapid growth. In 1999, the company had 700 employees, and had grown by around 50% in some years [65]. Some of the problems were: developers used long time to acquire programming and project management skills, the developers also had problems in coping with many different technological platforms and tools. It was also a problem that insight gained in one project was not applied in others, so the same "mistakes" were repeated many times in the company.
In 1997 sd&m started working with a knowledge management program which involved:

- A knowledge management group consisting of “knowledge brokers”; responsible for the core topics in the company. This involved maintaining a web page on the Intranet, related to these topics.
- The projects are supported by the knowledge brokers, who provide pointers to internal and external knowledge resources. The brokers participate in the project kick-off and touchdown meetings.

Several databases was also made available in the company, listing employees, customers, partners, projects and acquisitions (in Lotus Notes databases), as well as a Skill Database, where all employees assess their own skills.

The company claims that these efforts on knowledge management has reduced the impact of the problems described: “it can be seen very clearly that the problems described ... do not occur nearly as often as before, despite continuing double-digit growth”. The scientific method used is a lessons learned report.

# 4 Discussion

After reviewing the literature on case studies of knowledge management systems, are we able to confirm or disprove the research question from section 1.6? Can we say that increasing the focus on experience use will improve the situation of both organisations developing software, and improve the situation for employees?

In the following, we will first discuss differences in what the companies did, according to the model we outlined in section 1.2.2. Then, we go on to discuss what the companies claim to have achieved.

## 4.1 What the Companies Did

First, let us discuss what kind of goals the case companies had with their knowledge management programs, that is their “strategy”. Than we will discuss what “processes” and “tools” they made use of to achieve this.

### *4.1.1 Strategy*

We find several companies that wanted to improve the situation for their software developers, but did not have clear goals with respect to quality or development costs. Daimler Chrysler, Ericsson Software Technology, sd&m as well as both departments of ICL would come in this category, although ICL High Performance Systems also wanted to “improve the predictability of costs and delivery”. At NASA, The Australian Telecom Company, and Telenor Telecom Software they had cost reduction and quality improvement as a primary goal for their knowledge management activity. Further, if we categorise the

cases according to which type of strategy that was chosen, either to support “personalisation” or “codification”, we find that all of the companies had a codification strategy, and six of eight also support the personalisation strategy (see Table I).

*Table I: A list of what the Companies did, and what Knowledge Management Approach they chose.*

| Company | What did they do? | Knowledge Management Approach | | | | |
|---|---|---|---|---|---|---|
| | | Personalization? | Codification? | Quantitative? | Qualitative? | Reorganisation? |
| NASA SEL | Set up a separate organisation which collected and distributed experience. | Yes | Yes | Yes | | Yes |
| Daimler Chrysler | Created three experience factories in three different company departments. | Yes | Yes | Yes | Yes | Yes |
| Telenor Telecom Software | Made an expert system based on own empirical data for effort estimation and risk management, and modified roles. | Yes | Yes | Yes | Yes | Yes |
| Ericsson Software Technology | Set up new organisational roles to increase oral communication of experience. | Yes | Yes | | Yes | Yes |
| Australian Telecom Company | Collected existing explicit information regarding software development and made it searchable. | | Yes | | Yes | |
| ICL High Performance Systems | Introduced an Intranet-based system with an "engineering knowledge database" | | Yes | | Yes | |
| ICL Finland | Made an Intranet-based system with three structural layers. | Yes | Yes | | Yes | |
| sd&m | Set up a knowledge management group and Intranet system. | Yes | Yes | | Yes | Yes |

#### *4.1.2 Processes*

When looking at what kind of processes that are present in each of the cases, we find that many emphasise that developers should actively participate in collecting and distributing knowledge. Five out of the eight companies did a reorganisation as a part of the knowledge management initiative, to have a separate part of the organisation responsible for this kind of activities. The type of knowledge to be collected and distributed through these processes was both qualitative (like descriptions of experience) and more quantitative (like measurements on the size of code). Three companies had a focus on quantitative knowledge, whilst seven were focusing on qualitative knowledge. At sd&m they specifically mention that they organise kick-off and touch-down meetings in the beginning and end of projects.

#### *4.1.3 Tools*

Finally, let us discuss what kind of tools that the companies were using: Intranet systems for exchanging knowledge were developed . We find that the systems at ICL and Telenor Telecom Software has descriptions of technical work processes in them, and at sd&m and ICL Finland, we find lists of employees skills, as well as lists of customers, partners and projects. Telenor Telecom Software is the only company that developed an expert system for estimation that was available on their Intranet.

## 4.2 What were the Results?

Now we would like to discuss the results of the knowledge management initiatives mentioned in the case studies. We have listed the case studies as well as reported benefit in Table II.

*Table II: A list of Effects of Knowledge Management in the Companies.*

| Company | What was the effect? | Reported benefit | | |
|---|---|---|---|---|
| | | Developer satisfaction? | Lower cost? | Higher quality? |
| NASA SEL | Reduced number of defects, reduced software production costs, increased use. | | Yes | Yes |
| Daimler Chrysler | The case gives no information on the effect for the company. | | | |
| Telenor Telecom Software | The company indicates that estimation accuracy has improved, and focus on risk management has increased. | | Yes | |
| Ericsson Software Technology | The company claims that the initiative was "more valuable" than a database and measurement-approach. | | | |
| Australian Telecom Company | Good acceptance of product amongst users. | Yes | | |
| ICL High Performance Systems | A perception that it has facilitated a "new mode of working" | Yes | | |
| ICL Finland | Saved time, because it is easier to find documents. Easier to learn new project members about project work. | Yes | Yes | |
| sd&m | Previous problems due to rapid growth have diminished. | Yes | | |

If we look at our first research question, whether the introduction of a knowledge management system improves the quality of software, we only find an answer to that in the first article from NASA Software Engineering Laboratory. Although it is mentioned in the article from sd&m that people now do not make the same mistakes again so often, it is not directly said that the software now has higher quality than it used to be.

Then, does the introduction of a knowledge management system lower the cost of developing software? We find evidence for this in three of the cases. Again, this is documented with measurements from the NASA Software Engineering Laboratory. At ICL Finland, it is claimed that project managers and other project members "save about 30 percent in time, when making a new project member familiar with the system under development". At Telenor Telecom Software, the paper authors believe that the work has resulted in improved estimation accuracy. So three out of the eight companies say that the cost in some way is lower after introducing the knowledge management system.

For our last research question "how does the introduction of a knowledge management system influence the work of employees in an organisation?" – we find more in the cases: ICL Finland did an internal survey amongst employees that shows that the initiative "supported project work and saved time", and "made it easier to find documents". Another benefit is described as "better visibility through knowing what kind of projects are going on". sd&m claims that their problems due to rapid growth "do not occur nearly as often as before". And ICL High Performance Systems claim that there is a "perception by project members" that the company is in a "new mode of working". Also at the Australian Telecom Company, people said in a survey that the product "was good". So in all, in four of the eight cases, we find some evidence for improved developer or employee satisfaction.

So, how certain can we be of the findings in these case studies? One major problem with these studies is that they are mostly "lessons learned" reports. That is, it is the same people who have initiated the programs that evaluate them. Another problem is the rather limited number of case studies found, which also limit the possibility to draw general conclusions.

# 5 Conclusion and Further Work

We have analysed eight case studies of knowledge management systems in software engineering companies. We have found that approximately the half of them chose to set up an own

department in the organisation who are responsible for managing knowledge (like an Experience Factory). All of the companies report that they store experience in some way (codification), and many (six out of eight) also facilitate knowledge flow in the organisation (personalisation strategy). Most of the companies focus on transferring qualitative knowledge.

Concerning the benefits on knowledge management systems, it is difficult to draw firm conclusions. If we ask whether the introduction of a knowledge management system improves the quality of software, only one of the studies, from the Software Engineering Laboratory gives a clear answer.
Asking if such actions lower the development costs, again, only the article mentioned gives a clear answer. The other studies are "lessons learned" studies, without focus on collecting measurement data.
Our last subject for discussion was how the introduction of an Experience Factory influences the work of employees in an organisation. We find claims like that the systems have saved time, made work easier, and removed problems due to new personnel that existed before.

Does there seem to be any relation between what kinds of knowledge management initiatives a company engages in, and what the results will be? All the cases report some kind of benefit, and it is difficult to say anything about “how successful” each of the initiatives was. We note that most companies chose a combination of personalisation and codification.

Could it be that the results that are indicated by the companies result from other sources than introducing a knowledge management system? It is difficult to discuss this aspect, because it is not discussed in the source articles that we have examined. But generally, we could expect a kind of “Hawthorne-effect” also in programs that promote knowledge management – that anything you try to measure will increase because of increased attention to those areas. In addition, we can expect most of the companies in the software business to be “more effective” every year, because computers and software tools work faster.

After reviewing the literature on case studies of knowledge management in software engineering, we can conclude that there is a great interest in developing technology to support it, but empirical analysis of how experience sharing actually works is lacking. The validation methods used are mostly lessons learned reports, and it is difficult to compare these reports against each other.

In the future, we plan to do more detailed case studies of knowledge management programmes in software engineering companies, using more formal research methods to add to the existing pool of knowledge in this domain. We would also encourage others to do case studies on knowledge management and software quality, development costs, and improved work situation for employees. Another interesting issue is to look at the difference between companies that has a personalisation *and* codification strategy and companies that apply only one of these.

## Acknowledgements

This work has been partially supported by the project Process Improvement for IT Industry (PROFIT), which is supported by the Norwegian Research Council. We are very grateful to Magne Jørgensen at the University of Oslo, Norway, and Klaus-Dieter Althoff at the Fraunhofer Institute for Experimental Software Engineering in Kaiserslautern, Germany, for helpful comments and discussions on earlier versions of this article.

## References

1. T. Dingsøyr, "An evaluation of Research on Experience Factory," *Proc. of the Workshop on Learning Software Organisations at the international conference on Product-Focused Software Process Improvement*, 2000, pp. 55 - 66.
2. I. Sommerville, *Software Engineering*, Addison Wesley, 1996.
3. A. Bryant, "'It's Engineering Jim... but not as we know it' - Software Engineering - solution to the software crisis, or part of the problem?," *Proc. of the International Conference on Software Engineering (ICSE)*, 2000, pp. 78-87.
4. "Chaos," Dennis, Massachusets, The Standish Group Report 1995.
5. B. Joy and K. Kennedy, "Information Technology Research: Investing in Our Future," Report from the President's Information Technology Advisory Committee February 24. 1999.
6. J.-L. Lions, "Ariane 5 Flight 501 Failure," Report from the Inquiry Board, Paris 19. July 1996.
7. T. Collins and D. Bicknell, *Crash. Learning from the World's Worst Computer Disasters*, Simon & Schuster, 1997.
8. R. L. Glass, *Software Runaways: Lessons Learned from Massive Software Project Failures*, Prentice Hall, 1998, pp. 259.
9. R. L. Glass, "Talk About a Software Crisis - Not!" *The Journal of Systems and Software*, **55**, (2000) 1-2.
10. R. L. Glass, "The realities of Software Technology Payoffs", *Communications of the ACM*, **42**, (1999) 74-79.
11. *Oxford Dictionary and Thesaurus*, 1995.

12. M. Polanyi, *The Tacit Dimension*, Doubleday, 1967, pp. 108.
13. I. Nonaka and H. Takeuchi, *The Knowledge-Creating Company*, Oxford University Press, 1995, pp. 284.
14. A. Aamodt and M. Nygård, "Different roles and mutual dependencies of data, information, and knowledge - an AI perspective on their integration", *Data and Knowledge Engineering*, **16**, (1995) 191-222.
15. H. L. Dreyfus, *What Computers Still Can't Do : a critique of artificial reason*, MIT Press, 1992, pp. 354.
16. R. S. Taylor, "Information Use Environments," *Proc. of Progress in Communication Science*, 1991, pp. 217-254.
17. B. N. Weaver and W. L. Bishop, *The Corporate Memory : a profitable and practical approach to information management and retention systems*, John Wiley, 1974, pp. 257.
18. M. S. Ackerman and C. A. Halverson, "Reexamining Organizational Memory", *Communications of the ACM*, **43**, (2000) 59-64.
19. V. R. Basili, G. Caldiera, and H. D. Rombach, "The Experience Factory", in *Encyclopedia of Software Engineering*, Eds. J. J. Marciniak, John Wiley, 1994, pp. 469-476.
20. K. M. Wiig, "Knowledge Management: Where Did It Come From and Where Will It Go?" *Expert Systems with Applications*, **13**, (1997) 1-14.
21. T. H. Davenport, D. W. D. Long, and M. C. Beers, "Successful Knowledge Management Projects", *Sloan Management Review*, **Winter**, (1998) 43-57.
22. K. M. Wiig, *Knowledge Management Methods*, Schema Press, 1995, pp. 489.
23. M. T. Hansen, "The Search-Transfer Problem: The Role of Weak Ties in Sharing Knowledge accross Organizational Subunits", *Administrative Science Quarterly*, **44**, (1999) 82-111.
24. M. Jørgensen and D. Sjøberg, "The Importance of NOT Learning from Experience," *Proc. of the EuroSPI Conference.*, 2000.
25. Perfect consortium, "PIA Experience Factory, The PEF Model," ESPRIT Project 9090 D-BL-PEF-2-PERFECT9090, 1996.
26. V. R. Basili, "Quantitative Evaluation of Software Engineering Methodology," *Proc. of the First Pan Pacific Computer Conference*, 1985.
27. F. Houdek and K. Schneider, "Software Experience Center: The Evolution of the Experience Factory Concept," *Proc. of the Twenty-Fourth Annual NASA Software Engineering Workshop*, 1999.
28. C. Tautz, "Customizing Software Engineering Experience Management to Organizational Needs," PhD thesis, *Department of Informatics*, University of Kaiserslautern, Germany, 2000.
29. R. G. Smith and A. Farquhar, "The Road Ahead for Knowledge Management", *AI Magazine*, **21**, (2000) 17-40.
30. M. V. Zelkowitz and D. R. Wallace, "Experimental Models for Validating Technology", *IEEE Computer*, **May**, (1998) 23-31.

31. A. E. Goodall, "Survey of Knowledge Management Tools - Part I and II," in *Intelligence in industry*, vol. 8, 1999.
32. C. Seaman, M. Mendonca, V. Basili, and Y.-M. Kim, "An Experience Management System for a Software Consulting Organisation," *Proc. of the Twenty-fourth annual NASA Software Engineering Workshop*, 1999.
33. A. Aamodt and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", *AI Communications*, **7**, (1994) 39-59.
34. K.-D. Althoff, A. Birk, C. G. V. Wangenheim, and C. Tautz, "CBR for Experimental Software Engineering," *Proc. of the Case-Based Reasoning Technology - From Foundations to Application*, 1998, pp. 235-254.
35. K.-D. Althoff, F. Bomarius, and C. Tautz, "Using Case-Based Reasoning Technology to Build Learning Software Organizations," *Proc. of the Interdisciplinary Workshop on Building, Maintaining, and Using Organisational Memories, OM-98*, 1998.
36. C. G. V. Wangenheim, K.-D. Althof, R. M. Barcia, and C. Tautz, "Evaluation of Technologies for Packing and Reusing Software Engineering Experience," Fraunhofer IESE technical report 055.98/E, 1998.
37. M. Broomé and P. Runeson, "Technical Requirements for the Implementation of an Experience Base," *Proc. of the international conference on Software Engineering and Knowledge Engineering, SEKE'99*, 1999, pp. 1-9.
38. A. Birk, D. Surmann, and K.-D. Althoff, "Applications of Knowledge Acquisition in Experimental Software Engineering," *Proc. of the conference on Knowledge Acquisition, Modeling and Management, EKAW'99*, 1999, pp. 67-84.
39. C. Tautz and K.-D. Althof, "Using Case-Based Reasoning for Reusing Software Knowledge," *Proc. of the Second international conference, ICCBR'97*, 1997.
40. A. Birk and C. Tautz, "Knowledge Management of Software Engineering Lessons Learned," *Proc. of the 10th International Conference on Software Engineering and Knowledge Engineering, SEKE'98*, 1998.
41. R. Bergmann and M. Göker, "Developing Industrial Case-Based Reasoning Applications Using the INRECA Methodology," *Proc. of the Workshop at the International Joint Conference on Artificial Intelligence, IJCAI - Automating the Construction of Case Based Reasoners*, 1999.
42. J.-W. Van Aalst, "Knowledge Management in Courseware Development," PhD thesis, Technical University Delft, 2001, pp. 179.
43. R. Feldmann, J. Münch, and S. Vorwieger, "Towards Goal-Oriented Organizational Learning: Representing and Maintaining Knowledge in an Experience Base," *Proc. of the The Tenth International Conference on Software Engineering and Knowledge Engineering, SEKE'98*, 1998.
44. T. Dingsøyr, "A lifecycle process for experience databases," *Proc. of the ICCBR'99 workshops: Challenges for case-based reasoning:* , 1999, pp. 9-13.

45. S. Carlsen, S. G. Johnsen, H. D. Jørgensen, G. J. Coll, Å. Mæhle, A. Carlsen, and M. Hatling, "Knowledge Re-Activation Mediated Through Knowledge Carriers," *Proc. of the International Conference on Management of Information and Communication Technology*, 1999.
46. R. Bhave, N. C. Narendra, I. P. Pal, and S. Krishnaswamy, "A Product-Line Approach towards Developing Knowledge Management (KM) Systems," *to appear*.
47. R. Bhave and N. C. Narendra, "An innovative strategy for organizational learning," *Proc. of the World Congress on Total Quality*, 2000.
48. R. Conradi and T. Dingsøyr, "Software experience bases: a consolidated evaluation and status report," *Proc. of the Second International Conference on Product Focused Software Process Improvement, PROFES*, 2000, pp. 391 - 406.
49. S. Henniger, "Capturing and Formalizing Best Practices in a Software Development Organization," *Proc. of the 9th International Conference on Software Engineering and Knowledge Engineering, SEKE'97*, 1997.
50. S. Henniger, "Case-Based Knowledge Management Tools in Software Development", *Automated Software Engineering*, **4**, (1997) 319-339.
51. S. Henniger and J. Schlabach, "A Tool for Managing Software Development Knowledge," *Proc. of the International Conference on Software Engineering (ICSE01)*, 2001.
52. D. Skuce, "Knowledge management in software design : a tool and a trial", *Software Engineering Journal*, **10**, (1995) 183-193.
53. R. L. Feldmann, "Developing a Tailored Reuse Repository Structure - experience and first results," *Proc. of the Workshop on Learning Software Organisations, SEKE'99*, 1999.
54. V. R. Basili, G. Caldiera, F. Mcgarry, R. Pajerski, G. Page, and S. Waligora, "The Software Engineering Laboratory - An operational software experience factory," *Proc. of the Proceedings of the 14th International Conference on Software Engineering, ICSE 14*, 1992, pp. 370-381.
55. F. Houdek, K. Schneider, and E. Wieser, "Establishing Experience Factories at Daimler-Benz. An Experience Report," *Proc. of the 20th International Conference on Software Engineering, ICSE 20*, 1998, pp. 443 - 447.
56. D. Landes, K. Schneider, and F. Houdek, "Organizational learning and experience documentation in industrial software projects", *International Journal of Human-Computer Studies*, **51**, (1999) 643-661.
57. F. Sazama, "An organizational approach for experience-based process improvement in software engineering: The Software Experience Center", in *Software Quality : State of the art in management, testing and tools*, Eds. M. Wieczorek and D. Mayerhoff, Springer Verlag, 2000, pp. 73-90.
58. E. Wieser, F. Houdek, and K. Schneider, "Systematic Experience Transfer : Three Case Studies From a Cognitive Point of View," *Proc. of the Second International Conference on Product Focused Software Process Improvement, PROFES 2000*, 1999, pp. 323 - 344.

59. M. Jørgensen, R. Conradi, and D. Sjøberg, "Reuse of software development experience at Telenor Telecom Software," *Proc. of the European Software Process Improvement Conference (EuroSPI'98)*, 1998.

60. C. Johansson, P. Hall, and M. Coquard, ""Talk to Paula and Peter - They Are Experienced" - The Experience Engine in a Nutshell", in *Learning Software Organizations : methodology and applications; proceedings from the 11th International Conference on Software Engineering and Knowledge Engineering, SEKE '99, Kaiserslautern, Germany, June 16 -19, 1999*, Eds. G. Ruhe and F. Bomarius, Springer Verlag, 1999, pp. 171 - 186.

61. A. Koennecker, R. Jeffery, and G. Low, "Lessons Learned From the Failure of an Experience Base Initiative Using a Bottom-Up Development Paradigm," *Proc. of the Twenty-Fourth Annual Software Engineering Workshop*, 1999.

62. B. Chatters, "Implementing an Experience Factory: Maintenance and evolution of the software and systems development process," *Proc. of the IEEE International Conference on Software Maintenance*, 1999, pp. 146-151.

63. B. Chatters, J. Hood, and N. Jefferson, "Epik: Engineering Proceess Improvement and Knowledge Sharing," in *ICL Systems Journal*, 2000, pp. 83 - 104.

64. M. Markkula, "Knowledge Management in Software Engineering Projects," *Proc. of the Proceedings of the international conference on Software Engineering and Knowledge Engineering, SEKE'99*, 1999, pp. 20-27.

65. P. Brössler, "Knowledge Management at a Software House : An Experience Report", in *Learning Software Organizations : methodology and applications; proceedings from the 11th International Conference on Software Engineering and Knowledge Engineering, SEKE '99, Kaiserslautern, Germany, June 16 -19, 1999*, Eds. G. Ruhe and F. Bomarius, Springer Verlag, 1999, pp. 163 - 170.