Knowledge Management in Courseware Development

Jan-Willem van Aalst

Knowledge Management in Courseware Development

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.F. Wakker,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op 24 april 2001 om 16:00 uur
door Jan-Willem VAN AALST
informatica ingenieur
geboren te Vlissingen

Dit proefschrift is goedgekeurd door de promotoren:

Prof. dr. ir. J.L.G. Dietz, Technische Universiteit Delft
Prof. dr. G.A.M. Kempen, Universiteit Leiden

Samenstelling promotiecommissie:

Rector Magnificus, voorzitter
Prof. dr. ir. J.L.G. Dietz, Technische Universiteit Delft, promotor
Prof. dr. G.A.M. Kempen, Universiteit Leiden, promotor
Dr. ir. C.A.P.G. van der Mast, Technische Universiteit Delft
Prof. dr. H. Koppelaar, Technische Universiteit Delft
Prof. dr. H.G. Sol, Technische Universiteit Delft
Prof. dr. J.C.M.M. Moonen, Universiteit Twente
Prof. T.T. Carey, Ph.D., University of Waterloo, Ontario, Canada

Published and distributed by: DUP Science

DUP Science is an imprint of
Delft University Press
P.O. Box 98
2600 MG Delft
The Netherlands
Telephone: +31 15 2785121
Telefax: +31 15 2781661
E-mail: DUP@Library.TUDelft.NL

ISBN 90-407-2161-0

Keywords: educational multimedia, software process improvement, knowledge management

Printed in The Netherlands

"And reach for the heavens
And hope for the future
And all that we can be
Not what we are."

– John Denver

Table of Contents

Preface

In this thesis we examine ways to facilitate project teams involved in producing courseware, by looking at the possibilities that a certain kind of knowledge management can offer. The thesis contains the results of a research project called ICOM (Improving Control Over Multidisciplinary projects) that was initiated in 1996 by Delft University of Technology, the Netherlands, and Atos Origin, The Netherlands. Later, the University of Leiden, also from The Netherlands, was involved as well to assist in setting up a quantitative analysis of project interviews.

Atos Origin is an IT service provider, a company that works chiefly for the top 500 fortune companies. Atos Origin has over 27,000 employees in over thirty countries worldwide (2001); some 7,000 of these work in The Netherlands. One department within this company in The Netherlands is eBS, with several local units, one of which is called Business Communication. This local unit within eBS produces educational systems, often with a large multimedia component. Business Communication consists of some forty educational multimedia experts. Much of the research was done within this local unit. The research budget holder of Atos Origin in The Netherlands agreed to act as sponsor of research project. Figure p.1 shows the organizational diagram for the context of the research project:

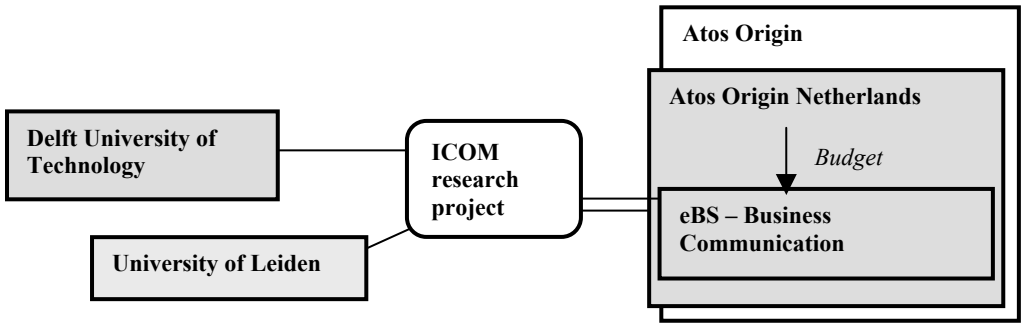


Figure p.1. *Organizational structure of the ICOM research project.*

The strategy of this research project focuses (*1*) on the improvements of the courseware development process it can provide for courseware project teams, and (*2*) on the applicability of this research for other researchers and other companies. This has implications for the environment in which the research is carried out. Since the research was done in an empirical situation with many poorly controllable factors causing "noise" on various levels, there were many influences that might have been better controlled in a laboratory-like environment. Since such an environment was not available, this thesis does not aim to present a theoretical methodology, but it rather presents the experiences, in a real life situation, with an instrument for a certain kind of knowledge management in courseware projects, the usefulness of which is made plausible by testing the research hypothesis through a case study and an experiment.

A steering committee, called the ICOM steering committee, was formed in 1996 to provide guidance and control over the research project. The steering committee held regular meetings three or four times each year. At each meeting the author reported on the progress and direction of research. The steering committee commented on this each time. Unclear issues were discussed and the research boundaries constantly verified. Corrective actions were indicated by the steering committee. The following people were involved in the ICOM steering committee:

- *Jan Dietz*. Professor of Information Systems at Delft University of Technology. The official "promotor" of the research project for the entire duration of the research.
- *Toon Witkam*. Consultant, researcher, and emeritus Professor of Ergonomics at Delft University of Technology. With Atos Origin up until and including most of 1999. Research budget holder of Atos Origin in this period. Retired in 1999.
- *Charles van der Mast*. Associate Professor at Delft University of Technology. Mentor and advisor on a regular basis for the entire duration of the research.
- *Gerard Kempen*. Professor of Cognitive Psychology at the University of Leiden. Co-promotor of the research project from 1998 on.
- *Johan Vader*. Project manager and Human Resource Manager at Atos Origin EEBS – Business Communication. Representative for the entire duration of the research.
- *Richard Thoben*. Service Line Manager at Atos Origin. With Atos Origin for the entire duration of the research. Responsible for research budget from 1999 on.
- *Marcel Theunissen*. Consultant. With Atos Origin from the start of the research project until the end of 1997. Moved to Australia in 1998; out of scope since then.
- *Johan Versendaal*. Consultant. With Atos Origin from the start of the research project until mid-1998. Went to work at BAAN company in 1998.

Zeist, The Netherlands
April 2001

1. Introduction

In this chapter, we introduce the reader to the main research topic. The aim of this chapter is to clarify the focus that we chose for this research project. We start by describing the context of the thesis, namely educational systems with a strong multimedia component, and we include a brief history of research in educational systems. We illustrate the special characteristics of courseware as opposed to "general" software. We also introduce the notion of software process improvement in relation to educational systems; a relationship that we feel deserves more attention than it has gotten so far. The chapter concludes with a comprehensive reading guide, showing all consecutive chapters in a schematic flow.

1.1. Educational systems and multimedia

An *educational system* can be seen, on a high level, as the combination of a teaching unit and a learning unit. The combination exists for instructional purposes. For the purpose of this thesis, an educational system always includes educational software, which we call *courseware*. In this thesis, we see *multimedia* as a collective term for the multiple media often used in courseware, plus the disciplines that are required to create the media. Recent educational systems often include multiple media such as graphics, animations, audio (sounds, music and speech) and video (see [FAL1995] for examples). More generally speaking, educational systems and multimedia are increasingly linked, partly because the possibilities for conveniently employing multimedia improve due to more powerful computers [AND1990; VAU1994; VBE1992B], and partly because it is now known that learning experiences can be enriched through the use of multimedia [BOY1996; CHA1990; CLA1994; JAC1997].

An important argument for using multimedia in educational systems is the *engagement* and *immersion* that multimedia can create with the learners that use the educational system (see [JAC1997, VAU1994] for definitions of engagement and immersion). This is sometimes called the *persuasion* factor [FOG1999]. Multimedia is often used to (*1*) stimulate the motivation and enthusiasm of the learner, and to (*2*) present scenarios to the learner that resemble real life situations, as opposed to offering only a text-based story [ENG1996; JAC1997]. We must be aware that the word "multimedia" implies the inclusion of multiple disciplines *in addition* to the specialized disciplines that are required for the development of educational systems (which in general need not always be computer-based, although in this thesis they are). Therefore, when using the word multimedia, we emphasize that we are referring more to disciplines than to actual media. Multidisciplinary is a term that recurs throughout the thesis.
In other words, we direct our focus to educational systems in which the courseware makes up an important part, and in which there is a large multimedia component.

Software applications that employ multimedia are mostly found in (a) games, (b) marketing and sales applications, and (c) educational systems [ALB1996; AND1990; CLA1994]. We choose to focus on educational systems only, and then only those with a large courseware component. There are various entities involved in the use of courseware. Van der Mast describes the relationship between the student and the teacher in the context of courseware using components and attributes of software concerning learning paths, scoring, content, and other facilities [VDM1991]. Figure 1 shows the model that describes these relationships.

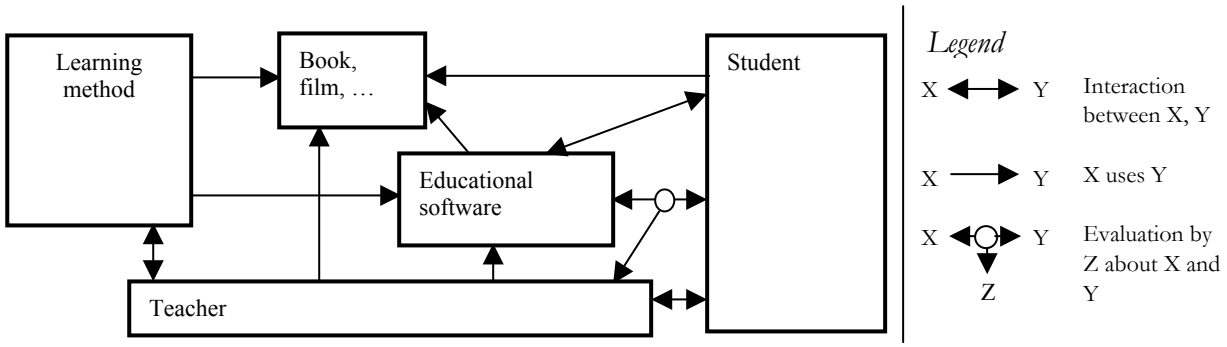


Figure 1. *An educational system: relationships between the teacher and the student employing courseware*

In Figure 1, the teacher makes choices regarding the learning method, which in its turn employs books and other traditional learning means on the one hand, and courseware as new learning means on the other hand. There is of course interaction between the student and the teacher. The student uses books and other traditional learning means on the one hand (one-way interaction), and courseware as new learning means on the other hand (two-way interaction). Finally, there is an evaluation by the teacher, namely that of the interaction between the student and the courseware. In this sense, we see the combination of the courseware and the teacher as the *teaching unit*. The student, then, we call the *learning unit*. We call the remaining entities *support* units. We see that the courseware is only a part (although an important part) of the entire educational system.

In "general"[1] information systems development, the purpose of the information system is usually to help people carry out their tasks more efficiently and effectively. We often forget that it may require an accompanying educational system to help people master the skills required to achieve that purpose, for example mastering the functionality of certain S.A.P. software modules (although in theory such an educational system is not required if the information system is well-designed). This is one of the important distinctions between educational systems and "general" information systems: there is an explicit instructional purpose involved, which in its turn requires the involvement of relevant disciplines. This makes the realization of educational systems generally more complex than realizing an information system that helps people achieve some task or set of tasks more easily. This observation is important, because in this thesis we are looking for topics specific to the development of educational systems. We will describe more of these characteristics in consecutive sections. This helps us to clearly identify the research goal that we present in chapter 3.

Let us now look at the components of an educational system, by zooming in on the teaching unit and learning unit as described earlier (The support units are not unimportant by themselves, but they lie outside the scope of the thesis). We can view an educational system as shown in Figure 2. From this viewpoint, an educational system consists of hardware, software, data, procedures and people. Sometimes hardware and software are conveniently grouped as *infrastructure*. The hardware, software, and (electronically stored) data are usually referred to as making up the entity in Figure 1 that we call courseware (which is part of the teaching unit). The learning unit, consisting of the

[1] We use the term "general" information systems development several times. With this, we refer to information systems that are developed for administrative purposes in the broadest sense.

student (who is part of the *people* that are involved), uses the courseware according to a set of *procedures*.

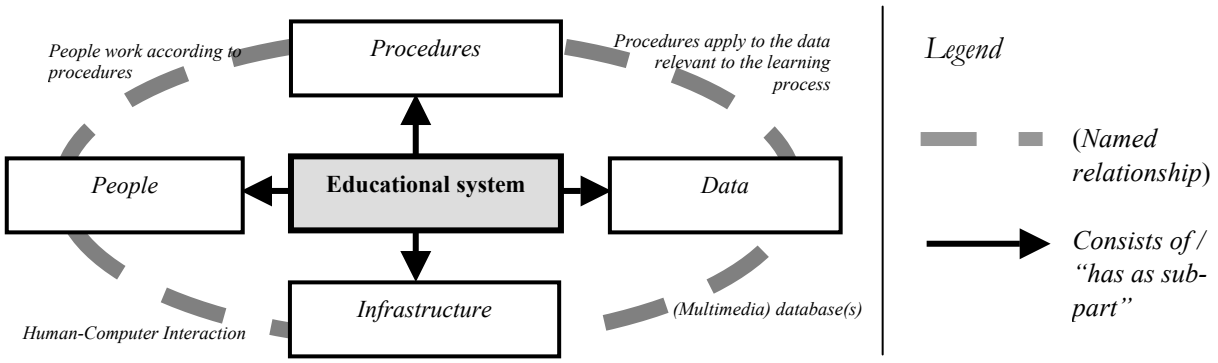


Figure 2. *Components of an educational system.*

Throughout the years, many names are used for the term courseware. Some of these names include Computer Based Training (CBT), Computer Based Learning (CBL), Computer Assisted Instruction (CAI), Computer Managed Instruction (CMI) and Computer Assisted (Aided) Learning (CAL). The difference in these names mainly depends on the importance of the role of the computer in the educational software, especially in the amount of control that the computer has over the learning process. For the purpose of this thesis, they will be considered synonyms, and we will use the term *courseware* throughout. Please note that courseware makes up the infrastructure and data part of an educational system, and that an educational system is more than just the courseware that is used; so these two may not be considered synonyms.

The occurrence of the word *people* in the components shown in Figure 2 is even more important than it already is in "general" information systems. This is because of the usually very high level of interaction between the courseware and the learner. It is well known that the failure of many software applications is largely due to neglecting human factors in the interaction between the software and the user [DEM187; GIBB1994; ROU1992; PRE1994; SHN1992]. Because human factors are even more important with courseware due to its highly interactive nature and the learning process that is involved, much research is now available on designing the (quality of the) interaction between the courseware and the user [BOY1995; GER1987; DIX1993; PRE1994; VDM1995A]. This thesis is concerned with the people aspect of courseware as well, although on a different level, namely the quality of the way of working of the *project team* that is involved in producing courseware. This means that our research is about (the quality of) the courseware development process, and not about (the quality of) the courseware product. We will discuss the relationship between the quality of the development process and the quality of the resulting product later. In this chapter we focus mostly on courseware as a product, or entity; while in chapter 2 we investigate the courseware development process.

To further investigate the specifics of educational systems, we must describe their main characteristics from a didactical viewpoint also. From this viewpoint, the learning unit is still the student, and the teaching unit is composed of four distinct, though not separate, components. We show this in Figure 3, where we see the student interacting with the user interface component of the educational system, which also contains a pedagogic component, a subject matter component, and,

if designed well, a student model component. Please note that because of the highly interactive nature of educational systems, especially the user interface component plays a crucial role in this model, often more so than in other types of software.

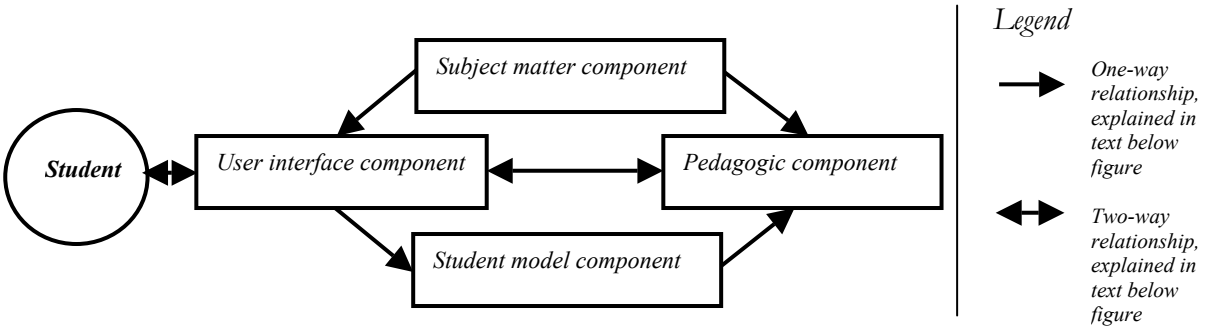


Figure 3. *Essential components of educational systems on a logical level* [VDM1995A]

In Figure 3, the two-way relationship between the student and the user interface component is commonly referred to as "human-computer interaction": the way in which the student interacts with the courseware. The user interface presents the subject matter as defined in the subject matter component. The user interface makes use of various media (text, graphics, audio, video, animations) according to an instructional strategy as defined in the pedagogic component. The pedagogic approach depends on the subject matter involved (different subjects require different pedagogical approaches), and additionally the kind of students involved, which is modeled in the student model component. When considering Figure 3 we again see why the development of courseware is generally more complex than general software development. The four components involved in the teaching unit each require their own discipline(s) to produce, while software in general is usually developed with less different disciplines that need to co-operate closely. In other words, the multi-disciplinary nature of developing courseware implies that there are courseware-specific issues to investigate to improve the development process.

Courseware development can be categorized into product issues and process issues, similar to information systems development in general. Both types of issues contribute to the success of a courseware product, and both deserve attention. The focus of this thesis, though, is on the process aspects, not on the product aspects. In other words, we focus on the quality of the process of developing courseware, and not so much on the quality of the courseware product itself [2]. The quality of the development process is partly dependent on the level of expertise of the people involved in the courseware project, but also on more complex aspects such as interdisciplinary communication, multidisciplinary project management, stress due to time and budget constraints, robustness of sophisticated multimedia tools that are used, and the relationship between the (knowledgeable) team and the (often not knowledgeable) customer.

Let us first look at important topics that must be addressed regarding the quality of both the product side and the process side of courseware. Van der Mast [VDM1983] proposed the following breakup of activities related to the *product* aspect of courseware:

[2] Of course, the quality of the product partly depends on the quality of the process. Just what we mean by "process" we define in chapter 3.

- **Development** of courseware. This includes requirements analysis; functional design; usability and interaction design; graphics design; technical infrastructure in relation to product performance on a specified hardware platform; and inclusion of voice, music, animations and video (see also [AND1990; ENG1996; GER1987; BOY1996]).
- **Presentation** to the user. Usually, the dialogue between the product and the student is displayed visually on the screen through a (graphical) user interface. Occasionally, audio material is used as well. The quality of this presentation contributes significantly to the effectiveness of the learning experience.
- **Recording** the performance of the user and the product. Too often undervalued, this includes measurements on the student's progress while the product is used. This is required for the next function, the evaluation of (the effectiveness of) the product.
- **Evaluation** of the product. In order to be able to improve both the educational software product and the learning effect of the user, an evaluation is done to list strengths and weaknesses of the courseware. *Formative evaluation* is concerned with the question whether or not the design is fit to the learning needs of the student or students, while *summative evaluation* is targeted towards the achievement and progress of the students in retrospect (did the training really help).
- **Distribution** of the product. An often-neglected function, part of the success of educational software depends on the way the product is brought (distributed) to the student. Recently, with the Internet becoming widely available, there are more possibilities for distributing courseware. Computer Based Training being distributed over the Internet is sometimes referred to as Flexible Distance Learning (FDL) or Web Based Training (WBT).
- **Deployment** of the product. Help facilities, whether online or by humans (helpdesk) support the use of the product. Also, formal procedures for the effective use of the product are required in this function to make it a success.

Process issues include issues – often hard to identify in a quantitative way – that arise when working on the main functions described above. Examples are [cf. VAA1998A; DEM1987]:

- Interdisciplinary **communication**. A programmer and a graphic artist often have very different views on the concept, design and realization of courseware. This can lead to misunderstandings and even irritations and frustrations [REI1996]. On the other hand, having several different viewpoints on a design can also be seen as a strength, if properly managed [DEM1987; ROU1992; TJO1991].
- The working relationship with **customers** that have little or no experience with courseware or multimedia. The expectations of the customer regarding the courseware must be carefully managed in order to avoid frustrations through misunderstandings. This is because a customer usually is not able to estimate the required effort of much of the desired functionality [VAA1998A; ENG1996; GER1987].
- Managing the motivation of a team of several disciplines. A specialized form of **project management**, this is often regarded as one of the most difficult roles in the educational multimedia project team [GER1987; VBE1992B, VDM1995A]. Also, because of all the disciplines involved, this form of project management should be seen as a group responsibility, not just the task of some individual.
- Difficulties with keeping up-to-date with new releases of multimedia authoring systems. Courseware sets stringent requirements on the quality of the hardware and software tools

that are used for development. Once programmers learn to cope with the bugs of version *x* of some multimedia authoring software application, new bugs are introduced in version *x*+1. Many courseware development experts feel that the pace of new releases and the pace of ICT in general is going too fast.

- **Stress** resulting from problems with the process issues described above. This probably contributes significantly to lower product quality, although this has not yet been confirmed in well-controlled experiments.

This thesis addresses primarily the process-related issues; we especially investigate the quality of the way of working of courseware development teams. We address the following questions:

- What problems regarding the development process itself do courseware development teams encounter while producing courseware?
- What causes these problems? Is there existing literature about the causes?
- How can we create solutions for the problems that we have identified?
- How can we determine if a proposed solution helps to solve these problems?
- Can existing software process improvement theory be applied to courseware development?
- How can the knowledge and expertise required by the courseware project teams be properly managed and facilitated?

Please note that these are topics that are on an operational level of developing courseware. Issues on a strategic level, such as how to link the deployment of courseware to high-level business goals, are not investigated here. The main reason for our choice of focus is that much research about product issues of courseware is already published (see for example [VBE1992B, AND1990, CLA1994, ALB1996, VDM1995A]), but research on the process issues of courseware development is much harder to find (see [VBE1992A; ROB1994; VAA1998A]). However, if we look at the questions that we have listed above, we must realize that there is a significant amount of literature available on topics such as software process improvement and, to a lesser degree, knowledge management. The time seems right for an investigation of how research results from these research fields can be employed to address the courseware development questions that we have just posed.

1.2. History of courseware

Any attempt to summarize the history of courseware should provide an overview of the developments in research about educational systems in general. In the first half of the twentieth century psychologists developed, for the first time, generic formal models of the human learning process. These were mainly concerned with how humans internalize new knowledge, and ways to help instructors achieve this transfer of knowledge. One of the first widely used formal models for educational instruction was called Programmed Instruction (PI), at first extensively used in the military world. PI included sections of text, each consisting of information to read, questions to answer, and instructions on how to proceed to the next section.

From a certain point of view, PI may be considered to be an educational system, only without the hardware and software, simply because these were not available then. With the advent of powerful computers from the nineteen sixties on, formal models such as PI started to become automated as

well. Throughout the nineteen forties and fifties, the number of formal didactic styles expanded too. An overview of the most widely used is given by [RUS1983], and is briefly summarized here:

1. **Drill and Practice**. One of the most basic didactic styles, it has also been one of the most effective for certain types of knowledge transfer. This is basically a programmed sequence of exercises, each exercise providing feedback as information of the result ("right or wrong"). Through basic repetition, the knowledge becomes internalized.
2. **Tutorial**. The tutorial can be seen as a significant improvement of the "drill and practice" method in that it provides a richer learning environment for the learner: within the tutorial, items are presented, and then followed by one or more questions. Depending on the answer given, one of various teaching paths is followed.
3. **Simulation**. Especially with the advent of powerful computers in the eighties, this is a learning style that is being increasingly frequently employed. A simulation is the execution of a learning scenario using variables that can be influenced by the learner. The outcome of the scenario can be analyzed and mapped to the theory.
4. **Modeling**. The most elaborate of the didactic styles and fit for certain types of learners only, modeling provides an environment in which students have extensive control over the learning environment, and the medium that is used is less important. Developing courseware using this didactic style is rare because it is highly time-consuming and very costly.

These didactical styles allow the learner to focus more on the content of the study and the actual learning goals, instead of on how the instruction is organized. Educational systems development teams have gratefully employed these didactical styles in courseware, allowing the user to concentrate on the learning process even more [VDM1983; EBE1988]. At first, however, in the sixties and seventies, partly due to technical limitations, partly due to lack of knowledge about human factors in computing systems (human-computer interaction), these courseware products – and highly interactive systems in general – were unavoidably of a crude nature, despite the best intentions [VBE1992B, DIX1993; NOR1988]. It was only with the introduction of the graphical user interface and multimedia into ever more powerful computers that courseware as we now know it became a reality [HEB1977; SHN1992].

As with many new technologies, the military world (especially in the U.S.A.) experimented with sophisticated courseware first, and soon universities and large companies followed. Thousands of companies worldwide are currently engaged in producing courseware [ENG1996, BOY1996], and the number of disciplines involved in this production process has increased steadily, as it did for example in the movie world in the first half of the twentieth century [LAU1993, MON1981]. In other words, we observe that educational software development is about half a century behind the movie industry, which is partly due to the rate at which computer power has developed throughout the decades.

Much research into improving the quality of educational systems has been reported since the eighties (see for example [ALB1996, CAR1990, CLA1994, GER1987, HAR1988, MYE1994, VAA1996, VBE1992A, VDM1995A]). Many governments have initiated formal programs to stimulate research to improve the quality of courseware. There is also literature that is geared towards learning from experiences and expertise from already existing media domains (e.g., movie, music, theatre,

architecture [LAU1993; WIN1996]), which is relevant to courseware development, though unfortunately not practiced widely enough.

The initial approach to developing courseware was chiefly a software engineering one, with a strong focus on technical aspects (see [HEB1977; RUS1983; DEM1987]). It is only halfway down the eighties of the twentieth century that other disciplines became involved in producing courseware as well, notably artistic and cognitive disciplines. By then, human-computer interaction had become a major field of research [MYE1994; DIX1993; PRE1994; SHN1992], with international conferences attracting thousands of researchers and practitioners each year. The value of the subsequent contribution of interaction designers, user interface designers and researchers to educational systems development cannot be overestimated [DIX1993; MYE1994]. Courseware development now pays much attention to aspects such as task analysis, interaction design, graphical design, didactical design and other non-technical disciplines. Many aspects of software engineering, cognitive psychology and human-computer interaction were combined to form new life-cycle models for courseware development, such as those by Stevens [STE1986], Ibrahim [IBR1990], and Van der Mast and Rantanen [VDM1992]. We further discuss life cycles for courseware development, and the many disciplines involved in producing courseware, in chapter 2.

With the steady increase of the need for courseware for both commercial organizations as well as for governmental organizations, there was a shortage of skilled courseware development experts at the end of the nineteen nineties, in spite of many new formal education possibilities. At the same time, because the much-needed integration of the various disciplines within educational project teams is finally taking place, the quality of courseware has also improved, partly due to more sophisticated hardware to support multimedia. In other words, although we have by now gained valuable insights into just what is required to produce high-quality courseware, we observe that it is often very difficult to find all required disciplines to realize this high quality. This dilemma means that we must investigate ways of improving the productivity of existing courseware development teams. This thesis investigates ways to do this by looking at ways to facilitate courseware development teams in what they need during the development process. We will elaborate on this in consecutive chapters.

We have now provided a brief overview of the history of research on developing courseware. Any attempt to also provide a look into the near future of courseware development is much more dangerous, especially because of the ever-faster pace of technological developments. However, current research efforts that at least look promising, include the following:

- Embedded courseware: for example, why not learn about a car in the real environment: the car itself. The Personal Computer is no longer the only feasible carrier of courseware;
- New methods of input and output in relationship to human-computer interaction: for example, advanced speech recognition and virtual reality may further enhance the engagement and immersion factor of courseware.
- Courseware on the Internet: with the ever increasing power of Internet cables, more sophisticated methods of distance learning now become possible, for example virtual classrooms with learners attending from all over the world; video broadcasting over the Internet, and case-based group-wise problem solving.

1.3. Software process improvement in courseware development

The development of software in general has had to cope with enormous difficulties ever since it existed on a serious scale; numerous large software projects that failed illustrate this [BOE1988; GIB1994; BRO1995]. Since courseware development is a special case of software development (because of its multidisciplinary character and the highly interactive nature of the product, as argued in the previous sections), the need for software process improvement has been reported for courseware development as well (see for example [CHA1980; HEN1991; LAI1994; MYE1994]). When looking at this literature, we observe that much of the research on improving courseware focuses on how to improve the quality of the *product* for the *users* (see for example [CLA1994]). This seems obvious because, in the end, it is the users who experience, and then judge the quality (usability, usefulness) of what they use. Such a focus consists of investigations into "how did it work for the user" on the one hand, and "how should it work for the user" on the other hand. The success of a courseware product is often measured along these lines.

The research focus of courseware has been much less on improving aspects of the courseware development process itself. Courseware development teams consist of experts with roles, tasks, and responsibilities. They work according to procedures and guidelines of a method or methodology. There are development methodologies for educational systems (see for example [GER1987; HAR1988; VDM1995] and chapter 2 for a discussion of existing methodologies), but in-depth evaluations of these methodologies in the field are hard to find [VBE1992B, LAU1993, VAA1994] and success stories[3] are sparse (see [FAL1995, ALB1996] for explicit descriptions of success stories). In this regard we observe, however, that the courseware industry has always been only a small subset of the very large software engineering industry[4]; therefore, it may be that software process improvement research has consequently and inadvertently undervalued the importance of productivity improvement in courseware development. We will now briefly consider research efforts in software process improvement in general software development; we provide a more detailed analysis in chapter 2.

In the general software engineering research field, existing methodologies for software development [BOE1994; BRO1995;] gradually evolved into advanced methods for improving the quality of the development process and increasing productivity [CAR1990; GIL1988; JON1997; HUM1997; PFL1996, STA1997; WER1994]. Most of these efforts were of a highly quantitative character with much emphasis on statistical measurement and validation [AFT1988; CON1986; NEJ1995]. Complex statistical formulas were devised to be able to better predict schedules for meeting time, budget, and estimated effort required, all with the purpose of increasing software productivity and decreasing occurrence of defects [GAR1996; AFT1988; GRE1996]. In the "traditional" software industry, some of these methods certainly produced good results, especially on administrative and industry-type information systems [PUT1996]. For educational software, however, achieving software process improvement in this way is very difficult. This is partly because the most important measurement variable in traditional software, namely application size, is not easily measurable because of the various types of other media involved such as graphics, audio, and video. These often make up a large amount of the development effort and of the resulting product size; in modern authoring systems, lines of code are not used at all any more. Consequently, statistical efforts for software

[3] Especially success stories which professional multimedia development teams can learn from are hard to find.
[4] And therefore, that software process improvement has gotten a disproportional small amount of attention in educational multimedia. This is, however, only an assumption, not backed up by investigation.

process improvement in courseware development have not resulted as yet in proven improvements [VAA1996; VAA1998B].

An example of a successful recent development in software process improvement is the Capability Maturity Model (CMM), developed by Watts-Humphrey [HUM1995], and improved by Paulk and Curtis. With the CMM, the focus of software development is on improving the "maturity level" of an entire organization involved in software development. The "maturity level" is about getting as many project variables as possible under control; we elaborate on this in chapter 2. Software improvement frameworks such as the CMM focus more on the quality of the development process of software than traditional software development methodologies [JOH2000, KEE2000]. This allows software development teams to continually work on becoming more professional, or "mature". Contrary to development methodologies for educational multimedia, the software process improvement methodologies for "general" software engineering have been extensively evaluated and reported upon (see for example [GIB1994; HSI1993; NEJ1995; PFL1996]). Software development methodologies such as RAD (Rapid Application Development) and DSDM (Dynamic Systems Software Development) increasingly take software process improvement explicitly into account [STA1997].

Since software process improvement is becoming increasingly successful (see [GRE1996] for success stories), a software process improvement approach that is similar to that in general software development may also work for courseware development. This is not to say that existing methods and frameworks can be directly applied, but the main principles may be re-usable. Therefore, it now seems appropriate to investigate approaches for software process improvement in courseware development. This is the main research focus of the thesis.

1.4. Direction and relevance of research

In the preceding three sections, we have described several viewpoints on educational systems, and courseware in particular. We have chosen to focus on courseware with a large multimedia component. We have observed that for courseware development, our choice of focus implies the inclusion of a number of different disciplines, although we have not yet discussed each of these in detail. We have argued that partly due to the multidisciplinary nature of developing educational systems, and partly because of the special requirements that the use of multimedia sets on courseware, the development of courseware is a special, often more complex case of software development. As a result, we have observed that when realizing courseware, developers face issues that are very specific to this type of software development, although at this point we have not yet discussed in detail just what these issues are.

We have also observed that existing research on improving the quality of courseware mostly deals with improving the quality of the product, and is less focused on improving the process of developing courseware. We have therefore chosen to focus our research to improving the quality of the process of developing courseware. Finally, we have recollected that information systems development and software engineering in particular are known to have benefited from software process improvement research, and we propose that this is at least partly applicable to educational software development as well. In other words, the direction of our research is to investigate possibilities for process improvement in courseware development.

At this point we do not yet know in exactly what way existing literature about software process improvement can aid in our research, but we have observed that software process improvement is closely linked to a fuzzy term which we commonly refer to as process "maturity", a term that we will have to make quantifiable in order to draw valid conclusions. We will quantify the term "maturity", using existing literature, in chapter 2. We will then investigate characteristics of maturity of courseware development, and we will propose ways for courseware development teams to improve this maturity. This will culminate, in chapter 3, in a formal research hypothesis, which is based on the research questions that we posed in section 1.1. We will test the hypothesis by conducting a case study and setting up a formal experiment; this experiment will include statistically valid measurements on the quantified aspects of process maturity in a real-life situation as opposed to an artificial setting.

Since we are addressing a problem that includes elements from multiple research areas (educational systems, information systems development, multimedia, human-computer interaction, software process improvement), these also have a role in this research project. We therefore provide a concise analysis of what aspects of those "related" areas of research are taken into account for this thesis; this is described in section 2.4.

Courseware researchers and developers around the world are currently faced with little accepted literature on improving the quality of the process of developing courseware. Existing methodologies for developing courseware usually focus on product quality but not on process quality. With this thesis we therefore hope to create on the one hand an increased awareness of the importance of courseware development process improvement, and on the other hand valuable suggestions to not only think about this process improvement, but to actually realize it in real-life courseware projects.

1.5. Reading guide

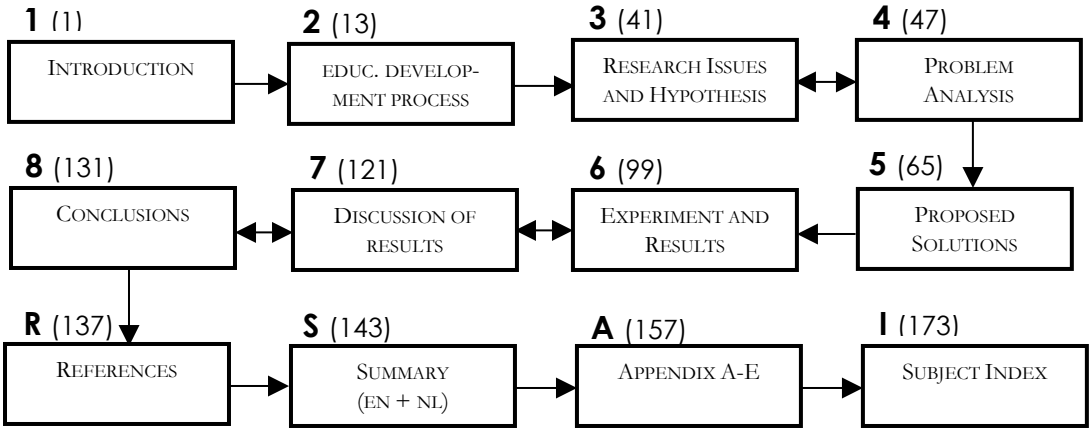
This section provides an overview of where the various parts of the research are located. The research is described in chapters 1 through to 8, which flow as follows.

- Chapter 1 attempts to briefly define the context of the research study, including a viewpoint on educational software in general. By reading chapter 1, you should be able to determine if you will find in this thesis what you are looking for.
- Chapter 2 discusses the educational development process, including strategies for development, required disciplines and roles, and issues that courseware development teams face while working. The related areas of research that are relevant to this thesis are also investigated, though less thoroughly than educational software. References to more in-depth publications are included.
- Chapter 3 discusses the research strategy and presents the research hypothesis. Here, we also include necessary assumptions, constraints and requirements of our research project. In other words, we define the research boundaries.
- In chapter 4, taking the hypothesis as starting point, we analyze the courseware development process problems in an empirical situation, and we identify the direction of the solution that we will present and evaluate in the thesis.

- Chapter 5 describes the solution we propose to address the most stringent problems identified in chapter 4. It presents the implementation of our approach to achieving software process improvement in courseware development.
- In chapter 6 we describe the pilot study and the experiment that we carried out to determine the degree in which our solutions helped solve the problems identified in chapter 4. The results that are most relevant to this thesis and to the project teams involved in this thesis are presented.
- In chapter 7 we discuss the outcome of the experiment, what it implies for courseware development project teams, what is means for courseware process improvement in general, and to which degree the results are generally applicable.
- In chapter 8 we present the conclusions of the research project, including recommendations for realizing courseware process improvement, plus an overview of issues that require more research.

Those who want to find more information on one specific subject of the thesis are referred to the appendices. The thesis contains a summary in both English and Dutch, plus a list of abbreviations that we use, and an alphabetically ordered subject index. Those who merely need to get a quick overview of the research scope, the experiment and the results, are referred to the summary section.

The sequence of chapters described above is schematically summarized as follows. The first number above each box denotes the chapter number, while the small number in brackets directly following it, denotes the page on which that particular chapter starts. Use the diagram as a convenient reference source for quickly finding the section you are looking for.



2. The courseware development process

In chapter 1 we have defined the main focus of the research, namely process improvement in courseware development. In this chapter we describe the courseware development process more in-depth. We analyze currently existing courseware development methods, and we identify aspects that we feel are currently undervalued, and that this thesis may contribute to. In other words, in this chapter we will identify the focus of our research *activities*. We will structure these activities in the shape of the research hypothesis that we present in chapter 3.

2.1. Strategies for developing courseware

In this section we analyze existing strategies for developing courseware, and we will identify aspects of the development process that we feel are currently (inadvertently) underrated. Building on this analysis, we then propose a list of research activities geared towards addressing these aspects.

The strategy chosen to develop courseware depends, among other considerations, on the context in which the application will be used, and the resources that are available to develop the application. The most important resources are the experts with their educational and multimedia expertise. In other words, the available expertise is an important factor in the quality of the eventual courseware product. The expertise may come from one individual, or the expertise of many experts may be combined. Let us consider the entire spectrum that lies in between (based on [MOO1987]). The most basic scenario consists of one person, in the role of the teacher, who develops the courseware alone, for one or more learners. In this case, the teacher must be knowledgeable in all areas mentioned in Figure 3: the user interface component (including all types of media used), the subject matter component, the pedagogical component, and the student model component. This is the *individual* approach. Since teachers who are experts in all of these areas are very hard to find, the resulting courseware will usually not be of a high quality level (although it may be sufficient for the student to achieve the learning goals, in which case both the courseware product and the development process may still be regarded as a success). In a more elaborate scenario we see an organized approach, in which several experts provide the required disciplines. In this case, there is often a dedicated project manager who is also the chief designer of the courseware. This is called the *team* approach. In general we may say that the team approach has a better chance of delivering high-quality courseware products.

In the most elaborate scenario we observe a large project organization where all required disciplines are available on an expert level. The courseware project may be divided into sub-projects. For each of the components in Figure 3, we may have multiple experts for each discipline on the project team, each bringing in his or her specific field of expertise. We will discuss the disciplines required in more detail later. In the most elaborate scenario, the development process becomes an extensive list of objectives to be met for various phases, each with tasks, activities, responsibilities, procedures, quality assurance indicators, and iterations to other phases. The role of project management is even more crucial here than it already is with the team approach. This is the *organizational* approach.

Regardless of the scenario that is chosen for courseware development, there are objectives that any courseware development project must meet; these are often modeled in the form of project phases. Van der Mast has conducted a survey of literature that identifies objectives that should be met for

high quality courseware development [VDM1995A]. The survey resulted in a description of the life cycle of educational software, from the initial problem analysis until the final deployment of the product. The top-level phases are (1) problem analysis, (2) educational design, (3) functional design, (4) design of other materials, (5) realization, (6) evaluation, (7) production of the final product, (8) implementation in the organization, and (9) deployment. Especially the market analysis, feasibility study and implementation are emphasized, because these are reported in the literature as key failure factors. In particular, if the problem analysis results in a project "go", the final product is still likely to fail if the actual implementation of the courseware in the organization (that is, the activities *after* the final product is complete) is neglected. We see that the phasing that is reported here generally agrees with many existing methodologies for information systems development (such as SDM). In other words, educational systems are information systems too; the main difference lies within the components of Figure 3 that are embedded in this phasing, and the implications this has for the required disciplines that must be involved.

Some researchers take a strictly educational (teacher-learner) approach to development strategies for courseware; others focus more on the media aspect. An example of the latter is the development strategy reported by England and Finney [ENG1996], which is more targeted towards inclusion of multimedia in the product and less on educational aspects. They call this the "client-centered" multimedia development cycle for courseware. The top-level phases are (1) Initiation and definition, (2) production, and (3) completion. This rather brief phasing does look very general at first sight. However, when zooming in, the researchers do propose some media-specific topics. They have a strong focus on agreement between the development team and the customer on the nature of the product, the multidisciplinary way of working, on the content and the media to be included, and on various ways of testing the product. More than most other research reports on developing educational multimedia, England and Finney advise on a tight control of activities by the project manager, and a formal administration of communication and agreements between the customer and the project team.

Vaughan provides another example of a media-focused development methodology, initially conceived by Blum [VAU1994]. Vaughan defines seven top-level phases, in a manner analogous to the method by Van der Mast, even though the focus is more on multimedia, similar to the England/Finney approach. Vaughan includes an environment analysis, the identification of instructional goals, a cognitive model, and makes extensive use of various forms of prototyping (including paper-prototyping). On the highest level, the Vaughan/Blum method for developing educational multimedia shows striking similarities with "traditional" software development methods such as SDM. In fact, most strategies for developing educational software are derived from "traditional" software development methods. Notable courseware-specific differences are the way in which attention is paid to the user interface, usability in general, the need for various media, the didactical or pedagogical design (the learning goals), and the nature of the acceptance test by the target audience. Most of these differences are due to the highly interactive nature of educational multimedia.

Another strategy for developing courseware that is worth mentioning in the scope of this thesis is by Hartemink *et al.* at [HAR1988]. This courseware development methodology is called PROMISE. Contrary to previously mentioned strategies for developing educational multimedia, here we provide a somewhat longer list of top-level phases, because we will be using this strategy later, in the experiment described in chapter 6.

1. **Arouse Interest:** present educational software development approach to customer.
2. **Make bid and contract:** assess quality requirements; plan the phases; sketch the product; define the way of working; agreements on guarantees; make the project bid; sign the contract; define sign-off documents.
3. **Staff the project:** select the project manager; select the project team (including all required disciplines).
4. **Set up project:** ensure quality guarantees; generate the project handbook; agreement on analysis phase approach; organize kick-off meeting; organize the way of working.
5. **Analysis:** administer analysis phase; ensure quality guarantees; analyze the problem.
6. **Design:** manage the design phase; design the educational software application; create global design; make detailed design; realize and test prototype; design didactical documents; design written communication material; design deployment of material.
7. **Realization:** manage the realization phase; realize product; realize quick reference cards (if required); realize didactical documents; realize written communication material.
8. **Test and accept:** manage test and acceptance; guarantee quality agreements; test and get acceptance.
9. **Implementation and product evaluation:** manage the implementation and product evaluation phase; implement the educational system; evaluate the educational system.
10. **Assess project:** evaluate the way of working.
11. **Evaluate project:** evaluate quality agreements; evaluate the entire project.
12. **Archive project:** documentation gathering; close the project; collect new templates and best practices.

As with other courseware development methodologies, we again see that PROMISE builds on general information systems development theory. More additional methods for developing educational software have been devised throughout the years. We mention the following: Instructional Management Presentation System (IMPS) by Godfrey and Sterling; Environment for Developing and Using Courseware (EDUC) by De Diana; the Gery Method by Gery; the Shiva method by Elsom-Cook; and the PRINT method by Van der Mast et al. To extensively elaborate on each of these methods is beyond the scope of this thesis. See [VDM1995A] for a comprehensive summary of each of these methods. Here, we suffice to note that each method has characteristics that are missing in others, or only partly present. Moreover, all of these methods are always influenced by the organization that implements it.

We observe that although in all development methodologies that we discussed education and multimedia specific issues are addressed, there is little advice on how to improve the quality of the way of working of the courseware development teams; the focus is mostly on activities to improve the quality of the final product.

All development strategies that we have surveyed so far are temporal-based, that is, they assume an ordered sequence of activities to be carried out in order to end up with a successful courseware product. However, we can also construct a component-based look at developing courseware. We then look at the various "building blocks" that need to be in place in to create a successful courseware product. Van der Mast proposes such an approach [VDM1995A]. Instead of listing objectives to be met in the development life cycle by ordering these into project phases, Van der Mast defines three perspectives to order the various components that must be taken into account

when modeling educational software [VDM1995A]. These are then conveniently grouped into the "why", the "what", and the "how" (a general principle). The organizational context is leading for the teaching and learning goals and the general requirements of the product (the *why*). This is a high level of abstraction that allows for development methods that are flexible and subtle. The actual pedagogic and didactic modeling constitutes the *what*. Here, the learning goals, rules, instructional strategies, student tasks and media usage are defined. In the *how* perspective, the interaction with the learner is defined in terms of interaction, engagement, and persuasiveness. On this level, semantic (e.g., presentation of learning objectives, giving feedback on results), syntactic (e.g., interaction styles: multiple choice, form fill in, direct manipulation), and lexical (e.g., layout design, functional areas on screen, use of colors and fonts) aspects of the interaction are discussed. The three perspectives are visualized in Figure 4.

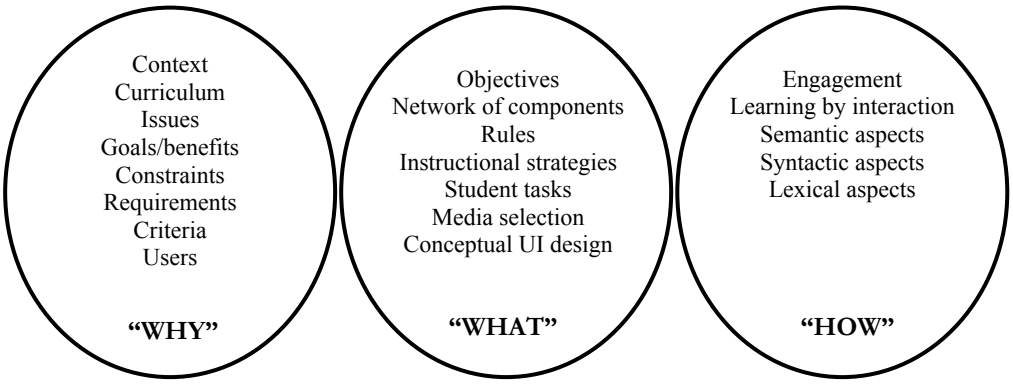


Figure 4. *Three perspectives on the development of educational systems.*

Perhaps more clearly than the time-based development strategies mentioned earlier, this alternative model clearly illustrates the issues specific to educational multimedia that arise when developing courseware. Many of the components above are absent in traditional software development. Therefore, Figure 4 again illustrates the special character of educational multimedia within the general software development world.

Some researchers have attempted to create a high-level abstract framework for developing educational multimedia that should combine the more concrete (and different) strategies defined above. One such effort worth mentioning is the Common Training Architecture by Verreck and Weges [VER1994]. They present their framework for organizing a courseware project in the scope of a European effort to reach agreements (they call it harmonization) in standards on interoperability and portability of courseware. However, our survey so far indicates that the field of educational multimedia is probably too varied and diffuse to be fitted into a "one framework suits all" solution. A similar conclusion has earlier been drawn for general information systems development.

Disciplines required

The list of disciplines required for a courseware development project is largely determined by its two characteristics *educational* and *multimedia*. Of course, the scope of the project also determines the range of disciplines required. More complex projects will generally require more disciplines on a higher expertise level.

We now look more in-depth at the list of disciplines that are required for professional courseware development. We will assume here a moderately complex project. When considering the keywords *educational* and *multimedia*, we can easily identify the following required disciplines:

- Project management
- Information analysis and consultancy
- Interaction design (manage the interaction of the user with the actual application, including navigational design)
- User Interface design (manage the interaction of the user with the computer, including layout design)
- Didactical / pedagogic design (the strategy of teaching and how this will be presented)
- Subject matter expertise (domain-dependent, usually someone from the customer's organization)
- Modeling / functional design (just what will the application be able to do and offer)
- Visual design / graphics production
- Audio design / sound production
- Development (software engineering, including programming, database design, testing and evaluation)

England and Finney list the following disciplines required for multimedia courseware [ENG1996]:

- Project management
- Video production
- Audio production
- Stills/graphics production
- Database design
- Modeling / functional design
- Development (software engineering)

When we combine the two lists and eliminate the double entries, we see that for full-scale courseware development at least ten different disciplines are required: project management, pedagogic and didactical design, interaction design, user interface design, modeling (functional and technical), development, graphics, audio, video, and subject matter expertise. Although many courseware products are not full-scale and therefore not all of these disciplines are always required, the project manager faces a great challenge in overseeing all disciplines and making sure the disciplines all co-operate in an efficient manner. We emphasize here that a discipline need not be the same as a person. Some persons are proficient in multiple disciplines; often, however, one discipline will have to be staffed by multiple experts. Later in this section, we will discuss the various roles in the educational multimedia project team in detail.

Tools and other aids

Many technical aids have been created throughout the years to support courseware developers with easy-to-use tools to significantly speed up the process of developing courseware, especially regarding inclusion of multimedia. Initially, these tools were geared towards programmers, but later on, the trend was to support less technical developers as well, with tools that allow developers to

visually construct a multimedia application (called fourth-generation development tools or 4GL for short). At the same time, tools for other disciplines became available as well. Here, we list the most important tools available, for a selection of the disciplines involved.

- *Graphic designer*
  - **CorelDraw**. This is a graphics tool that was originally geared towards the Desktop Publishing (DTP) industry; it allows for pixel-based graphics to be integrated with vector-oriented images and vector-based fonts. Not easy to learn for non-graphical experts, this package is mostly used by experts in their field.
  - **PaintShop Pro**. This is a relatively simple drawing application that allows for basic graphics to be produced. Layering of graphics is not well supported, and many filters for rendering graphics are not included. However, for producing quick sketches, it is a tool that is often used by a variety of team members.
  - **Photoshop**. By many dubbed "the" standard tool in graphics production, Photoshop supports virtually all graphics functionality required by professional graphics creators, including sophisticated color management, layering, and a large set of graphics filters. However, it remains a pixel-based tool, primarily targeted towards on-screen graphics production.
  - **Illustrator**. For broader use than its nephew, Photoshop, the Illustrator package is a competitor to CorelDraw and by many regarded as more professional. Used for both desktop publishing as well as on-screen graphics production, this is a sophisticated tool that is not easy to master quickly, and is therefore used by experts in their field only.

- *Functional and interaction designer*
  - **Frontpage**. An immensely popular tool for web-site development, Frontpage is often used to quickly construct a prototype user interface of an (educational) application. It allows for the user to quickly place graphics and buttons on specific parts of a page, while keeping a fair amount of control over the navigation of the various pages. Although any tool is as good as the person that uses it, many interaction designers use Frontpage as a quick alternative to paper sketching.
  - **DreamWeaver**. Sometimes dubbed as the "decent" alternative to Frontpage, DreamWeaver is a somewhat more sophisticated environment for web-development that, similar to Frontpage, allows for non-technical users to quickly construct screens with navigation items.
  - **Rational Rose**. More targeted toward functional and technical designers, this expensive package allows for the construction of flow diagrams for interaction design, and is capable of automatically generating a framework of code for the application, while also keeping an abstract, high-level view of the interaction between the various components and the end user. Rational Rose also allows for various "usage scenarios" to be constructed.

- *Video developer*
  - **Premiere**. Probably one of the most widely used tools for digital video construction for software applications, Premiere allows for detailed editing of video footage, layering and transforming scenes, and adding audio (music) to scenes. A large number of filters for enhancing various aspects is included. There are much more sophisticated tools for

video development, but these are usually not used for educational multimedia software development, but rather in the traditional media industry.

- *Audio developer*
  - **Cubase VST**. Supporting both the audio midi format as well as wave-based audio formats, Cubase is used by a large group of professional music composers, arrangers, producers and engineers to create musical scores for multimedia applications. Cubase includes a customizable number of music channels ("layers") which can be played back simultaneously and edited to minute perfection. Since Cubase is often regarded as being "the" standard for audio development, we forego mentioning its many competitors here.

- *Script level application developer*
  - **Toolbook**. A relatively easy-to-learn multimedia development tool, Toolbook facilitates a developer in placing all sorts of media elements (text, graphics, video, audio) on screen and provides an easy, high-level scripting language to allow for interaction between these elements based on user input. Interaction design modeling is not supported well.
  - **HyperCard**. In many ways, HyperCard is the Apple equivalent to Toolbook, in that it is based upon a high-level script language that allows the developer to add complex interaction between all sorts of media elements that are placed in a visual way on-screen. Initially only in black-and-white, later versions of HyperCard were used extensively to construct educational multimedia applications.
  - **Director**. This is one of the more sophisticated educational multimedia development environments. Based on the metaphor of the world of "theatre" or "movies", the developer is presented with a "performing stage" and a timeline, which are used as placeholders to structure the application to be developed. Scenarios are defined along the time line, and a simple scripting language called Lingo is used to provide for interaction between the application's elements and the user. Later versions of Director support more sophisticated levels of educational multimedia applications.
  - **CT**. An authoring system including its own scripting language, cT has been used extensively on universities in the U.S.A. and Europe. It is based on the creators' perception that courseware is best produced by one individual. This authoring system supports the production phase only, supporting the creation of rapid prototypes. It is almost entirely language based, the scripting language dating back as far as 1967.
  - **TenCore**. Like cT, this tool heavily depends on the use of a scripting language to produce courseware. Because of the large number of tags and commands available in the TenCore language, even experienced programmers are required to put a lot of effort into getting to know this language. TenCore does not support rapid prototyping and is mainly used for the educational design phase and realization phase.
  - **Authorware**. When speaking of script-level environments, Authorware is probably the most advanced and most widely used development environment. This environment is icon-based, in which each icon represents a chunk of functionality of the final application. Diagrams can be constructed to define the flow of interaction between the various building blocks; most types of media can be inserted as well, allowing for rich multimedia applications to be constructed. Authorware is probably one of the most

expensive scripting-level development tools available, but probably also has the best track record.

- *Coding level application developer*
  - **Visual Basic**. Of the "lower-level" development environments, this is probably the one that comes closest to the Script level application development environments mentioned above.
  - **Visual C++**. This environment is used by the most technical programmers and are of no use to other disciplines, simply because the language is too complex to master for most. Because the code of Visual C++ is compiled on the development environment, this allows in potential for the best performing educational multimedia applications; however, rapid prototyping with inclusion of a lot of media is not easy and requires a lot of experience.
  - **Java**. Platform-independent in theory, this development environment has until now not been used extensively for educational multimedia development, although in potential the possibilities are virtually endless. Slow performance and intricate coding requirements have up to now hindered the global success of Java, but it is still promising. Used by technical programmers only.
  - **Visual Interdev**. Programmers use this tool mostly to generate web-based applications. Use of other types of media is hardly supported, although database-driven applications are supported well.

Perspectives on the courseware development lifecycle

We have now provided an overview of existing strategies for developing courseware, the disciplines involved, and available tools that aid the various experts in their activities. We have observed that existing courseware development methodologies are primarily concerned with improving the quality of the product, and are less concerned with the quality of the development process itself. Since this thesis investigates the working process of the development team, we now briefly investigate research on the process of the development lifecycle in courseware (courseware process improvement); we will then zoom in on the various roles that are involved in this process.

An abstract, high-level perspective on the development process is presented by Henderson [HEN1991]. He provides a process lifecycle on a high level of abstraction that can serve as a checklist for the development strategies defined earlier in this section, namely to verify whether or not these steps fully cover the Henderson cycle as shown in Figure 5. The central component in this cycle is the *learning process* of the student. This is not to say that the quality of the courseware development process is not important in the Henderson cycle, but the focus is on the learning process. The observations on the effectiveness of the learning process are recorded and analyzed. Based on the patterns that are (hopefully) recognized in the analysis, the design of the application is improved, and a new implementation is executed from improved specifications. This suggests a "waterfall"-like way of working, which need not be the case in practice.

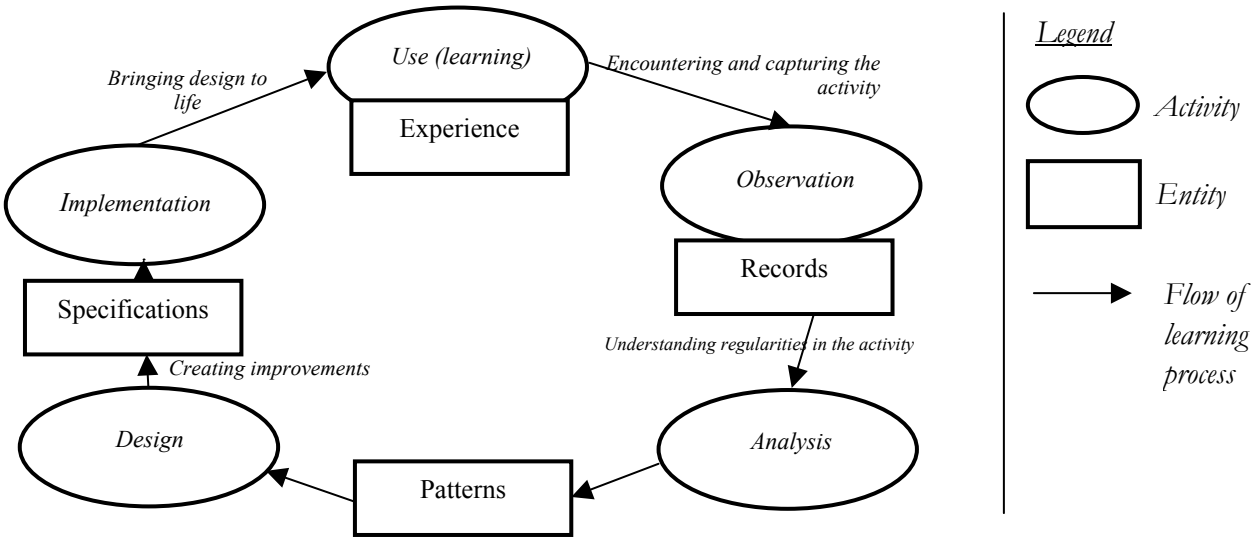


Figure 5. *A learning process perspective on educational software development, by Henderson* [HEN1991]

An alternative way to model the courseware development process is by using the traditional control model, later investigated by Blumenthal[BLU1968] and by De Leeuw [DEL1994] and sometimes called the Information System – Real System model [SOL1988]. This generally applicable model views a process as consisting of two systems, say A and B, where system A controls (not exclusively) system B; this is shown in Figure 6. The two systems constantly influence each other.

Using the traditional control model, we can describe the process of developing courseware up to and including the implementation as depicted in Figure 6 [VAA1996; VAA1998A]. In this sense, it is an applied representation of the Henderson cycle; here, however, the focus is more on the courseware development process itself, and less on the learning effect of the learners. In Figure 6, system A consists of the handbooks that provide procedures, guidance and steering to the project process, which is system B. The customer needs are translated into a project plan, which both serve as input to system A and system B, while the actual educational multimedia application and the new expertise by team members are examples of outputs of system B. The final courseware product is delivered to the customer, and the newly created expertise and experiences usually stay in the experts' heads. In considering this final remark, we see here a link to the research field of knowledge management; this link will be addressed in chapter 4, the problem analysis.

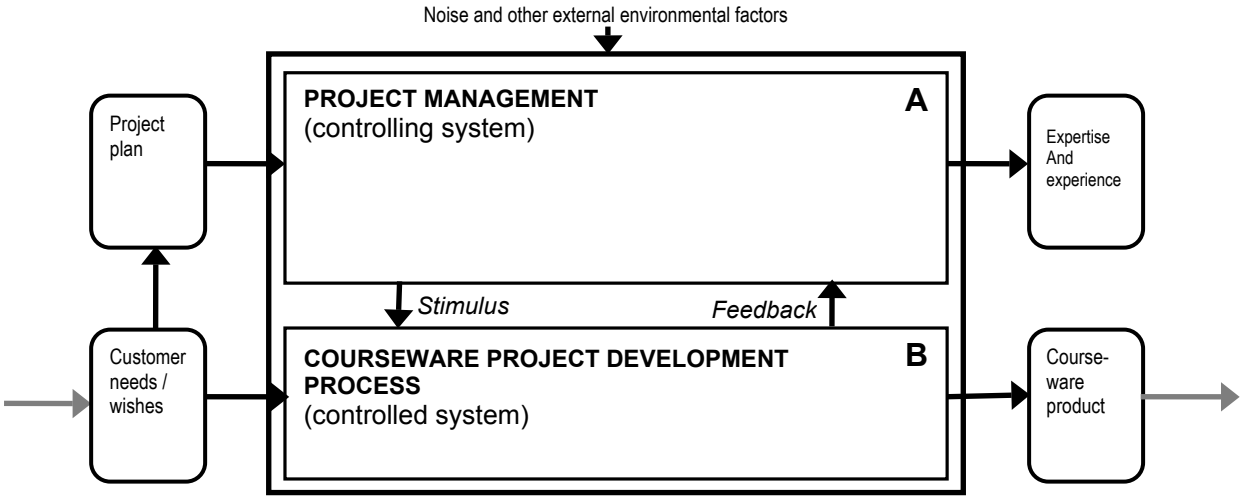


Figure 6. *The courseware development process mapped to the traditional control model.*

In the instance of this research project, the educational multimedia project development process is system B, and the project management is defined as system A. It is emphasized here that project management is seen as a group process and responsibility, not just a task for someone fulfilling the role of project manager. In principle, this model is independent of the development methodology used for the educational multimedia application.

An important problem of the multidisciplinary project team that is involved in this process is one of maturity: how can the team reach a level on which they can prove that they have the entire courseware development project process well under control? Please note here that we once again touch on the fuzzy term "maturity" which we already encountered in chapter 1. We must make the term quantifiable if we are to use it in the main objective of this thesis, namely to improve the maturity of the courseware development process. We will first investigate general characteristics of maturity in relationship to courseware development.

Maturity is about getting the project process under control. Manage the available interdisciplinary expertise. Be able to make higher quality project bids. Get the various disciplines to communicate effectively. Agree on the project tasks, responsibilities, activities and deliverables for each project phase and each project role, in other words, agree on the way of working of the multidisciplinary team. Measure the efficiency of the development process. Take actions based on these measurements to further improve the efficiency; in other words, to improve the overall maturity, on an operational level – the level that the project team itself works on. Having said this, we still have not defined the term "maturity" in a measurable way. We will look at this in-depth in section 2.4; we will first investigate just what such a multidisciplinary courseware development team looks like.

2.2. The project roles in the development process

The structure of the multidisciplinary project team involved in courseware production is becoming increasingly complex [ENG1996; ALB1996; VAA1996]. On the one hand, more and more people are becoming involved who are experts in their own field, but not in other fields (the *specialists*). On the other hand, there are also multimedia experts who know a bit about several disciplines, but are not really an expert in any one of them (the *generalists*). Both types of project members can be very useful to have [GER1987; LYN1993; TJO1991].

As for the required disciplines, at least project management, software engineering, graphic design, interaction design, didactical design and content matter expertise are required in courseware development projects [VAU1994; AND1990; VAA1998A]. The highly varied nature of these disciplines – and especially the consequences of getting these disciplines to co-operate effectively – is an important factor of the complexity involved in realizing courseware [GER1987; TJO1991; VDM1995A; VAA1996]. This complexity can lead to severe problems during the process of producing the courseware [VBE1992B; DEM1987; FAL1995; VAA1998B]. In other words, an important cause of the failure of educational multimedia projects lies in the complexity of managing a highly multidisciplinary team of experts [VDM1995A; VAA1996].

In the educational multimedia project team, various roles are required. The roles required may be derived from the list of disciplines required, as mentioned in section 2.1. We mention here that 'role' and 'discipline' are not synonyms: any given role in an educational multimedia project team may require skillfulness in more than one discipline. For example, a graphics artist should be knowledgeable about interaction design and usability in general, in addition to drawing techniques and technical possibilities and limitations.
In investigations about the roles required for courseware development, some researchers focus more on educational project teams, while others focus more on multimedia project teams; still, in all these lists, many of the same required disciplines are present. England and Finney [ENG1996] provide a short list and an elaborate list of roles required in a multimedia project. The short list suffices for the most basic of courseware projects and includes (1) a project manager who might also use his/her expertise from a media background to contribute to part of the project development (this role is then called the *managing designer*); (2) someone who will agree about the content and instructional design with the client and the relevant members of the team (3) someone to produce the computer graphics; (4) someone to program; and (5) someone to arrange and manage the audio and video production. Building on this, they then produce an elaborate list, based on their own experience. This list is presented in Table 1. In empirical situations, multimedia teams consist of a subset of these disciplines. The larger the multimedia project, the more disciplines are likely to be involved.

Table 1. *Multimedia roles according to England and Finney.*

Mgt/Administration	**Audio production**	**Database development**	**Design/documentation**
Project manager Project assistant General production assist. Secretarial support	Audio production mgr. Audio script writer Audio editor Voice-over artists Musicians	Data collection/mgt Database Integration/ development Indexer	Interaction designer Instructional designer Interface designer Interaction script writers Subject matter experts
Video production	**Stills/graphics production**	**Computing/Integration**	
Director; producer Camera; Lights; Sound Grips; Make-up Continuity; Logging Script writer Video graphics Offline/online editor Film/pic research	Production mgr Graphics production Picture researcher Animator; photographer Lighting; 3D modeller Digital graphics prod. Art director Illustrator/Artist Typographer	Programmer Software engineers Technical manager Network manager	

The list by England and Finney focuses on the inclusion of multimedia. They pay little attention to didactical aspects. Tasks such as writing instructional materials and choosing didactic methods (how to make students learn something) are not included in the roles that they list. There are, however,

researchers who focus more on the educational aspect. For example, Gery lists the following roles that are required in a courseware project, including the customer and the learner roles [GER1987]:

- *Project manager*. This role oversees the project from beginning to end. Gery, like other researchers, acknowledges that the project manager is the most critical role of all.
- *Program or Course sponsor*. Authorizes the program's development and provides the logistical, economic, and political support necessary to orchestrate all the variables and the individual involvements necessary for success.
- *Instructional designer*. Determines the strategies and techniques for imparting content to the learner.
- *Subject matter expert*. Consults on content for the training. He/she educates and provides input to the team on program content, including scope, complexity, and learning objectives.
- *Writer*. Composes the "script" of the training program. He/she translates design specifications into scripts for text and graphic screens to effect learning.
- *Editor*. Reviews the project for communication effectiveness. Reviews, alters, adapts, and refines scripts to conform to standards and to assure clarity from the learner point of view.
- *Authoring System specialist or Programmer*. Prepares executable code or runs the authoring system to create the actual program and consults on the capabilities and limitations of the computer system.
- *Data input or entry specialist*. Feeds the scripts into the authoring system.
- *Media expert*. Serves as a link to various presentation media that may be included in the program. Consults with designers and writers on media limitations, cost, and production requirements.
- *Graphics designer*. Designs the visual layout of the CBT presentation. This role participates in creation of screen display standards, including development of prototypes.
- *Technical systems expert*. In PC environments, the programmer typically performs this role; in mainframe environments, the role is frequently separated from the development team itself; it is located in a staff function charged with CBT system administration and operations.
- *Learner*. A representative of the intended audience who tests the effectiveness of the CBT program.
- *Production administrator*. Oversees the production and distribution of the media bearing the software.
- *CBT Administrator*. Makes the training program available to the learners who need it. This role develops and implements CBT administrative procedures.

We conclude our survey of roles involved with work by Vaughan [VAU94], who seems to combine the two previous lists above into one list containing only a few main roles, but many sub-roles for each role. The sub-roles may also be interpreted as disciplines, in this case. The following roles are defined:

1. The **project manager**. This role is at the center of the action. This role is responsible for overall development and implementation of a project, as well as for day-to-day operations. Budgets, schedules, creative sessions, time sheets, illness, invoices, and team dynamics – the project manager is the glue that holds everything together. Having said this, Vaughan, too, acknowledges that the project manager's role is the most crucial one.

2. The **multimedia designer**. This role is subdivided into Graphic designer, Illustrator, Animator, Image processing specialist, Instructional designer, interface designer, and Information designer. Therefore, this role actually consists of a number of different roles.
3. The **writer**. Multimedia writers do everything writers of linear media do, and more. They create character, action, and point of view – and also interactivity. They write proposals, script voice-overs, write text screens to deliver messages, and develop characters.
4. The **video specialist**. This role does more than just shoot and edit video. This role must understand the potentials and limitations of the medium, how these affect video production itself, and how to get most out of the video. This role must also understand interactivity and how it affects video.
5. The **audio specialist**. These include composers, audio engineers, and recording technicians. These are people who make a multimedia program come alive, designing and producing music, voice-over narrations, and sound effects.
6. The **multimedia programmer**. This role integrates all of the multimedia elements of a project into a seamless whole using an authoring system or programming language. Vaughan compares well-written code to the sheet music played by a well-rehearsed orchestra.

In Table 2 we summarize the lists of roles that we identified in this section. Whereas most of the surveyed lists focus on either educational or multimedia aspects, our list aims to present a better balance between educational disciplines and multimedia disciplines. The list is sorted on frequency of involvement of each role, where the first row is almost always involved, and the final row only occasionally. Please note that various roles may be combined into one person on some projects, especially the smaller ones. We also mention here that Table 2 is not a theoretical model, but merely a summary of our survey regarding required roles in courseware development.

Table 2. *Summary of roles in courseware development projects.*

Name of role	Primary responsibility
Project manager	Manage quality of project process and customer contacts.
Project administrator	Keep track of time, budget, effort, size, etc.
Human resource manager	Select required expertise from pool of employees.
Consultant	Chart customer's needs and wishes and advise on this.
Programmer	Make the solution work: build the software application
Content designer	Take content and mould this into a functional design
Graphic designer	Design and make visual elements of deliverables.
Didactical expert/designer	Design instruction strategy of desired solution.
Interaction designer	Make sure the solution works intuitively for users.
Audio engineer	Integrate required sounds / music into the software.
Audio artist (composer)	Make required sounds / music for the courseware.
Video engineer	Integrate required video/animations into courseware.
Visual artist	Make required video / animations for the courseware.

Our survey has, until now, focused on strategies for courseware development and the various roles required in the project team. We have observed that many courseware development strategies are aimed at improving the quality of the product and less on improving the quality of the development process itself; we have also observed that the roles and disciplines required for professional courseware development are of a highly varied nature. Taking the latter into account, let us now examine some of the typical project issues that arise in producing courseware.

2.3. Courseware project process issues

In this section we investigate issues that are likely to arise in a multidisciplinary project team while working on a courseware project. We do by no means pretend to be complete in our coverage; our goal at this point is to identify particularly difficult topics that we may be able to address in the scope of courseware development process improvement.

Customers
The courseware development teams that we described in the previous section work on courseware projects for a *customer* who commissioned the realization of a courseware product (perhaps as part of a larger educational system). Such customers are often departments of large organizations in sectors such as financial services, (heavy) industry, telecommunications, aviation, and pharmaceuticals, but also governmental institutes. They commission – by means of a request for project proposal (or tender) – various ICT service companies to produce an educational system, including courseware, to train their own staff. These products are usually realized in the shape of (hopefully well-defined) projects. Such a project is started if the customer accepts the project bid. Often, at this stage the customer does not have a clear idea of what exactly will be delivered in the end; a key failure factor of courseware projects is project teams that fail to clarify this sufficiently in collaboration with the customer [GER1987; VAU1994]. In other words, customer expectation management is important.

Transparency of expertise
At this point, when the project has barely even started, the person – or the group of persons – that has compiled the project bid, already encountered a range of problems. Hard-to-find resources have had to be allocated fictitiously to the project. Estimates have had to be made on how much effort it will require to construct various aspects of the product. Based on previous comparable projects, and possibly with the help of specialists, this bid-team generates estimates about time, cost and effort required for the educational project as good as they can.

Estimates are often in the form that we call *individual estimates* in this thesis (For Dutch readers: the official Dutch word is *ervaringscijfer*). We define an individual estimate as the quantitative answer of one expert to an experience question such as: "How much money, in US dollars, will it cost to produce a basic browsing CBT for the PC of about one hour?"[5] When a sufficiently large number of experts agree on the average of all individual answers (usually at least five or six, depending on the impact of the answer for a project), we say that we have found a *consensus estimate* (For Dutch readers: the official Dutch term is *kengetal*). Consensus estimates are extremely useful to have at hand when compiling project bids, much more so than individual estimates. However, consensus estimates are usually only sparsely available. That is, individuals approximately know them, but they are often not made explicit and documented, let alone integrated into the bid process. More often than not, the bid-team will enter a "wet-thumb" figure in the project bid (which often turns out to be incorrect later[6]).

[5] Please note that this implies that an experience question is always posed such that a quantitative answer can be given.
[6] A remarkable observation is that often the organization is not able to learn from this apparent lack of availability of consensus estimates.

Requirements management
Let us suppose that the bid is won. The project team then embarks on a risky journey. Many uncertainties must be addressed: "Do we know what the customer wants?" – "Does the customer know what the customer wants?" – "Will the content be delivered timely?" – "Who has decision authority to decide what?" – "Are there any hidden agendas in the customer organization regarding this project?" – "Does the customer have negative experiences with these kinds of projects?"

These are only a few of the many questions that arise [DEM1987; VAA1996; VAA1998A]. Often, using basic accepted project management instruments, a project plan and a functional requirements document are set up to address these questions (including an information analysis and sometimes a high level business analysis or a feasibility study), but these documents can never be complete from the start. Many important issues pop up only later, when the project is already well underway [BLO1995]. Therefore, much effort must be put into getting this document defined as clearly as possible, and this requires a lot of (sparsely available) time.

Focus of disciplines
In addition to the relationship between the project team and the customer as described above, the relationships between the various project team members is a factor that is also hard to manage. After all, every discipline that is involved usually has a different point of view on the design and realization of the final product. This is especially true of the relationship between artistic disciplines and the more technical disciplines such as software engineering [BOY1996; TJO1991]. In potential this is a great advantage (because the customer's needs are being looked at from as many points of view as possible), but in practice it frequently leads to misunderstandings, ineffective collaborative work and irritations [VAA1996; DEM1987]. The project manager must be able to effectively communicate with all the disciplines, and must act as the "glue" that makes the project team work as a whole [TJO1991; GER1987; ENG1996]. (This is why some researchers define the role of "managing designer": a project manager who is also the chief brain behind the solution being designed.) In this respect, it is useful for the team and especially the project manager to know what kinds of problems might arise, and how to handle each kind of problem. Successfully addressing these topics contributes to courseware development process improvement.

When looking at the courseware development process itself, we have now briefly identified a number of topics that can potentially cause difficulties: (1) customer expectation management, (2) the quality of estimates, and (3) the quality of the interdisciplinary communication and collaboration. If we can facilitate courseware development project teams regarding these topics, this may contribute to courseware process improvement. This forms the basis of the direction of our research, which we will elaborate upon in chapter 3.

2.4. Related areas of research

Research on courseware process improvement must include a survey of the related areas of research. This is especially true because there is little accepted research available on process improvement in courseware development. On the related areas of research, however, much published research is available. We present here a brief overview of each related area of interest (elaborate coverage is beyond the scope of this thesis); for each research area we investigate what is relevant to this thesis, and we use existing research results where appropriate.

For the scope of this thesis, the most important related areas of research are (not sorted on relevance or importance) information systems development and software engineering; software process improvement; multimedia development; human-computer interaction, and knowledge management. Please note that the research fields that we describe below are not mutually exclusive: in most cases, some overlap can easily be identified. We assume here that the research field of educational systems has been surveyed sufficiently in chapters 1 and 2; for additional information we refer to the list of references on page 137.

Overview of information systems development and software engineering

The development of information systems is particularly complex because fuzzy, hard to define organizational needs must be mapped to a(n ICT-oriented) solution that should be unambiguous and well-defined. Activities such as a business analysis, a task analysis, a feasibility study and an information analysis, which are conducted in complex organizational environments, must be mapped to a clear design that is fit for implementation by software engineers and other disciplines. Initially, a "waterfall"-like method for information systems development was proposed. In the waterfall way of developing information systems, a set of phases are consecutively walked through, usually starting with a feasibility study and ending with a project evaluation. Later, because of requirements changes due to new insights at many points in time, the need was felt for an iterative way of developing information systems. In this case, some phases can loop back to earlier phases to improve what has already been done. Finally, the incremental way of developing information systems uses an iterative way of working to construct an information system from a set of smaller components. Especially rapid prototyping, the construction of a number of prototypes in a relatively short time span, has gotten much attention as an effective aid in iterative and incremental information systems development. Examples of information systems development methods that are now widely in use are SDM (Software Development Methodology), RAD (Rapid Application Development), IE (Information Engineering), DSDM [STA1997], and YOURDON.

The deliverables of an information system development project comprise more than just software. We must realize here that software is not always by definition required to solve an organizational problem, the goal of information systems development. Deliverables can also include procedures for new ways of working, and new definitions of organizational responsibilities and activities. However, a software component is often an important part of an information system and becomes still more important with the ubiquitous use of computers. The development of the software component is called Software Engineering. The "engineering" of software includes a range of subtopics, including (extract from [BRO1995]) the theoretical foundations of computer science, the syntax and semantics of programming languages, problem solving algorithms and their mathematical background, functionality and data testing, and metrics with regard to quality assurance of problem solving activities.

Programming languages are often divided into "generations". We will limit our survey starting from the third level programming languages, and we will not concern ourselves with, for example, low-level assembly programming, since this outside the scope of this thesis. The scale of increasing "generations" may be seen as continuing steps in ultimately achieving a natural-language approximation in programming languages. Fourth-level programming languages (4GL) generally approximate our natural language closer than do third-level generation programming languages (3GL). Notable "classic" 3GL programming languages include Pascal, Modula, Ada, C, and Visual Basic (see [ref] for more information). Notable 4GL-programming languages, sometimes called

scripting languages, are Visual Basic Script, HyperCard Script, and Lingo. In principle, an algorithm for solving a problem is programming language-independent, but some programming languages lend themselves better for solving a particular type of problem. Also, there are usually multiple algorithm candidates to solve a given problem.

Especially quality assurance is an important topic in software engineering. In the case of a single software engineer who aims to provide a solution for a relatively simple problem, a standard set of software tests may be sufficient to assure a certain minimum level of quality. The software engineer is always up-to-date of his/her own coding statements and the way in which the data is modeled. Things become exponentially more complicated when more than one software engineer works on a solution for a given problem. Even with a programming team that is as small as two software engineers, agreements must be made on the problem solving approach, coding standards, data modeling standards, documentation standards, test procedures, and quality levels. This requires a specialized form of project management and a high level of formal interpersonal communication. As computers became ever more powerful in the nineteen sixties, large corporations such as IBM had software development teams consisting of dozens of software engineers for large organizations such as the NASA and the American military administration.

The great complexity of both the problems to be solved (e.g., making a space vessel flight a success) and the workability of these large development teams has been the cause of many software project failures, which eventually led to a poor reputation of the software engineering field in general in the nineteen seventies [GIB1994; BRO1995]. As software engineering researchers and practitioners realized the challenges in quality assurance, many efforts for improving the maturity of the software development process were initiated (see also the subsection on software process improvement). Boehm [BOE1987] proposed a more mature way of software development, in an iterative "spiral" model. Brooks [BRO1995] argued that there is a certain optimum size to software development team staffing: when exceeding that size, the overhead caused by *m-to-n* way communication becomes larger than the productivity of the team itself. He calls this the "mythical man-month", and this has been one of the major insights in software engineering development that has contributed significantly to getting software quality under control.

Information systems development and software engineering in this thesis

In chapter 1, we have argued that educational systems development is a special case of information systems development. In spite of the special characteristics that add to the complexity of producing courseware, many properties of information systems development are usable (and used) for educational systems development as well. It is especially in the software engineering part of information systems development that difficulties occur. Initially, developing courseware was often done either from a strong software engineering point of view, or without incorporation of software engineering at all. The educational and multimedia characteristics of courseware result in many software engineering principles not being applicable any more. For example, the work by Conte *et al.* [CON1986] gives a rigorous approach to software engineering measurements, but this holds only for traditional types of software, not for courseware. This is chiefly because the wide range of formulas they present are chiefly based on the size of the application measured in Kilo-lines of Code (KLOC). Within educational multimedia applications, this is not a suitable indicator for application size (see also appendix E). In other words, in this thesis, in our investigation of process issues in courseware development, we aim to take into account general information systems

development research results, while focusing on educational and multimedia-specific topics in relation to software engineering.

Overview of Software process improvement

As the software engineering world seriously experienced the need for software productivity improvement, the research field of software process improvement emerged. At first, this focused mainly on mathematically oriented solutions for improving control over the process through complex metrics and statistics [CON1986; JON1997; MUS1987; SOF1995]. But over the years that followed, the focus shifted more to people-oriented aspects of the software process [GIL1988; HUM1997; LAI1994; WER1994]. Researchers found that successful software process improvement is a balanced combination of people management, technical cleverness, and organizational excellence. One of the most important – and increasingly applied – theories about software process improvement currently available is that of the Capability Maturity Model, conceived by Watts-Humphrey in the eighties [HUM1995]. This model offers a way of assigning a maturity number to an organization's software development process, and also suggests ways to reach a higher maturity number. This is done using a five-level scale for software development maturity, and by addressing key process areas that must be properly set up in order to reach the next higher level of maturity. The Capability Maturity Model has been successfully applied in diverse environments [JOH2000; KEE2000; LAI1994; NEJ1995; ROU1992]. An increasing number of companies focus on offering software process improvement solutions for large organizations. These organizations are in need of such solutions because a failure of a large software process involves the loss of a substantial investment.

We now describe the Capability Maturity Model on an introductory level, to identify the efforts that are required to realize process maturity improvement in an ICT-environment, and to define the notion of "maturity" for this thesis. As we do so, we will also make the connection to educational and multimedia-specific aspects that are relevant to this thesis. The Capability Maturity Model as a whole has a complex structure to account for the many existing types of organizations and it is beyond the scope of this thesis to discuss this structure thoroughly; for details, see [HUM1995; HUM1997]. Perhaps the most familiar component and the one that is most relevant here is the five-layer structure of maturity levels. In Table 3 we summarize this structure.

Table 3. *Maturity levels in the Capability Maturity Model.*

Level	Name	Description
5	*Optimizing*	Continuous improvement in control over project variables.
4	*Managed*	Wide control over project variables; extensive measurements.
3	*Defined*	Project process fully defined and some metrics in place.
2	*Repeatable*	A first success can probably be repeated because of basic management.
1	*Ad hoc*	No maturity. No tracking of basic project variables.

The lowest level in Table 3 is called the *Ad hoc* level because there really is no maturity, at least not from an organizational point of view. There are no formal procedures set up for tracking time, cost, and effort. There are no responsibilities made explicit and authority for decision-making is not defined. More than once these projects go from crisis to crisis; if the project turns out to be a success after all, this is probably due to the heroic efforts of enthusiastic individuals who spend

many hours working overtime to get things right. There is no way of knowing whether successive projects will be a success as well. No software development methodology is used, no configuration management is in place, and internal communication is mostly informal. This seemingly hobbyist approach does not, however, exclude any successes. Especially small companies producing small products may work most efficiently this way. However, for large projects (with budgets over $200,000) for large industry-sector customers, this way of working would be unacceptable, and a more mature way of working is required.

The Capability Maturity Model holds that, in order to achieve higher levels of maturity, an organization must have – to extend above the bottom line – at least a repeatable process. This means that once the organization has done a successful project, there is a sufficiently large body of knowledge and experience available (and accessible) to repeat that success. Prerequisites for this are that there is basic tracking of cost, time, effort, size, and risk. Project resources are allocated and assigned in a formal, documented way. Responsibilities and decision-making authority is defined. There is also one standard way of working that has been chosen to use as guide for developing the application in question. Sometimes there are also measurements of group coordination and project actions. The product quality is still variable, however, which means that there is still no sufficient knowledge to guarantee a stable level of quality of the product that is produced. Still, the reasons for the success of a project are known on this level, and may be utilized to repeat that success in later projects. When an organization can prove that they have reached that level (through formal assessment), we say that that organization has achieved CMM level 2. For any particular organization, we can quantitatively and objectively assess whether or not this is the case, which is important also in the scope of this thesis.

To achieve a still higher level of maturity, called 3, the *defined* level, many more aspects of the project process must be defined and agreed upon by everyone involved. On this level, the entire project process is formally defined for both management and engineering activities. Moreover, it is thoroughly documented, and this documentation is tightly integrated with the development process. Projects on this level use a documented though tailored version of the organization's process to both develop and maintain software applications. Cost and effort spent on this level are reliable, though the product quality may still vary somewhat. Also, a basic measurement program must be in place to be able to make predictions on future success and improvement. This seemingly logical step appears to be a difficult barrier to many organizations, as the majority of organizations known to employ the Capability Maturity Model still fail to fully comply with level 3 prerequisites. There are still higher levels of maturity (Level 4 is called the *managed* level, and Level 5, the top level, is called the *Optimizing* level), but because of the apparent difficulties of reaching even level 3, these are beyond the scope of this thesis.

We summarize the typical characteristics of each CMM maturity level, and the actions required to reach the next higher level of maturity, in Table 4 below. We observe that both the characteristics and the required actions are of a general nature and may well be applicable to courseware development process improvement as well.

Table 4. *Summary of levels 1-3 of the Capability Maturity Model*

CMM level	Typical characteristics	Actions needed to reach higher level
3: Defined	Defined software process for both management and engineering activities is documented, standardized, and integrated into the development process. Projects use a documented and tailored version of the organization's process to develop/maintain software. Cost is reliable, though quality is still unpredictable.	Quantitative quality goals for software products through establishing and tracking process measurements and quantitative quality goals, plans, and cost/performance. Calculate cost of poor quality and compare to costs of achieving quality goals.
2: Repeatable	Basic project management processes are in place to track cost, schedule, and functionality. Necessary process discipline is in place to repeat earlier successes on projects with similar applications. Product quality is variable.	Org. standard process for developing and maintaining software, through documenting process standards and definitions, assigning process resources. Measurements of group coordination and project actions.
1: Ad hoc	Professionals driven from crisis to crisis by unplanned priorities and unmanaged change. Surprises cause unpredictable schedule, cost, and quality performance. Few processes are defined; success depends on individuals' heroic actions.	Process must become stable and repeatable through estimation, measurement, and planning (size, costs, risk, and schedule); requirements mgt; quality assurance; ability to manage sub-contracts

Much research is available on software process improvement in general. We mention [AFT1983; JON1997; LAI1994; OMA1990; WER1994; YEH1993]. The work on process improvement by Aft [AFT1983] is more generally targeted towards productivity improvement in a wide range of industrial sectors, but it is striking to see how the principles described in this research are equally applicable to the software development world. A major pitfall of productivity improvement is that it often comes down to increasing the workload of the people involved, thereby eventually achieving the opposite effect, namely decreased productivity. Aft lists the following basic tools for productivity improvement: Improved workflow and materials handling; supervisory training; work simplification; job enlargement and redesign; attitude surveys; incentive plans; suggestion programs, and cost reduction programs. Aft also argues that to improve productivity, you must investigate how the use of resources to perform different activities contributes to overall productivity. Although the step itself seems obvious, in practice these measurement programs are rarely set up in a proper way.

Software process improvement in this thesis

For this thesis, the term "software process improvement" is only partly applicable. In our case, it would be more appropriate to talk about courseware process improvement. We observe that courseware process improvement comprises more than software process improvement. The fundamental addition to software process improvement is the multi-disciplinarity of the process and the multiple media of the product, and the far-reaching consequences this has for communication, organization, technology, and customer relations (as discussed in section 2.3). It is the inclusion of disciplines such as interaction design, didactical design, and audio and video composition that makes this area of research different from regular software process improvement. The focus is no longer on management and engineering aspects alone. In other words, we should investigate to what degree the capability maturity model is applicable to courseware process improvement.

Overview of Multimedia development

Multimedia development is concerned with integrating text, graphics, animations, audio and video into a product to enhance the user experience. Concepts such as engagement and persuasion are important here. We observe that the word multimedia is often used on two different levels, which is a cause of confusion as to what we are talking about: (1) multimedia as a mere inclusion of several technical media such as text, graphics, audio, and video, and (2) multimedia as a collection of disciplines that co-operate to produce "persuasive" software [FOG1999].

Let us briefly examine what the experts in the field consider to be multimedia. Vaughan, author of "Multimedia: making it work" defines multimedia as "any combination of text, graphic art, sound, animation, and video delivered to you by computer or other electronic means. When one allows an end user to control what and when elements are delivered, it is called interactive multimedia. When you provide a structure of linked elements through which the user can navigate, it is called hypermedia" [VAU1994, P7]. Feldman defines multimedia as "the seamless integration of text, sound, images of all kinds and control software within a single digital information environment" [ENG1996, P1]. The European Software Publishers Association puts the word multimedia in a broader context by stating that "…the implementation of multimedia capabilities in computers is just the latest episode in a long series: cave painting, hand-crafted manuscripts, the printing press, radio and television… These advances reflect the innate desire of man to create outlets for creative expression, to use technology and imagination to gain empowerment and freedom for ideas" [VAU1994]. The difference seen here between the perspectives of multimedia and traditional software engineering is striking. It appears that multimedia is much more concerned with the user experience and engagement, while software engineering is much more concerned with analytical functionality. Some authors have explicitly proposed looking at completely different areas to increase the quality of multimedia, such as Laurel, who draws the analogy with the making of theatre [LAU1993], and Winograd, who compares software development to the architecture world [WIN1996].

Multimedia as we now know it has descended from (*a*) courseware development when multiple media elements were being introduced for instructional and engagement purposes (see chapter 1), and (*b*) from the gaming industry, as a result from many efforts to make games as realistic as possible (outside the scope of this thesis). In principle, any computer application using more than one medium may be called multimedia, but we usually only speak of real "multimedia" when at least the graphics play a major role, plus (preferably) an important role for video and audio, depending on the size and scope of the application to be developed. Only when computers became powerful enough to allow relatively easy production of persuasive digital multimedia (in the nineties), did the field of multimedia boom. These days, multimedia is mostly used in the gaming industry; computer based training (increasingly web-based); and marketing and sales CD-ROMs. In section 2.2, we have discussed the needs of educational system development teams regarding multimedia, so we will not repeat this here. Readers that want to go more in-depth into the use of multimedia for (educational) software development are referred to [AND1990; BOY1997; CLA1994; ENG1996; JAC1997; VAU1994].

Overview of Human Computer Interaction

Human-Computer Interaction is the study of the way in which people and computer technology influence each other. It is the subject of computer science, psychology and cognitive science [PRE1994]. Cognitive science in relationship with human-computer interaction investigates issues such as human input-output channels, the workings of human memory, how humans cope with

reasoning and problem-solving, and individual differences between people. Computer science in relationship with human-computer interaction is concerned with input and output devices, positioning and pointing devices, screen layout, and use of graphics, and the inclusion of multimedia elements. Psychology in relationship with human-computer interaction looks at models of interaction, learning curves, ergonomics (use of color etc), and the effectiveness of various usage scenarios. Summarizing, human-computer interaction is about improving the usability of software and addressing human factors in software development.

Of all computer science niches, human-computer interaction has probably contributed most to involving other disciplines in software engineering. Other fields such as movie production have learned much quicker how to incorporate multiple disciplines, and modern buildings could never exist without the input of several disciplines. Until the nineteen seventies, human-computer interaction was mostly limited to monochrome, character-based screens with little graphics, less audio and no video. Since the end of the nineteen seventies, more and more researchers and practitioners aired the need for a stronger focus on the human factors in software development. In 1983, Ben Shneiderman coined the term "Direct manipulation" [SHN1992] as a metaphor for the Graphical User Interface, commercially introduced by Apple in 1984 with the MacIntosh computer. This has greatly contributed to the awareness of the importance of the interaction between people and computers: windows-based direct manipulation user interfaces are now commonplace for an increasing audience. Nowadays the field of human-computer interaction spans a wide range of topics, including task analysis, interaction design, layout design, ergonomics, and usability testing [DIX1993].

The properties of a direct manipulation user interface are [SHN1992] (1) visibility of the objects of interest; (2) incremental action at the interface, with rapid feedback on all actions; (3) reversibility of actions, so that users are encouraged to explore without penalties; (4) syntactic correctness of actions, so that every user action is a legal operation, and (5) replacement of complex command languages with actions to manipulate directly the visible objects. This has resulted in the so-called WYSIWYG screen interface ("what you see is what you get"). Strictly speaking, though, most software only achieves an approximation of this. However, despite of problems in actually realizing WYSIWYG, usability of software applications has greatly improved. Important principles of usability are learnability (allow users to quickly understand how to use the software and then attain a maximum level of performance), predictability, familiarity, consistency, customizability, robustness, and task conformance.

Another milestone in human-computer interaction is the now widespread use of hypertext. A textbook has a linear information structure, but it is full of ideas and footnotes that persuade the reader to digress into a related topic. With online media, clicking on the relevant word takes the user to that topic. We call this nonlinear structuring of information. With the advent of the World Wide Web, nonlinear information linking has become commonplace and has greatly expanded the possibilities for users to investigate a certain topic. It has also, by the way, greatly contributed to loss of information context and navigational confusion.

Currently, an ever-growing worldwide community of human-computer researchers debate on issues on how to improve human factors in computing systems. These people unite each year in several conferences, most notably the ACM CHI (Computer-Human Interaction conference, which draws several thousand people each year (see www.acm.org/sigchi). The importance of the quality of the

user interface is now commonly accepted, although not always practiced on a mature level. For authoritative textbooks on human-computer interaction, we refer the reader to [DIX1993; HEN1991; MYE1994; NOR1988; PRE1994; SHN1992]

Human-computer interaction in this thesis
The field of human-computer interaction is important to this thesis because educational systems are among the software products with the highest levels of interaction (see chapter 1, and [VDM1995A]). Human-computer interaction related areas such as task analysis, interaction design, layout design and usability testing are all very important to the development of courseware. This is why courseware development projects often include disciplines in this area, notably interaction designers and usability testers.

Overview of Knowledge management
Knowledge management is relevant to this thesis because we have observed in chapters 1 and 2 that the courseware process improvement that we investigate may partly be realized by facilitating courseware development teams in their needs for quickly accessible knowledge. At this point, however, we must realize that we must first define the terms "knowledge" and "knowledge management" if we are to use them in our research. This is a difficult point because there are no generally accepted definitions for knowledge and knowledge management. Especially the word "knowledge management" seems to be applicable to a large variety of activities, ranging from collaborative work (non-technical) to expert systems (technical), seemingly making the term effectively empty of meaning. Since there are no accepted definitions of knowledge and knowledge management, we are faced with the necessity to carefully define just what we mean when we use these terms, and we shall do so here. Throughout the thesis, wherever we talk about "knowledge" and "knowledge management", it will be in the sense as defined in this section. First, however, we need to summarize existing notions of the words "knowledge" and "knowledge management".

Knowledge
Definitions of the word "knowledge" include (extracted from work by [WEG1997; WEG2000], and by definition incomplete):

- The ability to transform information into quality decisions.
- The whole of meanings, terms, skills, ways of working that we consider to be true and that guides us in our decisions.
- The ability to predict the behavior of entities and people.
- Knowledge is the ability that an individual has to carry out a task by selecting, interpreting and assessing of available data, thereby creating (new) task-relevant information.

We observe that these definitions all implicitly assume a certain purpose for which the knowledge is used or required, for example building experience, building expertise, improving reasoning, and identifying facts. Furthermore, when considering these definitions, there appears to be a connection between the terms "data", "information", and "knowledge". Data becomes information when we add context to it and when we know what we can do with the data. Information becomes knowledge once we start reasoning with the information in a way that allows us to achieve a certain goal. For the scope of this thesis, we will employ the following working definition of knowledge:

Knowledge is created through reasoning with information that people have available to help them make carefully balanced decisions in achieving a certain goal.

In other words, we explicitly link knowledge to the information needs of people; in this thesis, the people are experts who are involved in producing courseware. Furthermore, when we talk about *reasoning*, we refer to the reasoning that is done by human brains, not artificial intelligence systems.

Knowledge management
Definitions of knowledge management include (extracted from work by [WEG1997; WEG2000]):

- The effort that is required to manage the production factor called "knowledge". This includes the identification, gathering, categorization, dissemination, archiving and deleting of knowledge.
- Managing this "life-cycle" of knowledge in such a way that valuable knowledge is kept available and irrelevant knowledge is discarded.
- Knowledge management involves increasing the return-on-knowledge and general benefits of the production factor knowledge.
- Knowledge management is coaching and managing knowledge workers.

Although some researchers hold that we, as human beings, have been practicing knowledge management for many centuries, the modern notion of knowledge management was not in existence until about 1995. A relatively new area of research is often surrounded by a publication hype, and knowledge management is no exception. Though much has been written about it in the nineties, mostly in an organizational context [HSI1993; DAV1998; JAC1996; PRU1994; RUG1997; WEG1997], there are still relatively few organizations that have a true success story to tell. For the scope of this thesis, we will employ the following working definition of knowledge management:

Knowledge management consists of all activities and procedures required to supply people with all the information (!) that they need to create knowledge to achieve a certain goal. It includes the identification, creation, categorization, dissemination, and archiving of all knowledge relevant to achieving that goal.

When applied to courseware development, we see that knowledge management becomes the whole of activities and procedures that are needed to make sure that courseware development experts have available all the information to create knowledge that they need to produce the courseware product. Through reasoning with information, experts create knowledge that is relevant to themselves and to the entire courseware project. Our definition of knowledge management requires defined ways to identify, create, categorize, disseminate, and archive all relevant knowledge. This is addressed in chapter 5.

The main reason for the recent interest in knowledge management (in other words, why put effort in knowledge management at all) is mostly driven by commercial reasons as follows (from [BOO1997]). If knowledge within an organization is properly managed, that organization can save on knowledge procurement expenses, make better use of a collection of "best practices" wherever needed and required, and therefore sharpen the "competitive edge" of that organization. Competitive advantage will, theoretically, result in improved average service levels of that

organization. This in turn directly results in an increased shareholder value, a very important aspect to commercial companies. In this thesis we focus on supplying courseware development teams with the information that they need in courseware projects; we refer the reader to other publications for the link to competitive advantage [ALL1997; BOO1997; DAV1998; JAC1996; LEO1995; NON1995; PRU1994; WEG2000].

With the absence of generally accepted definitions of "knowledge" and "knowledge management", it is no wonder that most attempts at "implementing" knowledge management in an organization have up to now not resulted into many success stories. Goals such as "leveraging the intellectual capital of the organization" remain too fuzzy to result in success as long as this is the case. One of the most widely accepted insights regarding knowledge management is the notion that it is more than the acquisition of a robust technical solution; non-technical aspects are at least just as important. In the nineties, many knowledge management implementations that focused mostly on technical solutions failed because insufficient attention was paid to non-technical aspects such as:

1. **Organization**. This includes procedures for starting up knowledge management, keeping it alive and under control. Here, roles tasks, and responsibilities are defined.
2. **Culture**. This includes motivational reasons for people to co-operate in sharing their own knowledge and using knowledge of others. This aspect is about attitude, incentives, and awareness of group performance improvements. A quality communication plan should be set up to persuade people to share their own knowledge.
3. **Content**. An initial critical mass of knowledge is crucial for any knowledge management solution to succeed. People may be most important when managing knowledge, but it is actually the content (knowledge) that is in people's heads and in people's desks that will be organized. Therefore, this knowledge must be made explicit and visible. It must also be identified and categorized.

Summarizing, a knowledge management implementation also addresses motivational aspects, deployment, and quality of information. Davenport *et al.*[DAV1998] propose the following phases for a professional knowledge management implementation:

1. The **setup** phase: Set up the project: write an assessment plan, and generate a project handbook.
2. The **assessment** phase: write a modeling plan, organize an assessment workshop, organize an awareness workshop, typify the current "company culture", carry out a culture assessment test, determine the feasible scope of the project, carry out intake interviews, and present the usefulness of knowledge management to all parties involved.
3. The **modeling** phase: design the knowledge infrastructure ("hard" and "soft" aspects), determine the targeted culture, and determine the way of implementing knowledge management
4. The **pilot** phase: implement knowledge management in a small, preferably controllable environment;
5. The **realization** phase: implement knowledge management at the target organization, building on the results of the previous phases;
6. The **roll-out and maintain** phase: keep employees motivated, in other words, keep knowledge management alive; coach the knowledge "champions", and train the knowledge motivators.
7. The **evaluation and archive** phase: record lessons learned.

We observe that Davenport assigns only a modest role to technical solutions. On research in the same area, Boone [BOO1997] provides an alternative view on successful implementation of knowledge management. He views knowledge management as consisting of obstacles and spurs. The obstacles are (*a*) awareness barriers and (*b*) interest barriers. The spurs consist of (*a*) stimulators to initiation, and (*b*) facilitators to effectuation. Before any implementation of knowledge management in an organization can take place, there should be a stage where all parties become motivated to co-operate (the *interest* stage), which is in turn preceded by a stage in which all parties involved become aware of the importance of knowledge management (the *awareness* stage). Boone identifies the following awareness barriers:

- Tacitness of knowledge. Formal and systematically categorizable knowledge constitutes only a small part of the knowledge of people; this is because a large part of our knowledge is tacit (meaning that "people know more than they can tell"). This makes it more difficult to effectively employ that knowledge.
- Lack of "who-knows-what" facility. Since most of us do not know what our colleagues know, it is difficult to identify, gather, store, and disseminate important knowledge. If you do not know whom to contact to acquire a certain piece of information, both knowledge users and knowledge creators (most of us are both) experience difficulties managing knowledge in projects and in the entire organization.
- Bootlegging. A term coined by Bright in 1967, this has to do with doubtfulness of the quality of a piece of information. If a knowledge item is surrounded by uncertainty, people can lack willingness to diffuse the knowledge because of the risk of public scorn or even punishment.
- Cognitive limits to discern knowledge. People are by definition limited in their capabilities to focus on new information inconsistent with their current reservoir of knowledge. This has to do with habits and perception, but it is also something that makes us human and therefore cannot be entirely avoided.

Interest barriers include:

- Efficiency rationales. Setting up knowledge management is still seen by many mangers as something that is too fuzzy to result in an improvement in efficiency in the organization.
- Opposition by the knowledge donor. This is a person-related barrier, and includes (a) an uncertain return on the donor's investment in sharing knowledge; (b) the advancement of internal competitors, resulting in (c) a loss of the donor's power in the organization.
- Opposition by the knowledge recipient. This is also person-related, and includes (a) the "not invented here" syndrome (what I/we make is probably better); (b) resistance to change someone's own personality; and (c) resistance to change in social structure.

Breaking these barriers requires a substantial investment in organizational and cultural aspects. Boone proposes to employ a "Knowledge Sharing Persuasion system" that is specifically targeted towards minimizing the barriers mentioned above. Such a persuasion system is based upon motivation (achievable by for example expected rewards, and group pressure), which is in its turn dependent on trust (shared norms and values, and past experience). For later phases of implementation of knowledge management, a *knowledge sharing complexity reduction system* is proposed, and for the continuous use of knowledge management a *knowledge sharing media system*.

In a relatively short time, the available amount of resources on knowledge management has grown significantly. There are web sites with literally hundreds of links that all take the browser to research on knowledge management, for example http://cis.kaist.ac.kr/research/kmsite.htm. Regarding written literature, valuable references are [ALB1997; ALL1997; DAV1998; JAC1996; LEO1995; NON1995; WEG1997, WII1995]. But the reader must be aware of the fuzziness factor of this research field: it may well be that by the time you read this thesis, knowledge management as a research field has shifted to other focuses.

Hatchuel and Weil [HAT1995] look at knowledge management from an "expertise" point of view, in that it is ultimately the expertise of the experts that creates added value for a company. They describe expertise as "comprising a set of theories and questions on which an activity can be based, or from which data can acquire meaning by generating new theories or questions". Analogously, they discuss the difference between knowledge and reasoning. "Raw" knowledge is meaningless without some form of implicit rules to create statements or decisions, which is where the notion of "reasoning" comes in. Expertise, then, is built up of a combination of knowledge and reasoning with information. They stress that expertise is not by nature a discipline nor a science, and the capturing of expertise encompasses much more than just a technological approach. Even though, they claim that the careful employment of expert systems (as in artificial intelligence expert systems) can be used to improve control and diffusion of knowledge. Eventually, using expert systems for knowledge management, it could imply "the institution of new relations between the experts themselves and the system in which their expertise was used" [HAT1995; STE1986]. The most difficult issue for investigating this is the premise that a person's expertise is not something that is easily accessible (it is *tacit*). Even though, theoretically, expert systems are potentially very powerful instruments in capturing and controlling the diffusion of knowledge [RUG1997], many companies are not heavily investing in them and many expert systems projects in commercial organizations have not advanced beyond planning stages [HAT1995]. Since our research is of an empirical character, we have chosen to exclude the expert systems point of view on knowledge.

Knowledge management in this thesis

When combining the notions of software process improvement and knowledge management, we observe that there appears to be a relationship between the two: to reach a higher level of maturity, the way of working must be unambiguously defined, and earlier successes must be repeatable, which requires careful management of experiences and expertise. Knowledge management, at least in the notion as we have defined in this section, seems suitable to assist in achieving this. We must investigate if this holds true for courseware development as well.

Conclusion

In this chapter we have investigated the courseware development process. We have observed that existing strategies for courseware development mostly focus on product quality and less on the quality of the development process. We have identified the various disciplines that are required in courseware development projects, and we have investigated issues that may arise in the project process; these include the working relationship with the customer, the working relationship between the various available disciplines; the quality of individual estimates and consensus estimates, and the need for quickly accessible information to carry out tasks and activities more efficiently and effectively.

We have observed that courseware development inherits many principles from information systems development, but that especially from a software engineering point of view courseware development has special characteristics (for example, no unit for product size) which have up to now not been thoroughly addressed. We have also observed that accepted methods for software process improvement, particularly the well-known Capability Maturity Model, offers a quantitative way of defining the word "maturity", which we may employ in our investigation in courseware process improvement. Finally, we have defined our notions of the fuzzy terms knowledge and knowledge management. We will employ these working definitions to give shape to our approach to courseware process improvement in the next chapters.

3. Research approach and hypothesis

In chapters 1 and 2 we have described the topic of our research. In this chapter, we present the research approach, which consists of a research strategy that employs certain research instruments. In order to ensure a manageable research project, we define the research boundaries and the context in which our research will take place. The research is aimed at testing the research hypothesis, which we present in section 3.3.

3.1. Research strategy

In this thesis we investigate ways of realizing improvements in the maturity of the courseware development process. To this end, we have surveyed, in chapters 1 and 2, the field of courseware development, and we have identified a quantitative notion of the word "maturity", which we may use to measure the effect of the solution that our research produces. We have observed a number of issues that may arise while developing courseware. Since these issues take place in the real world with its numerous unpredictable environmental factors, we feel that our research cannot take place in a laboratory setting – we would risk producing a (valid) result that might hardly be usable in an empirical situation. Instead, we aim to consider a real-life environment where courseware development takes place in the sense as described in chapter 2. If we can then verify that this environment satisfies certain required research boundaries (i.e., the environment allows us to conduct research in a proper way), we may intervene in this environment by carefully offering a solution to the environment that is geared towards the goal of this research, in other words, to test our research hypothesis. This is sometimes called the *interpretive* view on research, which holds that research should be conducted by studying certain phenomena in their natural context, and that the researcher can intervene, and thus influence the environment, where desired or necessary.

The main goal of our research is to measure the effectiveness of our solution for increasing the maturity of the courseware development process in the sense of the CMM. This is unique because very little research exists (in 2001) on courseware development process improvement using a maturity framework such as the Capability Maturity Model. This means that we cannot test the applicability or validity of some existing theory; rather, we shall have to proceed in an inductive way, observing our environment and the effect of the influence that we eventually exert on it. We conduct our research at a commercial ICT services company (referred to as the *company of study*) that is committed to improve its courseware development process.

Building on this approach and taking into account our observations and research questions of the previous chapters we can now define a set of research steps, which collectively form our research strategy. The research steps are:

1. Identify the most urgent needs for improvement of the courseware development process, and investigate the problems that cause these needs (*problem identification*, we address this in chapter 4);
2. Investigate what causes these problems, and identify directions for addressing these causes; compare this to existing literature (*problem analysis*, chapter 4);
3. Conceive solutions for improving the courseware development process using the framework of the Capability Maturity Model and verify that the solutions indeed address the

needs for improvement reported in research strategy step 1. Then, implement the solutions into the project process (*design* and *realization* phases, chapter 5);
4. Objectively measure the amount of improvement into this project process through assessment of the process maturity in the sense of the CMM through statistical analysis of quantifiable research results, and further discuss the outcome of the experiment (*experiment, synthesis*, and *evaluation* phases, chapters 6 and 7).

When considering the research steps, we see that we inductively follow the interpretive research approach: we observe the research environment in an empirical situation, we then intervene by offering a solution to address the problems, and we observe (measure) the effect of our solution.

We employ research instruments to carry out the research steps. Since we do not have a laboratory environment available to carefully control all variables relevant to the company of study, our research instruments are geared towards an empirical approach to the research project. We do an empirical pilot study, including structured face-to-face interviews, for research steps 1 and 2 above, and we conduct a real-life experiment for research steps 3 and 4, in which we validate the effects of our research in a statistically valid way. A team of independent researchers will do this statistical validation, to ensure undependability of the research.

3.2. Research boundaries

In the ideal situation, a researcher has good control over the research context, with few and stable research assumptions and fully dedicated subjects of study. In real life, however, a thorough research is a struggle to cope with political issues, tight budgets, time shortage, and conditions of study that invoke certain amounts of noise to the analyses and experiment. Many surrounding aspects can influence our observation and measurement process. In order to minimize the effect of these unknown influences, we should define strict requirements for the research project boundaries and constraints. There are a number of research boundaries to identify if we are to draw valid conclusions at the end. In this section we define the types of courseware development organizations that are taken into account, the client organizations that are relevant, and additional boundaries.

Types of educational development organizations
A large number of organizations worldwide are actively involved in courseware development. These organizations have many faces, and we must necessarily restrict our focus to a small subset of them. A key distinction that is first made is that between non-profit government organizations, and commercial (as-much-profit-as-possible) companies. In commercial companies, projects are generally more hectic, deadlines are tighter; stress is generally higher, and the focus is more on Income From Operations (meaning money). Customers are also usually different (we will address this later). This is the type of organization that is relevant to this thesis. In non-profit organizations, these kinds of projects are less budget-driven, and stress is generally lower. The level of formal procedures does not really need to be different in any of these two types of organizations, though commercial organizations tend to be somewhat more flexible regarding this issue, unless the commercial organization becomes really large.

A second important distinction is that between media-oriented companies and ICT-oriented companies. In media-oriented companies, the focus is more on creativeness and the artistic process. These companies generally work in small project teams with few formal procedures. These types of

organizations are usually smaller as well. In ICT-oriented companies, projects are generally more formal and are seen more from a functionality point of view, as in "what should it be able to do", as opposed to the "what should the user experience" focus found at many media-oriented companies.

Since it seems reasonable to assume that research into the ICT-oriented type of (project) organization is better manageable and measurable (because of its more functional focus), we choose to restrict our research to that type of organization. This perhaps implies that the results of this research are applicable to that type of organization only.

Finally, we mention the cultural difference from a geographical point of view. The fact that the research project was carried out in a northwestern European culture does not automatically imply that the research results are fully applicable to similar types of organizations in other cultures, for example, Asian or even American. This is due to differences in culture (the way of thinking and the way of working). General conclusions of this thesis on the use of knowledge management to achieve process maturity may still hold; however; the author recommends conducting additional research with other cultural environments to determine if this is true.

Types of customers

An important consideration for our research boundaries is the type of customers that we consider. Sometimes, there is not even a specific named customer; in such a case, the application is made for a specific target audience (for example, children in the range of 10 to 16 years old). For this thesis, though, we assume that there is always a named customer. However, merely stating that there is a named customer is insufficient. Customers can range from a small, one-person organization, to international conglomerates consisting of hundreds of departments worldwide. For all these types of customers, especially the size of these projects is an interesting factor, when speaking of project size in terms of money and throughput time. Courseware project budgets can easily exceed US$200,000 [FAL1995]. This is usually the case if a product is made for a very large, international customer, or if the product is to be marketed for a large target audience. As project budgets increase, the value of the contract increases as well, generally resulting in more formal procedures for the way of working.

Another important distinction in types of customers is the level of familiarity the customer has with being involved in educational multimedia projects. If the customer already has some experience with these kinds of projects, its role will more likely be that of a "partner". If, on the other hand, a courseware project is the first time for a customer, managing that customer's expectations becomes extremely important [VAA1997]. This is because the customer in this case has no way of estimating the required cost for particular features. Any given functionality that seems easy to realize logically speaking may be very hard to technically implement, and vice versa. Misunderstandings can be avoided if the information analysis and functional design is produced by all disciplines involved, including the customer. For this research project, customers are usually large commercial organizations in industry, financial services, insurance, heavy industry, telecom, and aviation.

Project size boundaries

We have earlier observed the spectrum of courseware development project sizes: the individual approach, the team approach, and the organizational approach. It seems too ambitious to pretend to be able to offer a solution for the entire size spectrum of courseware development projects, and therefore we will have to choose. Since the individual approach is generally the least professional

approach, and also since we are considering the aspect of multidisciplinarity, we will exclude projects that employ the individual approach. Since the organizational approach may easily become too complex for the goal of our research, we will focus on projects that follow the team approach. We therefore choose to focus our research on project teams that are staffed with four to seven experts, in a throughput time of a few months to a half-year. This sets the money involved in these projects roughly between US$25,000 and US$250,000 (in the 1990s). This a different order from projects that involve millions of dollars. Our research results are by no means directly applicable to these larger projects.

We now summarize the research boundaries on courseware projects that we have chosen in order to make sure that our research is still manageable:

- The courseware project is defined to deliver a software application that includes multimedia elements, i.e., more than just text and still graphics.
- The courseware project duration (throughput time, the moment from which the project bid is accepted, until the day that the product is archived) must be more than six calendar weeks. Smaller projects will not be taken into account.
- Project staffing should be such that on average, more than three people with different expertise work on the project fulltime;
- The content for the final product should be stable, or it is reasonable to assume that the content will soon become and remain stable.
- The platform on which such an educational multimedia application runs is of minor importance, provided that the choice of platform does not require significant changes in the way of working of the project team.
- The operational level (administration, design, building, testing, evaluating) and tactical level (customer contacts, management, planning, estimates) of courseware projects will be taken into account. However, strategic issues ("do we want to do this project?", relationship networking games, politics, hidden agendas etc.) are harder to influence and will not be taken into account.

Requirements

In order to guarantee the usefulness of the recommendations and conclusions of our research, we have defined a set of research requirements, which can also be seen as research boundaries. We now discuss each of these.

Requirement 1: Content delivery

We analyze courseware projects that have a reasonable chance of their content being delivered on time or reasonably on time. If the customer fails to do so, the project team will most likely not have a successful project. One may say that to include this constraint might imply skipping the majority of all potential projects. However, our research will show that although this is an ever-present problem in many projects, the majority of these can be completed with reasonable success after all. This is because in many projects this problem is only marginally present.

Requirement 2: Agreement on the definition of courseware

Asking ten people to define what a courseware project is, is likely to result in ten different answers. Researchers who consult this thesis for advice or recommendations should know what to expect

when we talk about courseware. Therefore we will assume a common definition of courseware, which we have presented in chapter 1. This definition is applicable to all courseware projects that we analyze.

Requirement 3: Qualification of disciplines
For our research project we require that the availability of skills and qualification of disciplines is acceptable. This means that our company of study should be able to carry out a courseware project of the size that we have chosen with all required disciplines in place, as surveyed in chapter 2.

Requirement 4: Definition of success
A measure of just what "success" is for a courseware project should be identifiable. Since this research is about increasing the number of "successful" projects, we need to have an objectively usable (measurable, quantitative) definition of just what success is in the scope of our research. Within the scope of this thesis, we will express the success of a courseware project in (*a*) the quality of the estimates of the project variables (we will define this quality later in measurable terms), and (*b*) the amount of positive versus negative process experiences reported by the project team. In other words, we choose to define the success of a courseware project from the project team point of view; this definition excludes the notion of "success" as experienced by a customer. In the hypothesis that follows, we will further explain our notion of success by expanding on points (*a*) and (*b*).

Requirement 5: Definition of size
A very awkward problem to address in educational multimedia is to find a non-subjective, unambiguous way to determine (calculate) the size of a courseware product. We need the product size variable (among others) to be able to quantitatively measure improvements in the project process, as we will see in the experiment in chapter 6. In the scope of our research such a definition of size has been designed and published [VAA2000A]. You can read more about this in appendix E.

3.3. Research hypothesis

Building on our observations in the previous chapters and the chosen research approach and research boundaries, we now define our hypothesis as follows:

> The availability of an instrument that facilitates courseware development teams, as defined in chapter 2, in managing their knowledge, in the sense as defined in section 2.4,
> (1) in projects that satisfy the research boundary conditions as defined in this chapter,
> (2) in an empirical environment as described in this chapter,
> results in:
> a. Higher quality estimates of quantitative project variables of courseware projects, meaning that the estimates of these project variables approach their actual values more closely;
> b. Statistically significantly fewer occurrences of negative process experiences, and significantly more occurrences of positive process experiences.
>
> The aforementioned instrument and the effects of its usage will result in a more mature courseware development process, where maturity is measured according to the definition of the Capability Maturity Model.

Please note that the hypothesis does not say anything about more successful courseware products as experienced by the customer or as observed in real-life usage. We even cannot exclude the possibility that products are not experienced as being more successful even if we observe that we cannot falsify the hypothesis. However, we are convinced that even if only focusing on improving the development process of the courseware project team, we may contribute to the improvement of the quality of courseware.

Assumptions connected to the hypothesis

Since the world in which our research takes place is not fully controllable, we must make explicit some assumptions on factors that may influence the research project. Assumptions are statements that we believe to be true in the context of our research. We can hold these assumptions true if we can make them plausible (since an assumption can be difficult to prove or falsify). The major assumptions for our research are:

Assumption 1: Interapplicability

Our research results hold for all ICT-oriented project organizations as defined in this chapter. This seems reasonable to assume if the following conditions are met to a certain degree:

1. The organizational structure of the courseware department is sufficiently similar;
2. The company culture of that department is sufficiently similar;
3. The distribution of experience and expertise among employees is sufficiently similar;
4. The market in which this department operates (educational products for large customers) is similar.

Assumption 2: No "political" influences

The assumption here is that projects have a reasonable chance of not being bothered too much by organizational power conflicts. An important failure factor of projects has been shown to be of a negative political nature [DEM1987; BLO1983]. If issues such as hidden agendas and power "games" get in the way too much, the cause of the failure is largely unavoidable for the project team, including the project manager. Our research may make a range of recommendations for courseware development, but that does not usually help to solve the root of political problems. Therefore, we assume that political influences do not too heavily burden the educational multimedia projects that are analyzed in the scope of this thesis. Having said this, we must also note that the word "politics" in itself is not negative by definition. Politics will always be an inevitable element of human existence, and can also have a positive influence on projects. In this thesis, when we talk about "politics" we always mean the negative side of politics (as in organizational power conflicts), unless explicitly noted otherwise.

Assumption 3: Time independency

Our assumption is that our research results are still applicable five years from start of the research. This research project started in 1996. At that time, one of the most important hazards that were identified was the applicability of its results at its final stage: recommendations about tools, for example, would almost certainly be outdated by the time this thesis would be published. We attempt therefore to let the measurement experiment be as independent as possible on technological issues. Therefore, the constraint that our research only recommends on reasonably time-independent factors makes plausible the assumption that the results and conclusions of this thesis are still applicable at the date of publication of this thesis (2001).

4. Problem analysis

In this chapter we identify, clarify and analyze the needs for courseware process improvement in an empirical environment; we conduct a pilot study to do this. We then identify and analyze the underlying problems that cause these needs. The results of the problem analysis are leading in our construction of an instrument that is geared towards facilitating the knowledge needs of courseware project teams, which we describe in chapter 5.

4.1. The need for improvement of the project process

Consider the following situation at an information and communication technology (ICT) services company called Atos Origin (our *company of study*). The services offered by this company cover a wide range of ICT topics that include Professional Services (of which tailored software development is a large part), Managed Services (maintenance and deployment), and Enterprise Solutions (such as SAP and BAAN). Within this organization, in The Netherlands a special group of about forty educational multimedia employees is organized in a so-called local unit, involved in courseware development. There are various disciplines available within this local unit. Each expert brings in his or her own expertise on one or more of the disciplines required for professional courseware development according to the team approach (described in chapter 2). There is a managing director, three selling consultants, a human resource manager, several technical experts on software development, but also on the areas of graphics, video and audio; there are didactical experts, interaction designers, content designers, communication experts, graphical artists, composers, multimedia consultants, and, last but not least, multidisciplinary project managers. The latter are responsible for managing a courseware project team of commonly five to eight people, including him- or herself.

We started our research by observing the courseware development process at the relevant local unit of our company of study. It was generally felt that the courseware development process was not of a satisfactory quality level. We employed the instrument of structured face-to-face interviewing to identify needs for improvement in the way of working in courseware development at our company of study. To this end, an initial questionnaire was constructed. This questionnaire was of a general nature; it invited courseware experts to freely report on frequently occurring process problems. It was called the *zero-questionnaire*, because it laid the basis for further research. The experts that were interviewed were distributed across the disciplines according to the ratio of presence, meaning that several content designers were interviewed (because content design is present in almost all educational projects), but the music composer was not (because original music is not often used in courseware projects).

We devised the zero-questionnaire with the help of work by Oppenheim [OPP1992], called *"Questionnaire design, interviewing and attitude measurement"*. Oppenheim states that questionnaires must be "purely objective, devoid of any (hidden) assumptions, omit even the slightest chance of ambiguity, be structured according to a logical model that is devised from the reason to do the questionnaire, and be fit for statistical analyses and evaluation". Oppenheim repeatedly stresses especially the last point mentioned, because if statistical analysis cannot be properly fit to the questionnaire results, it becomes hard to conclude anything at all with certainty (or plausibility). Therefore, the zero-questionnaire had to be designed without any previous assumptions on the

kinds of problems that would emerge. This was not easy because we wanted to capture important process problems about all possible aspects of the work, without directing interview subjects into any direction on purpose. With this, we had to keep in mind that these aspects of work did not necessarily have to be the same as problem categories. On the other hand, at this stage we did not need to perform any complex statistical analysis on the questionnaire results: we merely needed an indication for the direction of future research. To achieve this, we count the number of problems and roughly weigh each of them; from this set, we then generate a first general categorization of project process problems.

The questions of the zero-questionnaire are formulated in a non-assumptive way, such as:

> *What are – according to team members – the most important causes of the failure to complete the multimedia product on time and within budget?*

We now give samples of typical answers that the courseware experts gave on questions posed in the twenty-two interview sessions:

"…A lot of pressure because many activities had to be done in parallel and things were happening simultaneously without one knowing about the other."
"… The specification team and the programming team were harassing each other too much…"
"… The project planning was adjusted twice in the midst of the project. That's a real de-motivator…"
"… If a project isn't fixed-price, estimates are of crucial importance… bid must be based on hard arguments then…"
"… If communication is disrupted during project process, errors will slip in which produces overhead."
"… A serious bottleneck is at one important person of the customer, who made us spend a lot of money on project bids but was afraid to make important decisions."
"… As to how everything should be realized… it's a matter of previous experience, there are no pre-processed handbooks for that."
"… Having somebody in the project team with exceptional expertise of one tool is most definitely a substantial push in the back."
"… Communication between [A] and [B] was difficult. They wanted some rather impossible things. It was hard to convince them. To me, that's the most complex part of the project."
"… A real problem was when we lost someone with multiple disciplines. We were stuck with knowledge of design and with knowledge of the programming tool but the link between them was missing all of a sudden."
"… Sometimes we have some small-seeming problem that is technically very difficult, and it's hard to convince the customer of that."
"… The project coordinator didn't really notice the uncertainties that were raised by the customer and that we had to solve…"
"… When planning a project, it's hard to fit everything within the boundary conditions… you'd need to organize a meeting with all project members but that takes too much time."

We organized the initial interviews in individual sessions, each lasting about half an hour. The total set of employees that was interviewed was a representative sample of the entire local unit because all the available disciplines were interviewed according to their ratio of presence. About forty percent

of the experts are female. The average age of the experts is about 32. The level of education of the employees is HBO or university. We employed the following procedure for each interview:

1. Interviewer poses question from the zero-questionnaire to person being interviewed.
2. Answer from person being interviewed is written down literally.
3. The list with all interview questions plus answers is sent to the person that was interviewed.
4. Person that was interviewed checks this list for inconsistencies and ambiguities.
5. The (corrected) list is sent back to interviewer who stores the list for later analysis.

The answers from the twenty-two interviews were analyzed and grouped according to seemingly similar issues. The following general list of issues was constructed from the set of answers that were collected in the first interview cycle.

- The quality of estimates for the project planning is very important for the rest of the project. However, making such a planning can be very difficult. The main source for planning is the planner's own experience with planning projects, and the expertise of other project team members that are involved.
- It is often difficult to convince the customer of the difficulty of technical issues. In addition, much time is often lost on ineffective discussions with the customer, which should have been dealt with effectively by the project-managing designer at an early stage.
- The project manager often has too many tasks to do in too short a time span; in other words, the workload is often too heavy.
- There are no formal standards employed for project planning, set-up, documentation, and meetings.
- When valuable lessons are learned concerning project control, they are almost never documented. If they are, it is done in such a way that they are not usable for later projects.
- The project manager often plans the project milestones too tight, with little room for any unforeseen consequences.
- Communication within the project team itself during the project is too often of a poor quality. Meetings are hurried, things are not accurately written down, people misunderstand each other, and interpret agreements in different ways.
- There is not enough expertise available with the innovative development tools. Much time is lost on mastering the abilities of the tools.
- High time pressure results in increased stress with the project members that can cause people to become ill, as well as decrease the quality of the end product.
- Project members do not often bring out problems when they identify them, which can result in a snowball effect of problems, or frustration during the rest of the project.

From this list, we now generate a view that summarizes the identified process issues in courseware development in relation to the project itself. We present this view in Figure 7. Since the target of this problem analysis is to identify further research steps, we especially look at the nature of the issues, for example if they are about technical issues or about communication, etcetera. The thick inner box represents the project process without regard to the flow of activities. Inputs to this process are documents such as conceptual design on the other hand, and tacit inputs such as

expertise of employees on the other hand. This together makes up the courseware project, denoted by the outermost box. The project ultimately results in the courseware product being delivered.

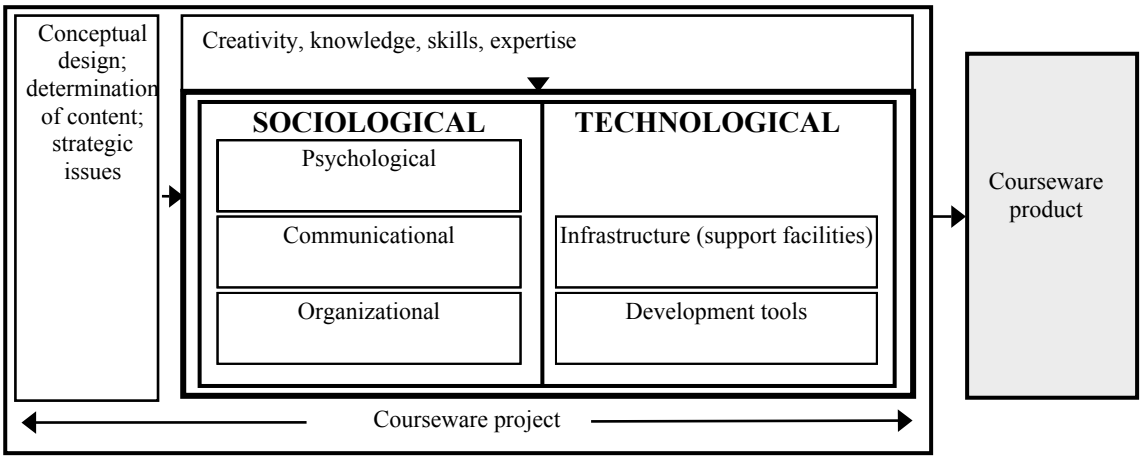


Figure 7. *Summary of observed courseware development process problems, separated into sociological and technological problems.*

Initially, all reported issues seemed to be either of a sociological nature, or of a technological nature. This may seem rather obvious at first sight, but it confirms the findings of DeMarco and Lister [DEM1987], who were one of the first (in 1987) to stress the importance of sociological problems as the prime failure of many software projects. In their book "Peopleware", they rebuked the – until then commonly held – wisdom that to improve the quality of software products the main part of the answer lay in the progress of research in software engineering. Although researchers such as Brooks [BRO1995] and Boehm [BOE1988] did not ignore the human factors in software engineering, the focus was chiefly on technical aspects. The school of thought was to improve the development methodology, and quality boost would follow almost automatically. DeMarco and Lister, however, found that far more than half of all project failures were largely due to sociological problems, and had little to do with technological issues. Their findings are now commonly accepted; the recent (from 1998 on) focus on knowledge management in IT projects can partly be explained from this perspective[7].

Having generated this view, we felt that we were now sufficiently prepared for setting up a pilot study to further investigate courseware development process problems.

4.2. Pilot study: Identifying problem categories

While initially all project experiences seemed to be of either a technological or a sociological nature, it seemed too simple to have only two categories of project problems. The initial set of interview results already indicated a much richer set of project issue categories, especially those that were of a non-technical nature. We therefore decided to design a pilot study that should help us identify a categorization of courseware development process problems; this categorization in its turn should help us in determining the direction of the solution that we can offer to address these problems.

[7] It is odd to see just how slow the software engineering industry has learned from other disciplines who had gone through roughly the same cycle, for example movie making and architecture. In both of these disciplines, though technical issues are important, they can hardly be called the main cause of failure or success for projects.

Design of the pilot study

"A good research design makes explicit and integrates procedures for selecting the data to be sampled, content categories and units to be placed into the categories, comparisons between categories, and the classes of inference that may be drawn from the data." This definition of a research design by Riffe *et al.* [RIF1998, CH.3] is often used in content analysis and excludes ambiguity as much as possible. In our case, the content consists of the interviews with the educational multimedia experts. He presents a general framework for conducting a content analysis (only the part after the problem analysis phase is reprinted here):

1. **Define the relevant content**. What will be needed to answer the research question(s) or to test the hypothesis?
2. **Specify the formal design**. How can the research question(s) or hypothesis be tested?
3. **Create analysis categories**. How will coders know the data when they see it? What units of content will be placed in the categories?
4. **Operationalize**. The heart of a content analysis is the codebook that explains how the variables in the study are to be measured and recorded.
5. **Specify population and sampling**. How much data will be needed to test the hypothesis or answer the research question(s)?
6. **Pretest and establish reliability procedures**. How can the quality of the data be maximized?
7. **Process the collected data**. Establish reliability and code content. What kind of data analysis will be used? Which statistical procedures are appropriate?
8. **Interpret and report results**. Has the research question been answered or the research hypothesis been tested successfully?

We now address each of these questions carefully in order to properly set up the pilot study.

Step 1. Define the relevant content.

Content must be reduced to units in order to be able to do measurements. Content is usually [OPP1992; RIF1998] defined in *study units*. Riffe *et al.* define them as shown in Table 5.

Table 5. *Study units in content analysis.*

Rank	Description
1.	Study units are the elements of content that are selected and defined for content analysis.
1a.	Sampling units are physical units that are selected for the study from the entire collection of content.
1b.	Recording units are elements of content that will be classified in the coding process.
1c.	Context units are content elements that should be examined in order to appropriately assign content to recording units. The context units can be the same as, or larger than the recording unit, but cannot be smaller.
1d.	Analysis units are the units that are analyzed statistically to test the hypothesis.

For the pilot study, the purpose is to identify the project problem categories within courseware projects. The *study units*, then, are all project experiences that project team members in educational multimedia projects report to the analyst (through interviews). The *sampling units* are those experiences that can be classified as "complaining" about a problem (negative), or alternatively

"enjoying" a success story, something other team members can learn from (positive). Neutral experiences such as *"the customer's project manager has left the company, so we now hope that we can continue working for the customer if our new project bid is good"* are not part of the sampling set. On the other hand, reported experiences such as *"Poor network facilities caused significant slowdown of work progress during the realization phase"* would be part of the sampling set.

The *recording units* are those elements of the sampling units reported in projects that meet the required boundary conditions as defined in chapter 3. This means that we only record experiences in projects that have a certain amount of staffing, are within a certain budget range, and that satisfy the other project boundaries. The *context units* are all factors that must be recorded in order for the recording units to be unambiguous. This is because a negative or positive project experience is often dependent on a variety of contextual (environmental) factors and seldom stands on its own. If we do not record the context units as well, the value of a reported experience may easily be misjudged (hence the mentioned importance of 'rival explanations' in an experiment). An example of a context unit is given in the following experience: *"Communication with Mr. X from company Y was very troublesome; however, we've dealt with this particular customer before and he has a reputation of being very picky about details, so we were able to take that into account"*. The context unit in this case is the known reputation of Mr. *X* of previous projects.

Finally, the *analysis units* are the units that are used as the basis for statistical analysis. Since we are interested in recording both negative and positive experiences, the analysis units span the spectrum of very negative project experiences to highly positive project experiences. In order to reduce complexity for the interview analyst, we define the analysis units as shown in Table 6. We deliberately left out the value "0" (zero), because we decided earlier that neutral project experiences were not part of the sampling units.

Table 6. *The analysis units that we used for both the pilot study and the main experiment.*

Value	Description
-2	Very negative project experience: "things went definitely wrong here"
-1	Somewhat negative project experience: "bothersome, but workable"
+1	Somewhat positive project experience: "a nice thing to happen"
+2	Highly positive project experience: "this is a real success story"

Step 2. Specify the formal design.
The tool we employed for identifying the project problem categories is the instrument of structured interviews, similar to the first set of interviews that we described earlier in this chapter. This time, however, we have structured the interviews in ten project interview questionnaires. We uniquely coded each of the questionnaires along two dimensions, namely object of interview and timeline. Nine out of ten questionnaires can be categorized as shown in Table 7:

Table 7. *Project questionnaires in time and for object of interview.*

Role	1: At project start	2: During project	3: In retrospect
A: *Project manager*	Questionnaire A1	Questionnaire A2	Questionnaire A3
B: *Project team*	Questionnaire B1	Questionnaire B2	Questionnaire B3
C: *Customer*	Questionnaire C1	Questionnaire C2	Questionnaire C3

The remaining tenth questionnaire contains general questions about the nature and context of the courseware project that is analyzed, such as customer name, estimated time path, resources allocated to the project, customer experience with these types of projects, and innovation factor of the project. This questionnaire was simply called "P" for Project questionnaire. The ten questionnaires {P, A1, A2, A3, B1, B2, B3, C1, C2, C3} together cover all possible interviews required for the pilot study. Table 9 gives a sample of such a questionnaire.

For one particular interview session, the *object of interview* becomes the entire project team – including project manager and the customer representatives involved. Sometimes an interview session was held with one person only; this was frequently the case with project managers. However, many interview sessions involved more than one team member being interviewed, and in this case each person was an object of interview, while the object of the pilot study still remained the entire project team. The *unit of study* is defined as being the entire project with all the project problems that are reported. Thus, we can have project *X* for customer *Y*, with project team members A_1 (project manager), B_1, B_2, …, B_n (team members, $n > 3$), and C_1 (customer representative). The object of interview depends on the way the interview sessions are organized. Objects of interview may possibly be $\{A_1\}$, $\{B_1, B_2, B_3, B_4, B_5\}$, and $\{C_1\}$, but in reality any combination of the roles involved is possible.

To further improve the quality of the pilot study, we decided to clearly define just what it is that we are attempting to analyze: the functional context, geographical context, and organizational context. These are the context units of our study. This helps us to look at the pilot study from various angles. To identify our context units, let us now examine the functional, geographical, and organizational context of our interview sessions.

The *functional context* of the pilot study (meaning "what is the function of our study") consists of the project problem categories that we attempt to identify. This is because the objective of the interview sessions is to identify the frequency of occurrence of the project problem categories involved, and later, to determine if these frequencies of occurrence change in time when employing our knowledge management solution. We note here that we will ignore project experiences that clearly cannot be solved by any knowledge management solution, such as hidden political agendas.

The *geographical context* of the pilot study consists of the geographic locations between the various roles {A; B; C} involved on the one hand, plus the person conducting the interviews on the other hand. In the case of this research project, the researcher conducting the interviews always traveled to the location of the person being interviewed. The entire project team was always working together at one location.

The *organizational context* of the pilot study consists of the various roles that we have identified {A; B; C}, the procedure we employ to conduct interviews with these roles, and the implications this has for the design of each of the interview questionnaires. Although we attempt to address all existing project problem categories in an interview, the nature of the questions asked is slightly different for each role.

Similar to our first set of interviews, we structured all the questionnaires according to the theory by Oppenheim [OPP1992]. An important requirement for the interview questionnaires was that we should not assume anything about the answers that would be given, and incorporate these

assumptions implicitly in the questionnaires (for example, a question such as "*why was communication so poor?*" should not be one of the main questions, although it may be asked at the end if this turned out to be a particular awkward point). On the other hand, we wanted to make sure that all the project problem categories that we had identified were treated in the questionnaire. We therefore made sure that there were questions involved that explicitly addressed these categories, but with an open view to other possible mentioned project problems. Thus, the structure of each of the questionnaires is set up as described in Table 8.

Table 8. *High-level structure of the project questionnaires.*

Part	Section title	Section description
1	*Naming data, Date data*	Person's name, project name, customer name, dates
2	*Description of tasks*	Assigned role(s); previous experience; things that are new
3	*Project problem categories*	The analysis categories defined in step 3.
4	*Other problems*	Things that were overlooked or not discussed in previous section.
5	*Critical success factors*	The critical success factors for this project and how they may be achieved more easily.
6	*Suggestions for improvement*	Tips of person being interviewed for improving the project and these kinds of projects in general. This section also includes suggestions for improvement on the metric itself.

In Table 9 we give a sample of one of the project interview questionnaires, namely B2: the project team, during the project. This was the questionnaire that was used most often. More sample sets of interview questionnaires have been included in Appendix A. In Table 9 and the corresponding appendix, the last column denotes the direct relationship to an online database question. All interview questions had been mapped to a corresponding database question, and the last column made it easier for the researcher to convert the offline interviews to online data, although it was not used for other purposes pertaining to the pilot study. The objects of interview were never present at this process, usually because there was no budget reserved for such activities.

Table 9. *Extract of interview questionnaire B2: Project team during the project. See also appendix A.*

Nr	Question	Database reference
00	Description of tasks for this particular project phase:	
01	Experience with managing multimedia projects: (0: none, 5: highly experienced)	person→mmexperienceind
02	Is the project still going according to project planning?	
03	Are there any contextual environment factors for this project that may have a considerable influence on the rest of the project? If so, which ones and can they be solved?	experience→[all] category=wo
04	What are current problems in managing this project?	project→management
05	What are currently the most important communication problems within the team?	project→communication
06	What are currently the most important problems with project meetings?	project→meetings

Step 3. Create analysis categories.
For the pilot study, step 3 is harder to define because it is the goal of the study to identify the problem categories in educational multimedia projects. However, we must be careful not to mix up problem categories with analysis categories. For the purpose of the pilot study, we will define the analysis categories as all problem types that may occur in courseware development projects. These include, among others, political issues, interdisciplinary communication issues, technological issues, customer-related issues, and project management issues. The complete set of analysis categories, however, is not known for this study. We have attempted to implicitly address a large set of analysis categories in the project interviews, without steering employees that were interviewed too much to a category, although there is always a "field of tension" between these two, and a certain amount of influence cannot be avoided.

Step 4. Operationalize.
The instrument that we use to operationalize the pilot study consists of the interview questionnaires as defined (and partly described) in step 2. These questionnaires are used in the interviewing sessions according to the following procedure (in the following, the terms "researcher" and "analyst" are synonyms):

1. Researcher determines the role(s) of the project team member(s) that will be interviewed. (The available roles are defined in the PROMISE handbook for educational software projects.)
2. Researcher determines the phase in which the project currently is: startup, ongoing, or in evaluation phase. This categorization of project phases was defined in step 2 of the study.

3. Researcher selects the appropriate interview questionnaire for this particular interview session. This is one of the questionnaires as defined in Table 7. In addition, questionnaire P is taken along if this is the first time the project is interviewed.
4. Researcher explains the purpose of interview to persons being interviewed. Questionnaire P is handled first to note some household data about the project to capture context units (first time for each project only).
5. Researcher starts asking questions in proper order of the questionnaire. The conversation may deviate from the actual question being asked if an important point is touched upon.
6. At the end of the interview, researcher determines if all questions in the interview questionnaire have been addressed. If not, these questions are asked after all. This stimulated people in elaborating on issues they would otherwise perhaps have forgotten.
7. Researcher enters the interview results in readable format into a word processor document.
8. Researcher sends the document of step 7 to the person(s) that has (have) been interviewed. Possible corrections are sent back and are incorporated.
9. The interview results are entered into the project experience database, which we describe later in this chapter.

Step 5. Specify population and sampling.
This step is defined as "How much data will be needed to answer our research question(s)?" Since this is a pilot study, we do not attempt to answer the hypothesis here yet. For this study, the research question is about how many and which project problem categories may be identified in courseware development projects. The population for sampling that we define is also partly dependent on the possibilities at the company of study. Not all courseware projects apply for analysis.

Most literature suggests, as a rule of thumb, a minimum sample set of six units of study [NIE1996; BAA1995]; in our case this means six projects to interview, with several interviews for each project. We analyzed twelve projects in all, which turned out to be possible within the company of study given the research boundaries and constraints, over the course of more than two years.

Step 6. Pretest and establish reliability procedures.
The pretest of our pilot study is described in section 4.1. It consisted of a small set of informal interviews to come up with an initial model of project problems. That set of interviews was not part of a carefully planned empirical experiment such as this pilot study; however, it did help us to clarify the scope for the pilot study.

The reliability procedures for the study include the boundaries, assumptions and requirements that we defined for the projects to be analyzed. The purpose of the reliability procedures is to minimize the margin of error that can occur when conducting interviews across a "large" set of projects. An example of a boundary is that project budget is larger than a certain minimum but not larger than a certain maximum. An example of an assumption is that the content of the product to be built is reasonably stable or that it is safe to assume that it will become stable on a short notice. An example of a requirement is that all required disciplines are in place; so no visual designer is missing if one is full-time required. Due to the business environment in which this research was conducted, we were not able to set any further (more formal) reliability procedures.

Step 7. Process the collected data.
The sixty interviews were carried out from 1996 until and including the year 2000. All in all, twelve projects were analyzed using the structured interview questionnaires. The author carried out and processed all interviews. The answers were then grouped according to the nature of the answer. There were no predefined guidelines to do this; it was done on the basis of answer-to-answer comparison, and based upon the initial model of project problems that we devised as shown in Figure 7. We provide a summary of the results from the sixty interview sessions in Table 10. Rather than listing hundreds of interview answers, we have already roughly grouped the answers, with the risk of influencing the final categorization process too much.

Table 10. *Reported project problems after the first set of interviews.*

Planning interdisciplinary co-operation	45 out of 60
Stress because of poor project planning	34 out of 60
Organizing the availability of skills, tools, methods	29 out of 60
Re-inventing the wheel for the x-th time	25 out of 60
Informal communication and meetings	54 out of 60
Listen to ideas brought up by other disciplines	53 out of 60
Conceptual design versus technical design	28 out of 60
Instable tools with rapid succession of new versions	41 out of 60
No time to become acquainted with a tool	32 out of 60
Customer fails to deliver content on time	55 out of 60
Customer has difficulties estimating the required effort for a wish	33 out of 60

The numbers given in the second column mean that in *x* of the 60 interviews, that particular problem was found to be present. Please note that the picture is consistent with Figure 7, and the point made by DeMarco and Lister [DEM1987] that by far most project problems are of a non-technical nature.

The objective of the pilot study was to create a categorization of project problems experienced by educational multimedia project teams involved in courseware projects. We aimed for five to six categories – on the one hand, this provides us with a sufficient number of placeholders to store distinct problems, and on the other hand, this was felt to be a manageable number of categories when analyzing project interviews. We decided upon the following categorization of project issues.

PR: Project management issues
QU: Qualification of skills
CO: Communication issues
TE: Technical issues
PS: Psychological issues
CU: Customer-user relationship issues

We now define what we mean by each of these categories. For the main experiment, described in chapter 6, a formal codebook was developed to further explain what each category includes and does not include; here, we suffice to present a summary from that codebook.

Project management (short code: PR)
Often, the term project management is seen as a task solely for the designated project manager of a project. Other definitions see project management as a group responsibility: the success of the project is a result of the effort of the whole team, and this explicitly includes project management tasks from that whole team. This category agrees to the latter view.

Qualification of skills (short code: QU)
This theme is about the skillfulness of all project team members. Often, project process problems can occur because one or more team members are simply not yet experienced enough to have foreseen that problem. Since almost every educational multimedia project has junior employees, this category is a valid one.

Communication (short code: CO)
In the end, one might say that almost all problems are due to poor communication: if we would communicate in a perfect way all the time, we would not have any problems. But that does not hold true, because even if we communicate perfectly, we still cannot foresee all pitfalls on the way ahead. However, since we are unable to communicate perfectly, this category occurs all the time.

Technological (short code: TE)
With this theme, we refer to problems that are of a direct technological nature in the sense of technology: a network that does not function properly, development tools that are not robust, slow connection to the Internet, conversion utilities that do not work well, and so on. You should be able to "touch" the cause of the problem.

Psychological (short code: PS)
This category is one of the hardest to identify, let alone define. When a project is in danger of running out of budget or time, a certain amount of stress occurs, not just with team members, but for the entire group, too. Perhaps some team member feels that things are going completely the wrong way but is afraid to speak up for it. These kinds of things are categorized as psychological problems.

Customer/user (short code: CU)
The classic example of a problem of this category is the customer who fails to deliver content on time, even if that deadline was formally agreed upon. A customer's expectations should be properly managed, which requires a special form of expertise and experience. Also, many customer-related issues are ultimately caused by communication problems. If, for a particular problem, there were aspects that are due to the acting of the customer, we consider that problem to be a Customer/User problem.

4.3. The first step in addressing process problems

In what way can the project problem categories that we found in the pilot study help us to address these issues? The needs for improvement reported above are about improving the level of professionalism, or maturity[8], of the project process. Clearly, if we want to improve the level of maturity of courseware development projects by offering a solution that meets the needs that we list above, that solution must at least take care of the problems identified in the pilot study. It must eventually lead to more positive experiences on (preferably all) project problem categories.

According to the Capability Maturity Model [HUM1995] that we discussed in section 2.4, the maturity of a software process is on level one (the lowest level) if there are no formal procedures for keeping track of time, cost, effort, and quality. No measurements are performed to assess the quality of the process; subsequently, there is no way of guaranteeing improvements. It is no surprise that this level of maturity is called the *Ad-hoc* level. While this may seem a rather non-professional way of working, according to market analysis organizations such as Gartner, many organizations worldwide are still incapable of fully reaching CMM level 2. On that level, if a project is a success, the reasons for that success can often be readily identified, and the success is likely to be repeatable in the future. Therefore, this level is called the *Repeatable* level.

Many organizations are still operating somewhere between levels 1 and 2 on the CMM scale. At the start of the research project, the courseware development unit at our company of study had a basic project management method in place, called PROMISE, in the form of a paper handbook. This handbook was developed especially for educational multimedia projects, and is targeted towards a multidisciplinary project team and instructional deliverables. PROMISE describes basic project phases, roles, and *objectives* to be achieved for each phase and role. Each objective is supplemented by activities for a particular role. Although PROMISE was widely used at the company of study for educational projects from 1988 on, its paper-like nature made it difficult to be used effectively by the various team roles in the various phases; this resulted in the observation that it was especially the project manager employing the manual, much more so than the other team members. In addition, version control was not easy. Moreover, best practices for many objectives and activities could not be linked to their description in the handbook (though templates could). Similarly, knowledge, expertise and experiences were very hard to share because of the lack of a central knowledge repository that was accessible to everyone and kept up-to-date.

As in every organization, employees were sharing knowledge and experiences all the time during projects, but that process was in no way formalized or explicitly recognized: it was mostly done in informal chats during short breaks, or, at its best, in informal technical meetings that focused on one particular issue; these meetings were usually held in the evening hours. A CMM assessment identified the level of maturity of this local unit as being close to, but not quite on, CMM level 2.

From the pilot study and the pre-pilot interviews in section 4.1, we identified the following needs for improvement, ordered by decreasing degree of occurrence in the interviews:

[8] Although there is much debate on the difference between "professionalism" and "maturity", for the purposes of our research we consider them synonyms.

1. "Know what other colleagues know": Easy access to all available knowledge and expertise of all employees at any time, at any place;
2. Project "experiences" should be easily available for later re-use, for example to avoid making the same mistake in later projects;
3. Standardization of the educational multimedia project process and support for that way of working;
4. Access to available individual estimates and consensus estimates, for use in project tenders.

Everyone at the courseware development unit saw addressing these needs as a critical success factor. The following structural benefits were envisioned once these solutions would work correctly, again ordered by decreasing degree of priority (from interview sessions with experts of our local unit at the company of study).

1. More mature project process, in the sense of the Capability Maturity Model
2. Sharper project bids
3. Stop "re-inventing the wheel"
4. Stop the loss of knowledge and expertise, especially when employees leave the company
5. Ability to bring new employees up to speed in a quick and easy way.

To address the needs, we should design a solution that meets as many of the needs as is reasonably possible within the time frame of the research project. Clearly, our solution must provide for a way to formally define the courseware project process, and be able to store various types of information, knowledge, expertise, and experience (and we must define what we mean by each of these). This leads us to the conclusion that a knowledge management solution (with "knowledge management" as defined in section 2.4) is probably most suitable to address the needs for improvement; this is because virtually all desired needs for improvement are of a knowledge-based character. Subsequently, our knowledge management solution must be targeted towards addressing the process problems that we identified. We will investigate this in chapter 5.

An important part of the process problems appeared to be the inaccessible character of the knowledge and expertise that was available amongst the employees of the local unit of our company of study. There were senior, "medior" and junior employees on all disciplines available, but the sharing of knowledge and experience was barely organized: every once in a while a special topic discussion evening was organized, but this was of an informal character and was not formally organized. Consensus estimates were not explicitly available except one about the amount of money a customer would approximately have to spend on one hour of basic browsing CBT. All expertise and project experience were available only in the heads of the individual employees, or at best in files in people's desks.

At a relatively early stage of our research (the fall of 1996, after the initial interviewing sessions with the zero-questionnaire) we asked all employees (some forty) what they thought about having available a freely accessible database containing various types of project experiences on projects that they and their colleagues had done in the past. There was not one employee who did not like the idea, though some wondered if such a solution would be easily achievable, given the experiences that they had themselves with sharing knowledge and experiences in the past years. Some initiatives had already taken place in the years before, but all of them had silently "bled to death". However,

now that we had a clear list of wishes for improvements of all employees, we could set up a list of actions that would have to be undertaken in order to start working on the solution:

1. Set up an online freely accessible instrument that enables the experts to categorize, store, and find expertise and experience of employees. Experiences could be in the form of interview data, and expertise could be in the form of a list of expertise areas (in other words, qualification of skills).
2. Gather these experiences and convert them to digital format in order to be accessible on the aforementioned platform.
3. Set up a stable procedure to ensure continuous gathering of these project experiences (in other words, organize the continuity of experience gathering and usage).
4. Stimulate use of this solution by enthusing people through interviews and workshops.

We were then faced with the many aspects of the words knowledge, expertise and experience. Of course, we have chosen a working definition for the term "knowledge" in section 2.4. In a way, *expertise* can be seen as the union of the *knowledge* and *experience* a person has [HAT1995]. So we had to design a solution to capture both the knowledge of employees as well as the experience of these employees. We decided to design an instrument that would capture the project experiences of these people, as well as which domain areas they were expert in. This was called the ICOM experience database, or experience database for short. The experience database was implemented early 1997 on a Windows NT server. This server could generate HTML (HyperText Markup Language) files using Server-side Visual Basic script through a mechanism that was called IDC-HTX, which would later evolve into ASP (Active Server Pages). The very first version of the multimedia experience database was completed early 1997; we present a screenshot of this version in Figure 8.

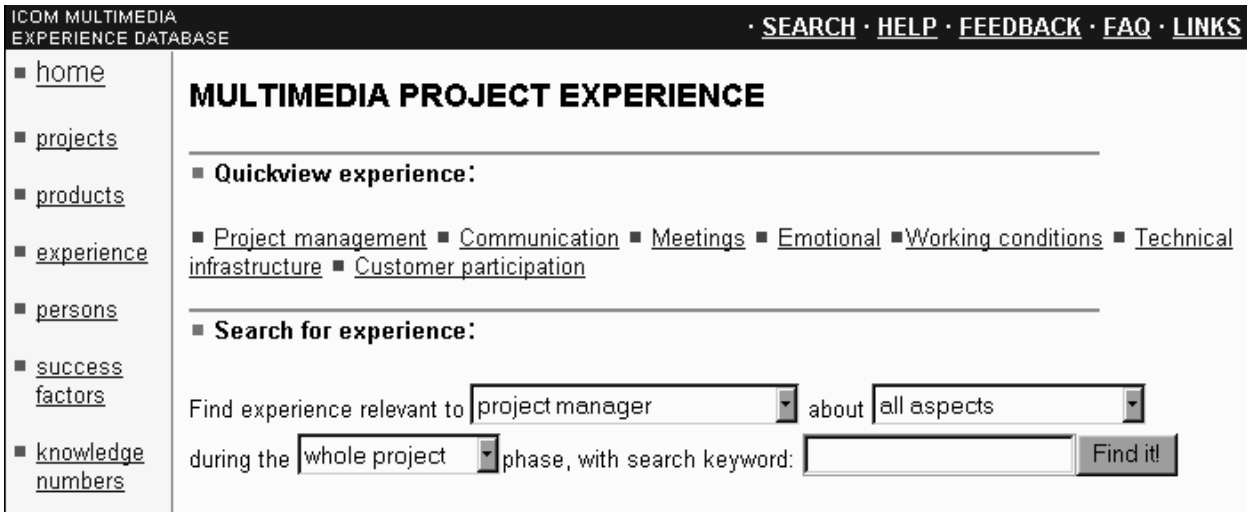


Figure 8. *Early screenshot of the ICOM experience database (1997).*

We structured the database for the experience database as shown in Figure 9 (the diagram is shown in Information Engineering notation), in which we captured experiences about both the process (projects experience data) and the product (product experience data). In practice, however, it turned out that the experience data about the process was by far the most important. In Figure 9, a project can deliver one or more products. Four or more employees staff a project, and an employee can be involved in more than one project. A project results in experience data that is relevant for more

than one project. A product is made for one customer, and a customer can commission the delivery of one or more products. A product results in experience data about that product, which is relevant for similar products.

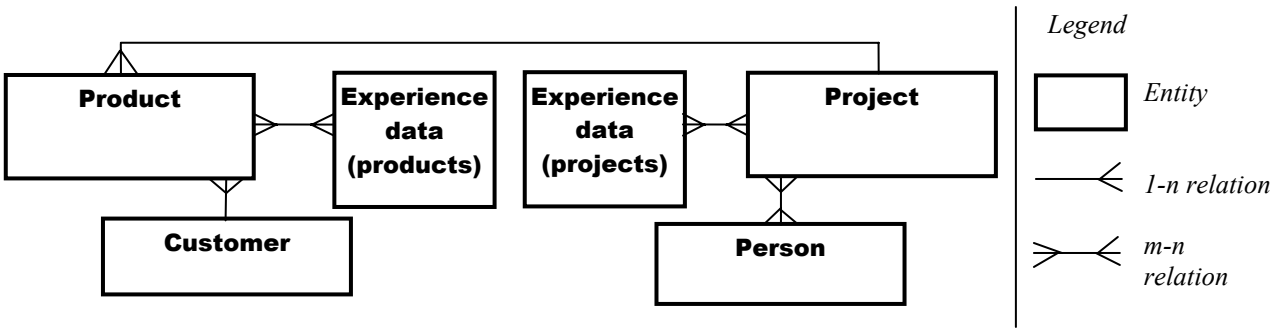


Figure 9. *Global data model for the first version of the ICOM experience database (1997).*

4.4. Conclusion of problem analysis

We now look in retrospect to the pre-pilot interviews and the pilot study interview sessions. We may ask ourselves what are, on an abstract level, the most important key causes for the occurrences of the problems that we found? When taking a step back and looking at the process problems in courseware projects that the company of study experienced, we have observed that the level of professionalism, or maturity, of the way of working caused most of the reported problems. In particular, we observed the following:

1. The process of educational multimedia projects was not well defined. The project phases, project roles, and general responsibilities and tasks were documented [HAR1988] but this document was usually either employed very loosely or not at all. There was no clear bird's eye overview of what the project process looked like; therefore, few people formally employed it.
2. There was no organized way of controlling the acquisition, dissemination, use, and archiving of the knowledge and experiences of the educational multimedia experts. In other words, knowledge management was not employed.
3. There was no metric program that allowed the organization to assess the basic process and product quality at timestamp *t*, and measure possible improvements at timestamp *t*+1. In other words, these are no efforts on software process improvement, and the company of study had no organized way of determining whether or not any improvements were made.

By addressing these causes, we can ultimately improve the level of professionalism, or maturity level of the development process of courseware at the company of study. To realize this, we concluded that our solution should be in the form of an instrument to facilitate courseware experts in their knowledge needs. By conducting well-defined measurements on the effectiveness of this instrument, we can identify the desired project process improvement.

In this chapter, we have investigated the project process of developing courseware at our company of study. We have identified a set of project problem categories and we have found that the needs for improvement can be seen as a desire to improve the level of maturity of the project process. In this case, this need is probably best met by a solution for formal knowledge management. In

chapter 5 we describe the knowledge management solution that was eventually devised to address the needs for improvement described above. We also construct a metric to be able to conduct measurements on the usage and effectiveness of the knowledge management solution. Subsequently, we set up a main experiment to test the hypothesis that our knowledge management instrument does indeed help to improve the maturity process of the educational multimedia projects. We describe this experiment in chapter 6.

5. Proposed solutions

In this chapter we investigate ways to improve the maturity level of the project process by designing an instrument to facilitate courseware development teams in their information needs. In addition, we describe a metric program that allows us to measure any improvements.

The project experience database described in chapter 4 allows employees to quickly locate pitfalls on a variety of project problems. Project experiences can be recorded and disseminated to the entire group of multimedia experts. But with this tool in place, the project process was still not described in an unambiguous way. There was no central knowledge repository to store and find the best practices and best templates for the various objectives. In other words, there was a need for an integrated solution for knowledge management. Of course, the department of study was not the only department at which this need was felt. Within Atos Origin, efforts for setting up knowledge management solutions properly were already underway, and we gratefully employed them. The most important of these efforts is called the KnowledgeWorks model for knowledge management.

5.1. A method for knowledge management implementations

In section 2.4, we have provided an overview of references on setting up knowledge management in organizations. The company of study has built on such research [DAV1998; NON1995; WII1995; WEG1997] and existing practice to develop its own methodology for knowledge management in IT projects. It was developed internally within the company of study in 1997 outside the scope of this research project. Three years later, it is still used widely in IT projects in which knowledge management plays a prominent role. The methodology is called KnowledgeWorks, and it is mainly built upon the (now commonly accepted) notion that knowledge management is more than merely installing a workflow tool as a central repository for storing knowledge [DAV1998]. KnowledgeWorks has been successfully used in knowledge management initiatives since 1997. The KnowledgeWorks methodology looks at knowledge management from three main points of view (or three dimensions), which we will now discuss in some depth.

1. *Four aspects* of managing knowledge: Organization, Culture, Content, and Infrastructure;
2. The nature of knowledge dissemination: "flow" knowledge and "stock" knowledge;
3. The degree of knowledge management *formalization*: from "loose" communities, to formal environments for managing knowledge.

Four aspects of managing knowledge
The KnowledgeWorks methodology holds that there are four crucial aspects of knowledge management. If these aspects are well balanced (*balance* is the keyword here), a knowledge management implementation has a significantly higher chance of succeeding. If any one of these aspects is out of balance, the chance of failure increases exponentially. Thus, the balance can be very fragile, and moreover, the weight of each of the four factors can be different from the others. This implies that there are no "theoretically optimal" values for the budget for the four aspects, not even relatively (in percentages) speaking. The four aspects are:

1. **Organization**. This is the point that is most often forgotten or neglected when setting up knowledge management within an organization. The organization aspect includes procedures for starting up knowledge management, for keeping it alive, and for keeping it

controllable and under control. It is within this dimension that the following roles and their corresponding responsibilities are defined. Usually there is a so-called "champion" role with "motivational" tasks; a "moderator" as a leading role with the highest level of responsibility, and a "content owner", who looks after the (quality of the) actual knowledge [VAA1997]. In addition, there is often a *Knowledge User Group* and a *Control Board* for requesting and approving, respectively, new elements of the organization of the knowledge management. The organization aspect also has an umbrella function over the other three aspects mentioned below, because the other aspects must all be organized as well: the gathering of content must be organized, as well as keeping people involved and enthusiastic; the infrastructure must also be organized.

2. **Content**. Any knowledge management effort without a good grip on the actual knowledge and expertise (which we shall summarize as *content*) that is available is doomed to fail. It may be true that people are the most important entity when managing knowledge, but it is actually the content that is in people's heads and in people's desks that must be organized and managed. There must be a certain minimum amount of content already available by the time the knowledge management solution is implemented [VAA1997; DAV1998]. Therefore, this knowledge must be made explicit and visible. It must also be identified and categorized. Moreover, some knowledge becomes obsolete and this must also be visible. Content always has its own particular status. Identifying this may require a lot of effort.
3. **Culture**. This is probably the most difficult dimension to capture. Culture includes the motivational reasons for people to co-operate in sharing their own knowledge and using knowledge of others [DAV1998; WEG1997]. The latter point is usually easier to achieve than the former point. Many employees derive a certain degree of "power" or "status" from their own knowledge (this also relates to the "not-invented-here" syndrome). After all, somebody with a lot of specialized knowledge becomes more valuable in the organization. Sharing each other's knowledge can therefore mean a loss of power, which is a risk not many employees are willing to take (initially). Therefore, a high-quality communication plan should be set up, including presentations, hands-on workshops and participation sessions. Bit by bit, skepticism (or even fear) should be turned into enthusiasm. This difficult task can often take up the majority of the total budget required for setting up knowledge management, depending on the type of organization.
4. **Infrastructure**. Deliberately mentioned as the last dimension, the infrastructure includes the software package(s) and other technical components that enable online use of the knowledge management solution. Too often, setting up knowledge management is seen as putting the infrastructure in place, based on the assumption that employees will automatically use it once it is available. This is also one of the main reasons that knowledge management efforts often fail [DAV1998; WEG1997] The software package may be a solution offered by a commercial company, or it may be a solution developed in-house. The choice between these two is not unimportant, but it usually requires less effort than organizing culture and content. This is not to say that the infrastructure component is a trivial one: many aspects concerning browsers, security settings, authorization policy, network connectivity and application performance must be taken into consideration. Analogous to the other aspects, if everything is taken care of except for the infrastructure component, the knowledge management solution will probably still fail.

These four aspects of knowledge management are all intertwined, as we show in Figure 10. The infrastructure cannot work without the necessary content in place and without employees that are

willing to use it. Since a knowledge management implementation often requires a culture change, this must be carefully organized, as must the gathering and categorization of content. The fact that the four aspects are intertwined, however, does not mean that each of the aspects weighs equally heavy regarding the available budget. We will see that for a knowledge management implementation the "culture" aspect draws most heavily upon the available budget (see also chapter 7).

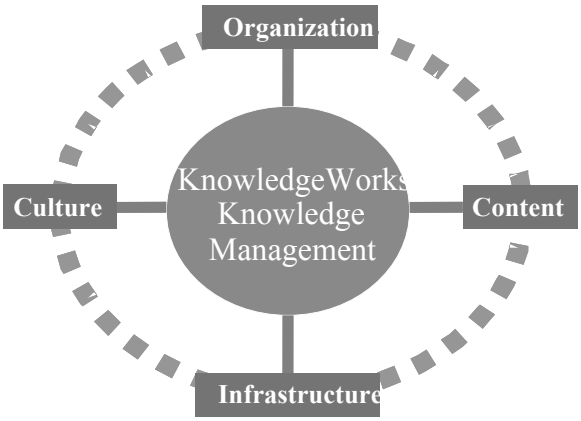


Figure 10. *Four intertwined aspects of knowledge management according to KnowledgeWorks*

Flow knowledge versus Stock knowledge

KnowledgeWorks distinguishes between two fundamentally distinct types of knowledge that are relevant within organizations. These types are the *tacit* knowledge on the one hand, and the *formal* knowledge on the other. These types are also found in published literature on knowledge management [NON1995; RUG1997; WEG1997]. Tacit knowledge has an informal, "flow"-like character, is hard to make quantitative, hard to capture formally, and is usually disseminated through informal conversations in the hallway or through informal coaching by senior employees. Discussion forums usually disseminate tacit knowledge; they result in employees "knowing" things that they cannot express in numbers. *Formal* knowledge, on the other hand, has a defined state, has a "stock"-like character, can often be expressed in numbers or other quantifiable units, and is surrounded by formal procedures for gathering, categorizing and disseminating that knowledge. It includes formally approved templates, best practices, and generally accepted consensus estimates.

The important thing to remember here is that according to the KnowledgeWorks methodology, both types of knowledge are indispensable for a smoothly working knowledge management organization. Knowledge that is shared and disseminated through informal conversations and coaching is important, but it will remain on a fuzzy, hard-to-identify level, which makes it hard to really turn the power of knowledge into a competitive edge, which is ultimately what organizations aim to achieve. Conversely, setting up knowledge management on a highly formal, defined level without facilitating – or at least allowing – room for implicit, *tacit* knowledge sharing, will result in an inflexible, bureaucratic organization in which people will not be motivated to share anything at all regarding knowledge [HAT1995; VAA1997]. The company of study has explicitly provided for facilities for both types of knowledge, which we will investigate later in this chapter.

The degree of knowledge management formalization

A final distinction we make is the degree of formalization of managing knowledge in teams [see also NON1995; WEG1997]. One can – in principle – distinguish as many degrees of formalization as desired; we suffice by mentioning three, shown in Figure 11, which shows that a higher level of formalization does not imply leaving behind the previous level of formalization.

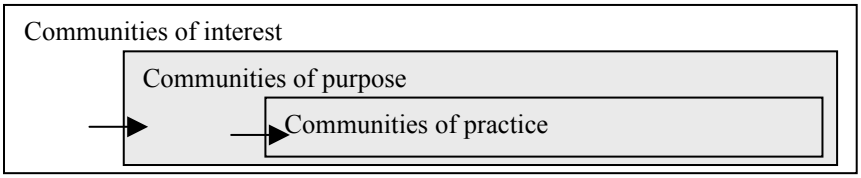


Figure 11. *Three degrees of formalization in setting up knowledge management according to KnowledgeWorks.*

On the lowest level of formalization, we find that groups of people who share similar interests start finding each other, through publications, seminars, or by means of the corporate intranet. We call these groups *communities of interest.* On this level of formalization, there are no procedures in place for gathering, categorizing, dissemination, and quality assurance of knowledge and experience. At this stage, the group is especially busy getting to know each other and each other's interests and expertise. Someone might put some individual effort in setting up a solution to facilitate discussion, such as a public or private newsgroup. Links to websites are exchanged, and perhaps a mailing list is set up. This level of formalization can work in all types of organizations, but will typically be more effective in smaller companies.

On the next higher level of formalization, the need for structuring the knowledge management dissemination efforts becomes clearer. The e-communities start to organize themselves into a more formal state of existence. On this level we see *Networks of professionals* emerge (sometimes called *communities of purposes*). If these networks are lucky, formal budget is now allocated for employees; this budget allows them to spend some of their monthly working hours on knowledge management of their particular expertise. Within the company of study, on this level of knowledge management formalization, each employee was given four working hours each month to be spent freely on knowledge management activities for their particular expertise. This illustrated commitment of the highest management to the importance of knowledge management as a means for leveraging the intellectual capital of the company. Moreover, a large budget was spent on implementing an online solution for sharing "flow" (tacit) type of knowledge. This is called the N.o.P., short for Networks of Professionals[9], shown in Figure 12. This technical solution facilitates a discussion forum to discuss various relevant topics; it also contains a library of knowledge elements in progress. These are usually non-definitive documents and references to possibly interesting work. Employees can rank the quality of each other's contributions. As for the organization aspect, a certain status of "membership" is introduced, though in principle everybody is free to join, as long as he/she is involved in the particular field of knowledge of that Network of Professionals. On this level, no formal definitions of the way of working are incorporated. If we relate this to the Capability Maturity Model that we discussed in section 2.4, we observe that this level of formalism is typically found on level 2, the repeatable level. Level 3, the defined level, requires the highest level of formalism in the sense that we observe it.

[9] The Networks of Professionals, both on organizational and technical level , were implemented outside the scope of this research project.

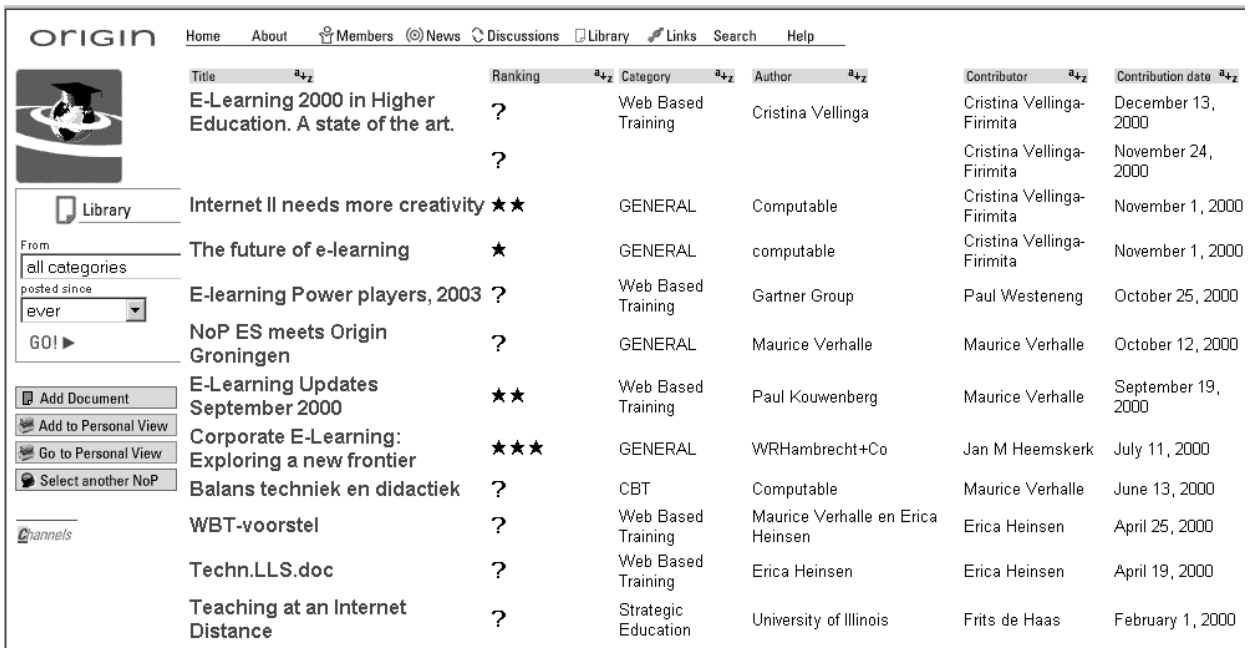


Figure 12. *A screenshot from the Networks of Professionals.*

On the highest level of formalization, the management of knowledge can be mature. The most important prerequisite for this level is that the way of thinking, the way of modeling, and the way of working are all formally defined and approved of by all involved. There is a formal organizational budget for organizing this level of formalization. Contrary to the level described above, on this level the access to the knowledge is not any longer open to all. Not only must an employee be interested in that particular area, but he or she must also be involved in (important) activities regarding this matter, and be actively involved in contributing to the body of knowledge available on that area. Also, there is a formal model for the maintenance of both the content (the actual knowledge) and the technical infrastructure that supports the dissemination of that knowledge. There is a *knowledge repository* (or *library*) available that contains formally approved knowledge elements only. These knowledge elements are usually defined templates, best practices of software applications, approved presentations, and references to other (scientific) literature. In addition, individual estimates are gathered to be able to construct consensus estimates out of these. Finally, if properly set up, this level of formalization also has a measurement program in place to be able to determine if any improvements are realized in due time.

The KnowledgeWorks model for implementation of knowledge management seemed to fit the needs for improvement that we identified in the problem analysis well, since we observed that these were primarily about knowledge needs. At the company of study, we implemented a solution for knowledge management using the KnowledgeWorks methodology. The resulting tool for supporting networks of professionals is called NOP. The resulting tool for formal knowledge management is called Performer; it facilitates the defining of the way of working, and offers possibilities to connect formally approved knowledge elements to objectives defined in that way of working. Of all levels of formalization, Performer is connected closest to this research project. We will therefore describe Performer in some detail in the next section.

5.2. Functional design of Performer

Performer was initially conceived to be a workflow tool on an operational level, for the project team involved in projects where PROMISE was used as the project handbook. Later, Performer turned out to be a solution for formal knowledge management for these kinds of projects as well. By the time Performer had to be actually implemented (see section 5.4), the knowledge management aspect had become the prime factor of importance; however, at the start in 1997, it was thought of as a way to "automate" the PROMISE project handbook [HAR1988]. This means that Performer models the project phases, project roles, and activities for each project role in each project phase. Activities were grouped into so-called objectives (targets). These were, initially, the main concepts of the design of Performer. Later, additional Performers were implemented for other project areas, notably data warehousing, desktop migration services, and SAP. In this chapter, we only address Performer for educational multimedia projects, the Education Performer. The phases and their corresponding sub-phases were translated almost identically from PROMISE, with some slight adjustments. The PROMISE phases for educational projects are:

Table 11. *Phases and sub-phases for Performer.*

Phase	Corresponding sub-phases
Conception	1. Arouse Interest; 2. Make Bid and Contract
Initiation	1. Staff the project; 2. Set up project
Realization	1. Analysis; 2. Design; 3. Realization; 4. Test and Accept; 5. Implementation and product evaluation
Closure	1. Assess project; 2. Evaluate project; 3. Archive

Not all project phases are equally relevant to the various project roles. For example, a system developer is usually not involved in the Acquisition phase[10], and the project consultant does not have much to do in the Realization phase. We needed, therefore, a quick view showing the relative importance of each project sub-phase for each role. The Promise roles for educational projects are (in no particular order):

- Commercial manager (or *account* manager)
- Consultant (or *business* consultant)
- Project leader (or *project manager*)
- Professional: content (or *functional* or *didactical* designer)
- Professional: design (or *graphics* or *interaction* designer)
- Professional: technics (or *programmer* or *scripting programmer*).

This categorization of roles corresponds with the most global of team role descriptions found in chapter 3. The "Professional: content" is a person with either a didactical or communication background; the "Professional: design" is a person with graphical and/or interaction design expertise; the "Professional: technics" is usually a programmer.

[10] Without any judgement about whether or not this is correct.

The company of study also wanted an additional view on the tasks to be achieved in each project sub-phase, regardless of the role in the project team. This view is called the *aspects* view, and is set up like the roles view. The PROMISE aspects for educational projects are (in no particular order):

- Culture (company culture, geographical culture)
- Content (subject matter of the subject domain involved)
- Project Management
- Organizational (procedures, roles, responsibilities)
- Technical

Although some employees doubted whether the aspects view was such a different view from the roles view, it was adapted as a separate matrix. In Figure 13 the roles view and the aspects view can be switched by selecting the desired view in the dropdown box at the left top corner of the matrix.

Each project role is able to get a quick overview of the most important project sub-phases for his/her particular role, merely by a quick glance at a matrix made up of these project phases and roles. This matrix is called the *Objectives* matrix, because of the contents of the cells: each cell shows the number of targets (objectives) to complete before moving on to the next cell.[11] Each objective is achieved by carrying out a list of activities, including prerequisites and deliverables. The objectives matrix for educational projects is as follows.

[11] This does not imply that Performer, or Promise for that matter, is built upon the "Waterfall" way of working: in Promise, special sheets were available showing the dependencies of any objective to any other objective. Within Performer, html limitations erased that possibility, at least graphically speaking.

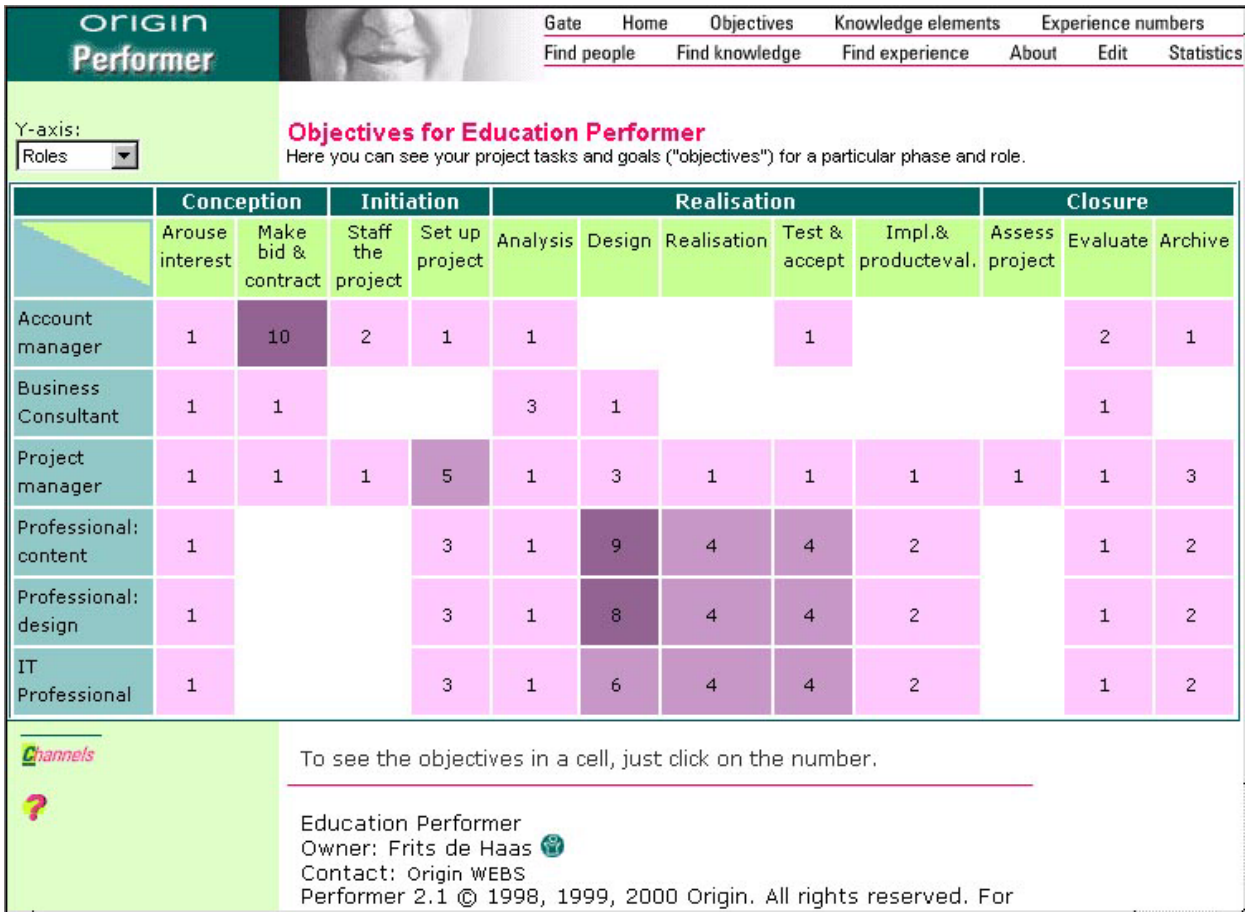


Figure 13. *Objectives matrix for Education Performer.*

Let us consider Figure 13. On the x-axis of this matrix, the phases and their corresponding sub-phases are defined. The x-axis is also a time bar, because projects usually start with arousing interest of a customer and end with an evaluation and archiving of valuable deliverables. Especially in the Realization phase though, various iterative loops amongst the sub-phases are possible. So the x-axis does not imply any compulsory development methodology such as S.D.M., or D.S.D.M. [STA1997].

On the y-axis of the matrix, the project roles are defined. Later, an additional y-axis (with the same objectives cells, but ordered differently) was added; we call this the "aspects" view, which is briefly discussed on the previous page. Within the objectives matrix, each cell denotes the number of objectives to be met by each team member role in each project phase. The deepness of the pink/purple color of the cell corresponds directly to the number of objectives to be met. A white cell implies "no objectives for that role for that sub-phase". This way, any project team member can instantly get an idea of which sub-phases are most important for that role[12]. By clicking on any of the pink/purple cells in the matrix, a list is generated describing all objectives in that cell, including a brief description. An example of this is shown in Figure 14.

[12] This is only partly true. While, for example, a given role may have five objectives to meet for a particular project phase, this does not say anything about the weight of each objective within those five. Theoretically, it is possible that one objective for sub-phase x and role y far outweighs five objectives for another x and another y, speaking in terms of required effort.

Figure 14. *List of objectives of one cell in the Objectives matrix.*

A project team member who uses Performer can then zoom in further on the objective by clicking on the title of the objective – a list of preconditions, activities, expertise required and deliverables is then presented. However, the attribute of the objective that is regarded by many employees as most valuable is called the set of *Knowledge Elements* belonging to that objective. For each objective in the objectives matrix, templates and best practices of courseware have been generated through the years. By allowing these templates and best practices to be connected to the right objective(s), employees have quick access to the best tools available for their role. In other words, we have the logical data structure shown in Figure 15 (Information Engineering notation), simplified somewhat for the purpose of this thesis.

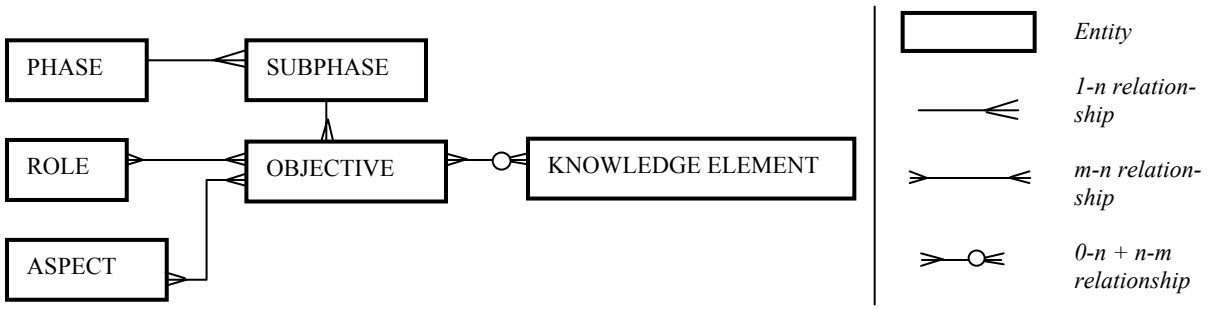


Figure 15. *Relationships between the main entities within Performer.*

The following rules apply to Figure 15, modeled by the nature of the relationships between the entities:

1. A phase is subdivided in one or more sub-phases; conversely, a sub-phase belongs to exactly one phase.

2. A sub-phase has one or more objectives; conversely, an objective belongs to exactly one sub-phase.
3. An objective should be met by one or more roles; conversely, a role has one or more objectives to meet.
4. An objective contains elements of one or more aspects; conversely, an aspect is relevant for one or more objectives.
5. A knowledge element helps in activities that belong to one or more objectives; conversely, an objective contains zero or more knowledge elements (although the zero-case is usually not very useful).

The term 'Knowledge Element' can mean many things, and certainly comprises more than just templates and best practices. Later on in this section, we will elaborate on just what a "knowledge element" is; for now, it suffices to note that a Performer Knowledge Element always pertains to formal knowledge about, in this case educational, projects. For junior or new employees, the objectives matrix provides a good starting point for finding the way within an educational project. For experienced employees, however, it becomes bothersome having to click in the objectives matrix first, then finding the required objective, and then selecting the list of knowledge elements connected to that objective. To facilitate easier access for more experienced users, three ways of approaching the knowledge elements were designed and realized.

First, we can change the matrix view to show not objectives in the cells, but rather the number of knowledge elements for each cell directly. In this case, a project team member sees the number of templates and best practices that are available in a particular sub-phase for a particular role, instead of objectives. This is visualized in Figure 16.

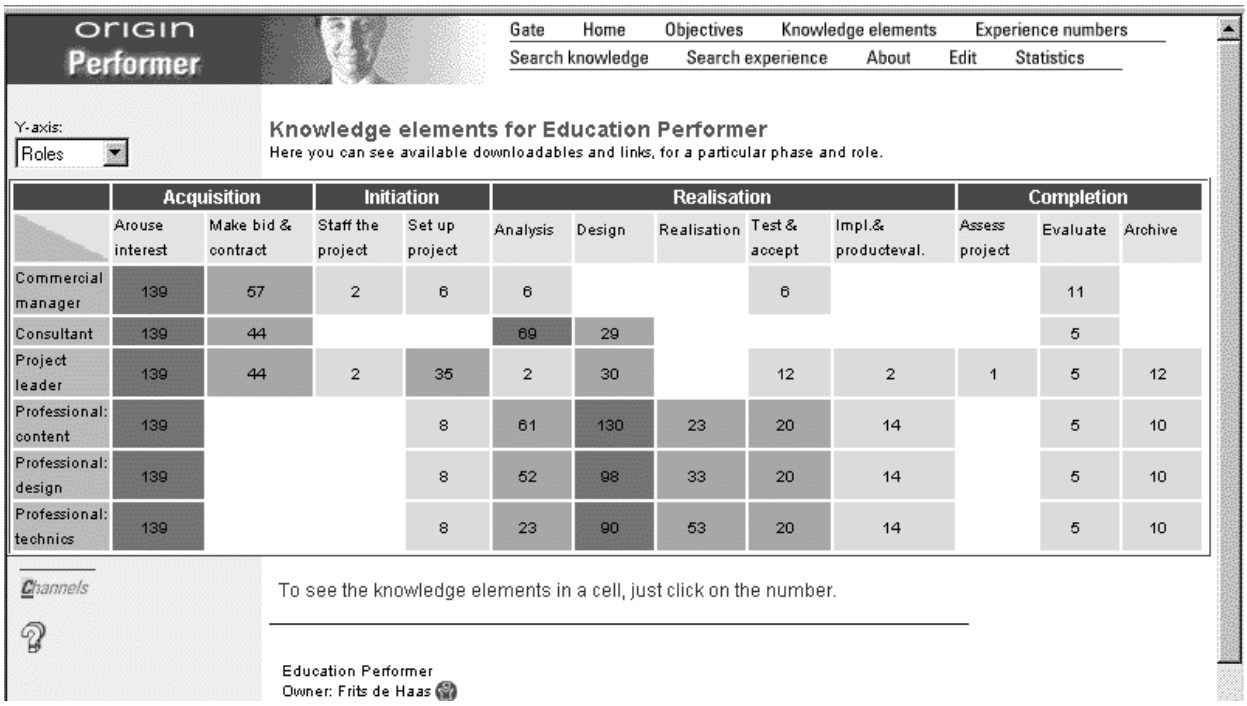


Figure 16. *The Knowledge Elements matrix within Performer.*

Using a slightly different coloring, a deeper shade of purple-pink corresponds directly to a larger number of knowledge elements available in that cell. Also note that a deeper shade in the objectives matrix usually – though not always – corresponds to a deeper shade in the Knowledge Elements matrix. Also note that the six cells containing the number "139" all represent the same knowledge elements (mostly presentations) which are relevant for *all* roles involved.

A second view on the available templates and best practices is the Knowledge Search page within Performer. Accessed through the top menu, the Search page provides the project team member with a list of filters that allow the team member to create a tailored view on all knowledge elements within the Education Performer. These filters define the nature of a knowledge element. The filters, plus an example of a search with its corresponding search result (namely: list all knowledge elements in English for the role "Professional: Content") is shown in Figure 17.

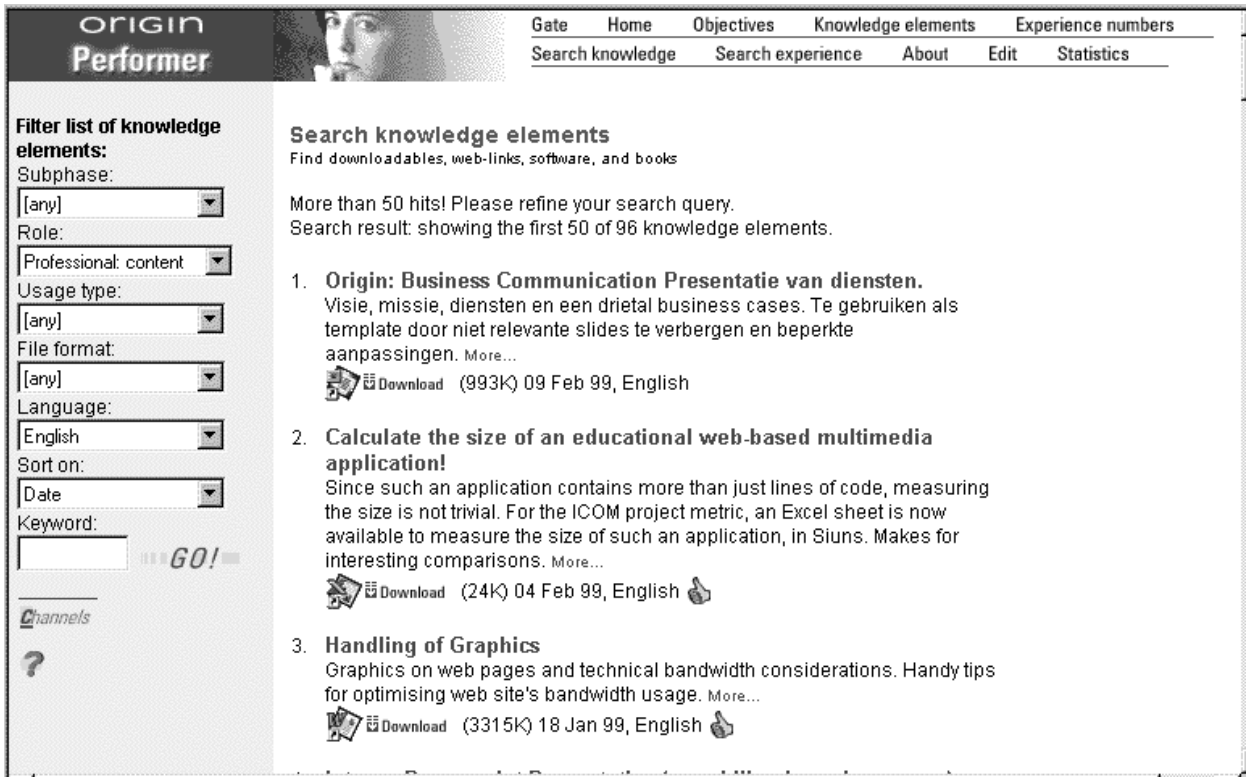


Figure 17. *Search for knowledge elements using the filter list.*

The filters shown in Figure 17 can each be seen as way to describe knowledge element characteristics. We list the six characteristics in Table 12.

Table 12. *Dimensions of Knowledge Elements within Performer.*

Dimension	Values	Brief description
Sub-phase	[All sub-phases as defined in a Performer]	x-Axis of the Objectives matrix
Role	[All roles as defined in a Performer]	y-Axis of the Objectives matrix
Aspect	[All aspects as defined in a Performer]	Alternative y-axis of the objectives matrix
Usage type	Examples, references, templates, bids, literature, course	The nature of use of a knowledge element
File format	Document, presentation, software, spreadsheet, Zipfile, URL, book	Storage format of a knowledge element
Language	Dutch, English, French, German, Spanish, Portuguese, Italian, Japanese, Chinese	Language of the actual downloadable file; description should always be English.

A third quick way of accessing the knowledge elements is in the list of objectives generated by clicking on a cell in the objectives matrix, and then following the path to the list of knowledge elements connected to that objective. This is shown in Figure 13.

Another type of knowledge we wanted to capture in a similar way are the individual estimates. For example, employees at the company of study knew what the approximate cost for one hour of basic browsing CBT on a PC would require. This is a consensus estimate that is especially useful for the consultant or project manager in the earlier phases of the project. In a similar way, other individual estimates for other roles may be identified[13]. In other words, for each combination of project phase and project role, we were able to define a set of questions about consensus estimates, the individual answers to each of these being individual estimates. This led to the creation of a third matrix, in addition to the Objectives matrix and the Knowledge Elements matrix, namely the Individual estimates matrix, which is shown in Figure 18.

[13] Which they were, in a special consensus estimate collection session.

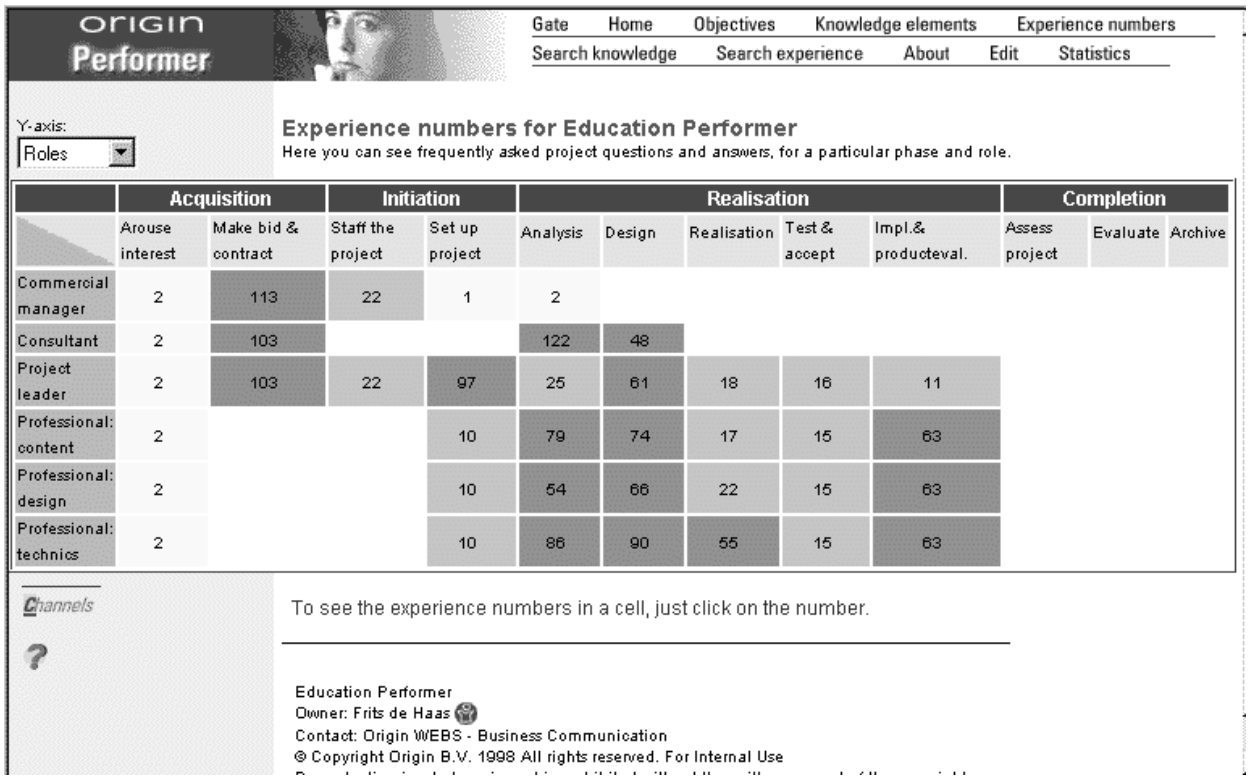


Figure 18. *Experience questions and their answers in the Performer Individual estimates matrix.*

In the Individual estimates matrix, the number in each cell again corresponds to the depth of the coloring. Each number denotes the number of *experience questions* available in that project phase for that project role. All of these questions were formulated in the following way: "How much {time | money | effort} does it take to achieve this or that?" Contextual attributes, preconditions, constraints, and a best-before date accompany each answer to such a question. Individual team members are able to provide an online, freely accessible answer to such a question. If a sufficient number of colleagues submit their own answer to such a question, the overall average becomes an approved consensus estimate. These consensus estimates serve to improve the quality of project bids, and they help project team members estimate the effort required to carry out their tasks.

In Figure 19, one experience question is shown, with some of the available individual estimates listed. The individual estimates include boundary conditions and a best-before date. The individual estimates taken together may form an accepted consensus estimate. An employee examining these individual estimates can then contact one or more of the persons who have submitted their individual estimate for more information.

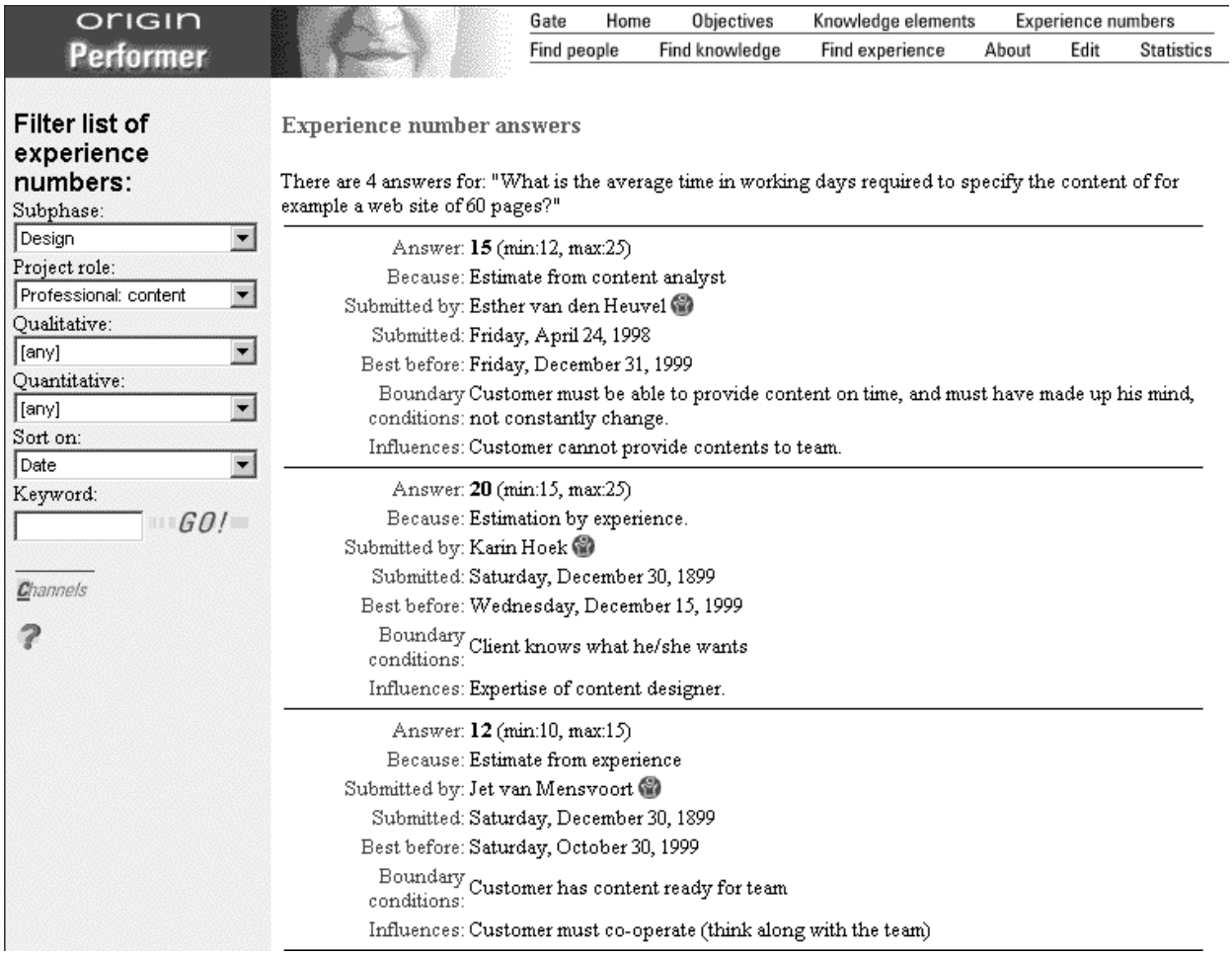


Figure 19. *List of individual estimates provided for one experience question.*

When taking into account the experience questions, individual estimates and consensus estimates, we now have main entities with their corresponding relationship structures within Performer (in Information Engineering notation) as shown in Figure 20.

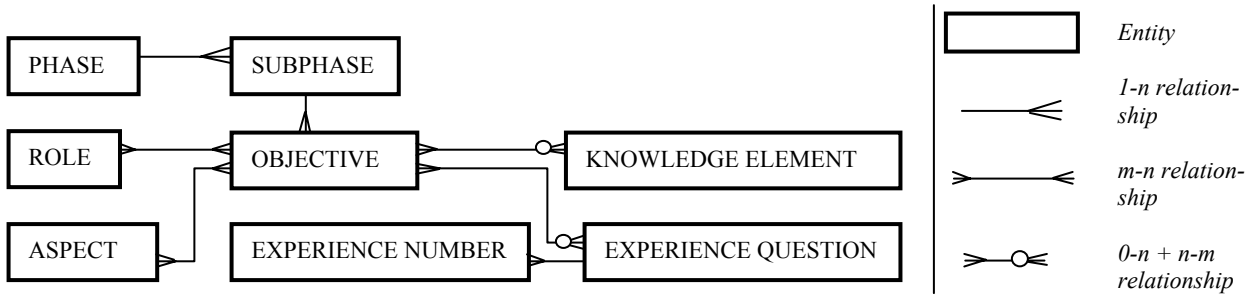


Figure 20. *Extended relationships between the main entities within Performer.*

The following additional rules apply to the logical database structure in Figure 20:

1. An experience question's topic is belongs to one or more objectives; conversely, an objective contains zero or more experience questions.
2. An experience question contains zero or more experience numbers (individual estimates); conversely, an experience number belongs to exactly one experience question.

As with knowledge elements, the experience questions may be queried in multiple ways. First, an employee may locate the desired project phase and project role in the matrix of Figure 13, then browse through the list of available experience questions, and finally select the required answers. Alternatively, an employee may select "Find Experience" from the main menu bar. In a way similar to finding knowledge elements, the employee has various filtering options on all the questions available. Turning on one or more of these filters will produce a list that contains a subset of all available experience questions. These options are also available on the screen depicted in Figure 19.

Finding expertise of employees

A third approach to finding knowledge elements was to capture and represent the available expertise of all multimedia experts involved using the knowledge elements each of them had submitted to Performer. All knowledge elements are tagged with various keywords in addition to their titles and description. This allows us to search on a given topic through the list of all available knowledge elements, and generate a list of employees who have submitted knowledge elements with that topic. We can even count the number of knowledge elements, for each employee, that satisfy these conditions. In this way, we can relatively easily generate a clear overview of the expertise of each employee. We show this in Figure 21.

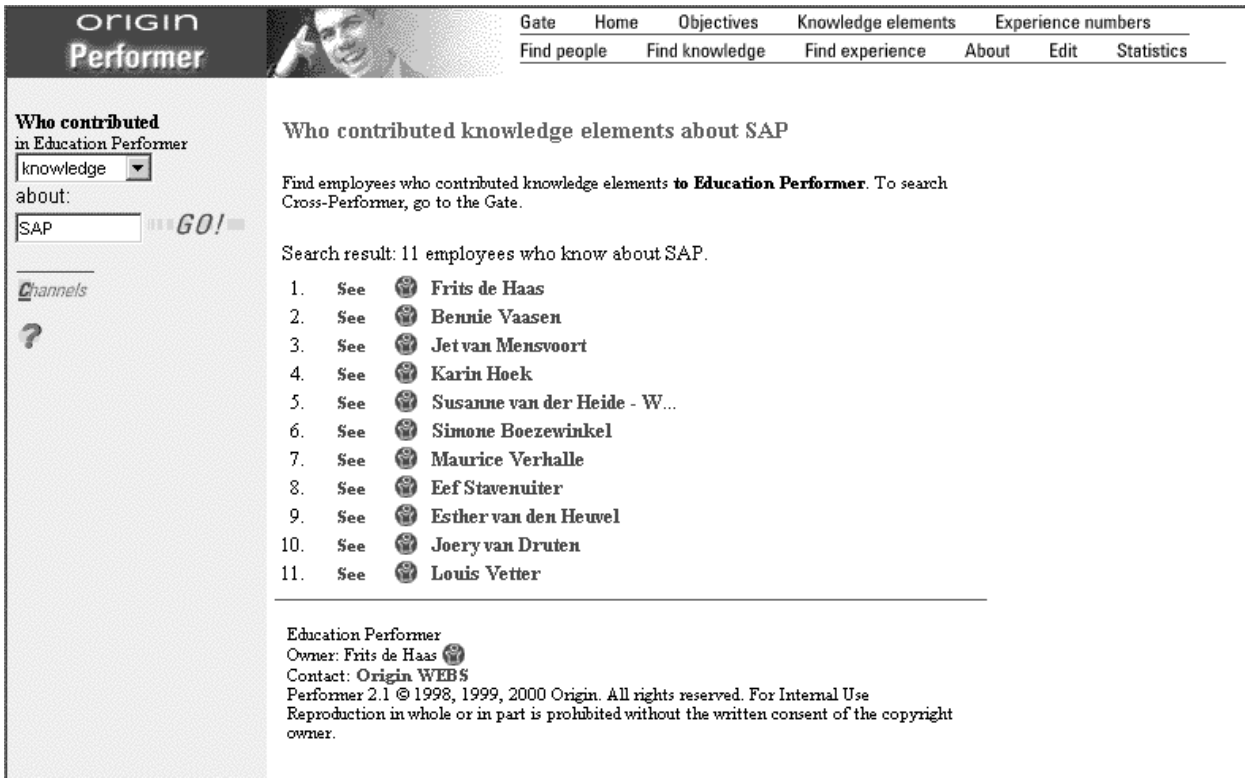


Figure 21. *Finding employees who contributed knowledge about a certain topic.*

Contributing Knowledge

In order for any knowledge repository to be successful, valuable new knowledge needs to be added in an as simple as possible way. If employees must go through bureaucratic, cumbersome procedures or high authorization and security thresholds, the motivation for structurally contributing knowledge will not increase; employees will quickly stop their attempts [DAV1998]. We therefore opted for very easy access to adding a new knowledge element, even if the quality of the

knowledge element is doubtful initially. The procedure for adding a new knowledge element is as follows:

1. Locate the knowledge element to be added on the local hard disk or the local area network.
2. Select the "Add new knowledge element" page from the Performer Edit menu. The screen shown in Figure 22 pops up.
3. Upload the knowledge element from the hard disk using the upload button. The knowledge element is stored at the right place automatically. If the knowledge element is a reference to a book or to a website, this step may be skipped.
4. Enter descriptive information about the knowledge element, such as its title, brief description, descriptive keywords, and, if a reference, where the reference may be found (in case of a reference to an Internet site, this would be a URL).
5. Locate the spot (cell) in the objectives matrix that is best suited for the knowledge element.
6. Select the objective from that cell which best fits the purpose of the new knowledge element. If the knowledge element is useful for more objectives, repeat steps 5 and 6.

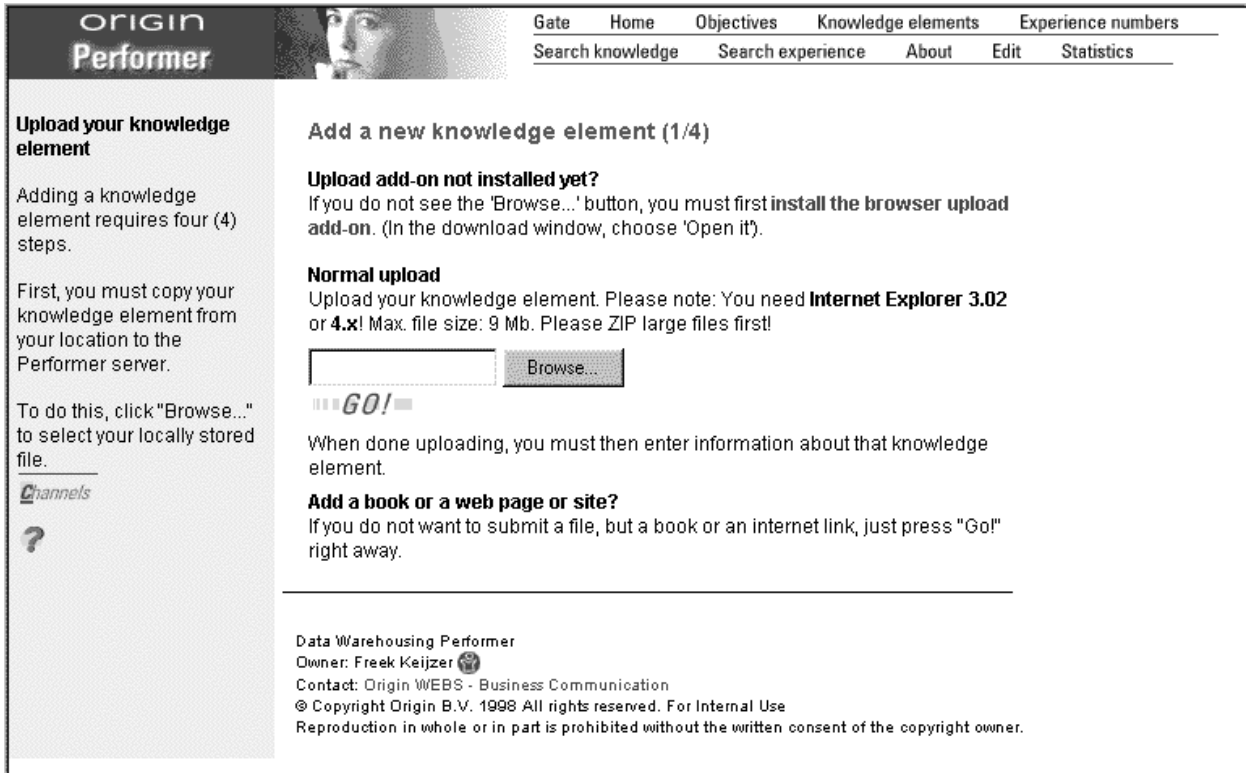


Figure 22. *Uploading a new knowledge element: page 1 of 4.*

As described in the next section, the Performer moderator can then add a "thumb-up" sign to the new knowledge element if it is considered a good asset for Performer; in addition, any other employee can add his or her opinion about that knowledge element. If a sufficiently large number of people add their opinion to knowledge elements, the value of that element will become evident eventually. Some knowledge elements are discarded shortly after they have been submitted, for various reasons; one may be that the knowledge element was already in existence at that time; another may be that the content owner considered the knowledge element to be of a poor quality. In each case, the employee who submitted the knowledge element is informed; formal budget was available for these kinds of activities.

Contributing Expertise
While multimedia experts have valuable templates and best practices to add, there is also another category of knowledge that is just as valuable to store and disseminate, namely the individual estimates. Employees can at any time give their own personal answer to one of the experience questions in the Education Performer. The procedure for adding a new individual estimate is as follows:

1. Locate the cell in the matrix where the experience question is likely to be located.
2. Select the experience question from the list of questions belonging to that cell.
3. Provide the answer: Fill out the form including context data, preconditions, constraints, and a best-before date.

Performer start page
Performer starts with an opening page showing the four most recently added knowledge elements. Some proposed taking the Objectives matrix screen as opening page, which after all shows the entire educational multimedia project process in one view. However, the more experienced employees did not really use the Objectives matrix as much as new employees. Since the real added value of Performer seemed to lie in allowing employees quicker access to useful templates and best practices, we decided to show the most recently added knowledge on the start page – also to stress that Performer is a "living" system with growing content each week. The Start page of the Education Performer is shown in Figure 23.

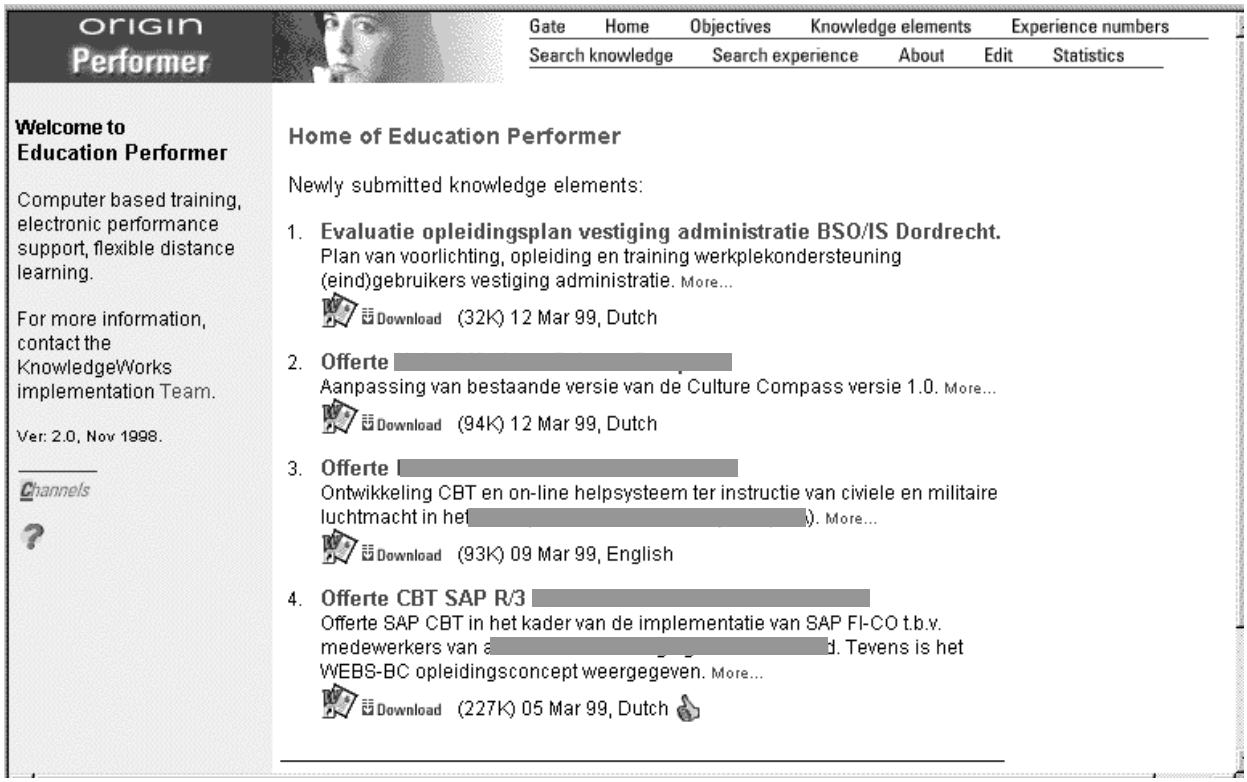


Figure 23. *Start page of the Education Performer.*

Authorization
There were, of course, also some security and authorization issues to face in the functional design. After all, if all useful templates and best practices are stored centrally with free access to all employees, this poses a potential hazard for the safety of all this knowledge, and, consequently, for the security of the intellectual capital of the organization. There is a personnel turnover of more than 10 percent each year; which means, in the case of the company of study, that every year about four to five new employees are welcomed, but about as many employees leave the company, taking their valuable knowledge with them, usually to a competitor, but also perhaps a copy of the entire knowledge database. This is clearly an unwanted situation. On the other hand, Performer was set up with the idea of sharing knowledge between people in the first place. The threshold for contributing knowledge to the system was kept deliberately low: anyone can add suggestions for improved templates and best practices. If the authorization constraints become too tight, employees may forego contributing to Performer. Eventually, we added an authorization mechanism in the sense that everyone has access to browse through Performer and see what objectives and knowledge elements are available, but not everyone has immediate access to the actual downloading of the knowledge elements.

The Performer moderator or other Performer Editors (see the next section for a description of the various organizational roles within Performer) can grant or deny download access to all individual employees. Initially, nobody has download access; ideally, only employees involved in educational multimedia projects who are not leaving the company, have download access. Moreover, each employee with download rights who accesses Performer for the first time must agree to a special *code of conduct*. This Code of Conduct states that the employee shall not use the knowledge and expertise contained within Performer unrightfully, e.g., using knowledge elements of customer X at customer Y, if X and Y are competitors.

5.3. Implementing Performer

While the realization of the functional design of Performer would – theoretically – solve (at least partly) the need for quicker access to knowledge elements and the way of working, getting Performer to work (to be used structurally by people) is quite another matter. Two times before at the company of study, a knowledge repository had been set up technically, and though some enthusiastic team members kept adding knowledge, it was not used much, and both efforts eventually died out[14]. The fact that the functional design of Performer was better than its predecessors is by no means a guarantee for success where the others failed. The KnowledgeWorks model (see section 5.1) has taught us that, in addition to the technical component of knowledge management (which Performer basically is), there are at least three other major components, namely Content, Organization, and Culture. We will now discuss these to illustrate the issues that arose while trying to get Performer to "perform" for all employees.

Content
One of the main causes of failure of any knowledge repository is the lack of available content [WEG2000; DAV1998; HAT1995; VAA1997]. It was clear that to come up with a viable knowledge management solution, there would have to be a certain critical mass of content – templates and best

[14] The reasons for this failure are various, but the most important reason was probably because the effort was not organized: no formal budget was reserved, no roles and responsibilities were defined, and no initial content was gathered.

practices among them – readily available. This may seem obvious, but to actually generate this initial collection of content is not easy in practice. It implies a lot of effort to invest in a solution that is not guaranteed to work at all. Much of this effort consists of extracting knowledge from people's cabinets, desks, and heads. Subsequently, all elements in this inventory must be judged on quality and usefulness. One of the many difficult questions here is who does the judging of the quality, initially and later on? This is a problem that is found in library sciences in general [RUG1995]. This leads to the other major components that must be in place, namely Organization and Culture. First of all *organization*, because the availability and quality of content must be managed by various roles with tasks and responsibilities. Culture, because employees must be willing to share their own knowledge and trust the quality of the available knowledge.

One senior employee was assigned the task of making an inventory of all available and useful templates and best practices so far. This employee was then formally assigned the role of *content owner* of the Education Performer. The initial inventory would serve as the solid content basis of Performer. Employees who would first work with Performer would be able to find most things they were looking for even the first time. This senior employee spent fifty percent of his time, for two months, interviewing employees, copying files and gathering piles of paper. Another month was spent on filtering the results of the inventory: what elements were still useful, which ones were redundant, and which ones would be discarded? To avoid endless discussions with employees about the quality of one element relative to another, this filtering process was assigned to the senior employee only. All other employees are able to submit their opinion about a knowledge element later, using Performer functionality. If a sufficient number of employees give their opinions about a knowledge element, the usefulness of that element becomes evident eventually.

Many knowledge elements were discarded, some were combined, and others were checked with their initial creators (if still employed at the company of study). Consequently, three months after the initial gathering of content was started, we ended up with about 170 megabytes of knowledge elements, dozens of links to paper reference works, and an equally large number of hyperlinks to useful sites on the World Wide Web.
The knowledge elements were then assigned to one or more objectives. Since the matrix of objectives had been defined before, almost on a one-to-one basis with the Promise handbook, there was a generally accepted umbrella to assign the knowledge elements to. Each objective was assigned to precisely one project sub-phase. Furthermore, each objective was assigned to at least one project role and at least one project aspect. Consequently, from the moment that a knowledge element is connected to an objective, it is automatically assigned to the project sub-phases, role, and aspects of that objective.

The gathering of the content for the individual estimates was done differently. We first asked all educational multimedia experts about experience questions that either they would like to see answered, or they knew the answer to. These questions were then categorized and put in the objectives matrix. We eventually ended up with 146 experience questions, which were all connected to their most suitable objectives. We then organized a joint session, where all employees were gathered. We formed four groups, three according to the Performer roles [Professional: content; Professional: design; Professional: technics], and a fourth group to represent the other roles in Performer. This was done because the total number of people for the other roles would not justify forming as many new groups. These groups then used Performer to give their own answers to the

experience questions that were relevant to them. We eventually ended up with 337 answers for the 146 experience questions. This seemed a solid enough basis for the initial required content.

Organization
Getting the knowledge management process organized was one of the most difficult challenges in making Performer a success. It does not suffice to merely get the technical solution online and have people add and use knowledge without some form of moderation. We eventually identified the following roles:

1. **Educational Multimedia expert**. Most employees are in this role, and use Performer for daily or weekly project tasks.
2. **Performer Editor**. The employee in this role has additional editing rights, for example granting download access to an employee or adding a "thumb-up" sign to a new knowledge element.
3. **Performer Champion**. The employee in this role must make sure that people remain motivated to use Performer.
4. **Performer Content Owner**. This employee is responsible for the quality of the knowledge and expertise involved within Performer. This also includes the description of the objectives.
5. **Performer moderator**. Actually the "boss" of the Performer, this employee can do all administrative tasks within Performer, such as re-arranging knowledge elements in the objectives matrix, granting or denying people download access, adding new roles/aspects/objectives, etc.
6. **Performer Change Control Board**. This is actually a group of people that guard the functionality and version control of the current Performer. As Performer is used, additional functionality is desired; the Change Control Boards decides on the acceptance of these wishes, depending on whether or not they conflict with the concept of Performer. You can read more about this in the subsection "Functional maintenance", described below.

The *Educational Multimedia expert* contributes knowledge and expertise to Performer, and uses knowledge and expertise from Performer for project tasks. It depends on the level of seniority of the employee which of the two is done more: a junior employee will use more than contribute, while a senior employee will contribute more than use, on average. At the beginning of each calendar year, agreements are made on the degree of use of, or contribution to, Performer, in a special agreement called the IOP (Individual Development Plan, Dutch: *Individueel Ontwikkel Plan*). At the end of the same calendar year, the agreements are evaluated. The amount of contribution by any employee can readily be measured through the statistics pages of Performer. If contribution wasn't as expected, the reasons for this are discussed.

The *Performer Editor* plays an active role in moderating the quality of the content of the Performer and the authorization maintenance. Any Performer user can be assigned the role of Performer editor, although in practice this will usually only be employees who have used Performer for a prolonged period of time. An editor has the right to formally approve a new knowledge element (place a "thumb-up sign"), and may grant or deny download access to Performer. An Editor can also update the news content on the opening page of Performer.

The *Performer Champion* is more geared towards motivation and enthusiasm, rather than on technical or content issues. An employee in this role typically presents the concepts and advantages to various audiences, organizes workshops, answers frequently asked questions, motivates employees to become – or stay – involved, and generally tries to keep people enthusiastic and motivated.

The *Performer Content Owner* guards the integrity and consistency of the objectives matrix and the knowledge elements that are connected to the objectives in the matrix, as well as the quality of the experience questions and corresponding individual estimates. Employees who add a new knowledge element, for example, often do not immediately have a clear picture of which objectives are most suited to the knowledge element that is submitted. The content owner periodically checks newly contributed knowledge elements and corrects the links to the objectives.

The *Performer Moderator* has the overall responsibility for the integrity, accessibility, and selling value of a particular Performer. He or she is responsible for regular news updates, and for notifying Performer users about scheduled server maintenance or other issues that arise at any given point. It is also usually the Moderator who is member of the Origin Performer User Group (OPUG), who meet on a regular basis to discuss new Performer functionality (See below). In most cases of existing Performers, the roles of Content Owner and Moderator are combined into one employee.

The *Performer Change Control Board* (PCCB) is a group of three employees who guard the functional concept of Performer. As the number of Performers increases[15], so do the wishes for extra functionality and features. Since not everyone has a thorough understanding of the functional concept of Performer, some of these wishes call for quite another solution, and would harm or even destroy the functional integrity and concept of Performer. Therefore, the Change Control Board decides on the approval of these wishes. This is organized in cycles of four months. During these four months, the Performer User Group gathers the wishes for extra functionality and features. This group holds a representative from each Performer, who also holds part of the budget for the improvements. Together, the Performer User Group holds a certain budget each year for improvements. Those that are approved by the Performer Change Control Board are then implemented every four months. Thus, the process of functional maintenance is as shown in Figure 24:

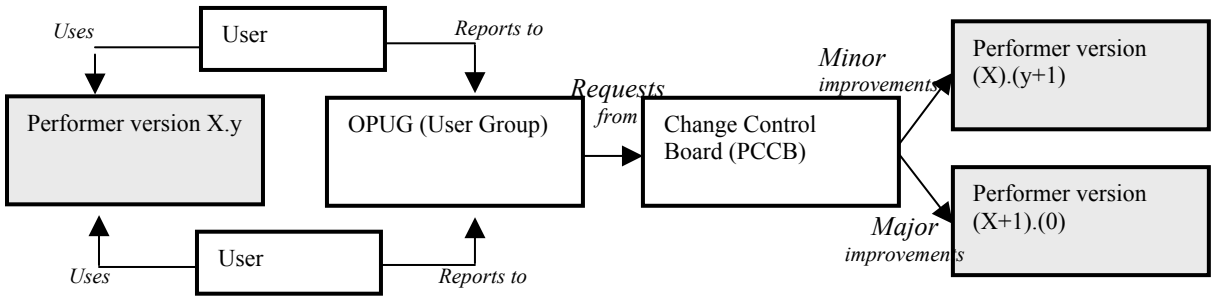


Figure 24. *Functional maintenance of the formal knowledge management solution Performer.*

[15] At the time of writing of this thesis, there were eight Performers in Production (live) status on the Origin Intranet: Education, Knowledge Management, Data Warehousing, Migration Services, SAP Implementation Services, SAP decision support, Interoperability, and others. More were scheduled to be implemented. All Performers use the same programming code and use the same physical SQL database.

Culture

The largest part of the budget for implementing Performer is spent on cultural aspects (some forty percent as a rule of thumb). After all, why would an employee be willing to put time in giving away his or her own knowledge, if there is no guarantee that the possible advantages will actually work? This is especially true for employees who have experienced previous failures of similar initiatives. Therefore, employees must be convinced that this effort is on a sufficiently professional level, and, more important, that it is to their own advantage to participate in building this solution. The best results are achieved if employees become enthusiastic about the idea and sell the working solution with pride, which happens if the solution becomes a success story. In line with [ALL1997; DAV1998; LEO1995; WEG1997], we have taken the following steps to achieve this enthusiasm:

Presentations

The company of study held a technical group meeting each month. On such meetings, where all multimedia experts are present, two or three employees inform others about the projects that they are currently involved in. Several of these meetings were employed to inform employees about the design of the knowledge management solution called Performer. The disadvantages of the use of the existing methodology Promise were highlighted, and the advantages of using Performer to work in very much the same way, were emphasized. Also, the main reasons for the failure of previous similar initiatives were discussed, and ways to avoid making the same mistakes (for example, involving all employees in generating the required critical knowledge mass at startup).

Performer champion

The Education Performer Champion is the person who initially held interviews and browsed through employees' desks and cabinets to find useful knowledge. This process turned out to be a motivator in itself, because employees felt that their knowledge was in demand. It also made them curious to find out just what it was that was being designed and how much it would help them in finding the knowledge that they required at any given point in an educational project. This curiosity was gratefully used in the workshops described below, which in turn generated a certain level of motivation or even enthusiasm with employees.

Workshops

At an early stage, we wanted to organize hands-on training sessions about using Performer, in an effort to keep the momentum of enthusiasm that we had created with the presentations and gathering of content. So, even though the technical code still wasn't complete, we organized Performer usage sessions in which employees could experience in a direct way just how quickly certain types of knowledge could be traced, without having to go through a whole string of telephone calls or emails.

In addition, we hired a special motivation-training expert, who brought in a large set of percussion instruments. The idea was to show to employees that working together (and sharing each other's knowledge) is much more productive than doing everything all alone. Most multimedia experts were a little skeptic about this idea at first, but once the percussion session got underway, more and more of them understood the message that the trainer was trying to convey. In retrospect, the motivation for sharing the power of their knowledge had indeed increased significantly.

Help files

We decided to design and develop context-sensitive help for each Performer screen that could result in difficulties for users. The idea was to add a small question mark sign on these screens. Clicking the question mark would bring up a small popup window containing a Windows-like help file layout. The text in this popup window would take into the account the context of that particular page. To leave the context-sensitive help and return to Performer, the user would simply close the small help window. In the end, some fifty-five separate help screens were designed and developed by a graduate student in communication sciences. In contrast with the functionality of Performer, whose browser pages are generated through scripts extracting data from a SQL server database, the help files are static html pages that need adapting only if new functionality as approved of by the Performer Change Control Board is implemented.

Statistics

Also in the context of keeping employees motivated to use Performer, we added a Statistics section in the main menu. The first of these is called the Education Performer Hall of Fame. Each time an employee adds a new knowledge element, the Knowledge Elements counter of that employee is increased by one. Eventually, a list of employees is generated; it shows the number of knowledge elements each of these employees has contributed. The list is sorted on number of knowledge elements. Looking at Figure 25, it will be clear that employees remain motivated to move up on the ranks by adding new knowledge. Of course, if the Performer Content Owner does not approve this new knowledge, it may be deleted.

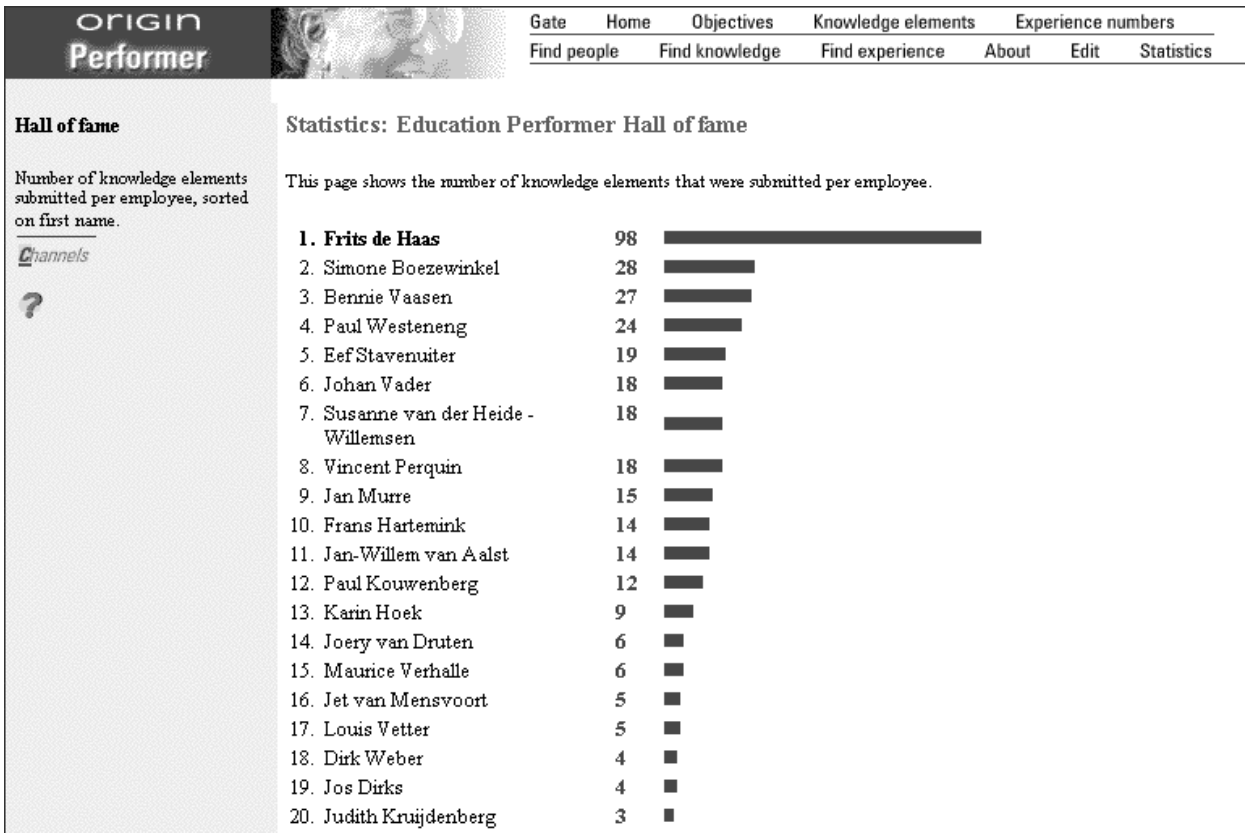


Figure 25. *Performer Hall of fame with numbered list of people.*

Other statistical tools include access counters for all objectives (and subsequently a list of which objectives are most frequently consulted), access counters for all knowledge elements available (and correspondingly a list of the most popular knowledge elements available within Performer), and statistics on the number of performer users (both users with and without download access). However, these statistics in itself are only a small subset of the data required to determine if Performer really is a success. This issue will be discussed more in-depth, later, starting in section 5.5.

Administrative tools
Performer was also enriched with some administrative tools. These are:

1. **Submit a change request**. We wanted to facilitate an online possibility to suggest improvements to Performer. Every employee can add valuable suggestions for improvements in functionality (or report bugs). Every four months, the Performer user group evaluates this list.
2. **Edit or delete knowledge elements**. For Performer editors, content owners and the moderator only, this tool facilitates guarding the quality of the knowledge elements.
3. **Control the content of the matrix** (add/delete phases, roles, aspects, and objectives). In order to avoid having to work in the database directly, administrative screens were developed to facilitate the structuring of the matrix. If a new role is to be added, or objectives to be edited or moved within the matrix, this can be done using these screens. Of course, only the Performer moderator, content owner and Editors have the permissions to do this.
4. **Grant/deny download access** and Grant/deny Editor access to a Performer. For the content owner and moderator only, access to Performer for any user can be administered using this tool. User accounts are administered on the central intranet of the company of study, and are re-used in Performer (not duplicated).
5. **Administer a newsflash**. Important upcoming events or bug fixes can be announced through the news flash, which is shown on the home page of the Education Performer.

Technical infrastructure
Within Atos Origin, the company of study, an intranet had already been set up using Microsoft technology. All employees had free access to this intranet, and so we decided to set up Performer as a sub-component of the intranet. The program code was developed in Visual Basic Script using Active Server Pages technology; all edited using the Microsoft Interdev environment. The database was initially set up in Microsoft Access, but was later ported to SQL server because of the more powerful capabilities of the latter database environment. Cascading Style Sheets were used to ensure a consistent user interface throughout the website. In appendix C, we give an overview of the site-map of Performer.

Authorization was done using existing Windows NT accounts on the corporate intranet of the company of study.
A standard client browser was defined within Atos Origin, both the html code and the cascading style sheet could focus on that browser, instead of having to comply with a variety of browsers. All the points mentioned above ensured that the technical infrastructure aspect of Performer was the least expensive of the four KnowledgeWorks aspects.

5.4. The ICOM metric

Performer is a knowledge management solution geared towards day-to-day, operational use by all educational multimedia experts involved in the projects of the company of study. As such, it helps the company of study achieve level two of the Capability Maturity Model. On that level, the reasons for a successful (courseware) project can be identified and usually repeated. In Table 26 we list the actions required to reach level 2. On the third level of the CMM, measurement programs are also in place, in order to assess the quality of the process of the team itself and to initiate improvements. These measurements are also required to determine if the use of Performer actually results in more successful projects, and if yes, to what degree. In other words, the measurements are employed to determine the usefulness of Performer in a statistically valid way. On a more abstract level, we also need the measurement program to be able to test the research hypothesis given in chapter 3.

Performer itself is not a measurement tool. And even if it were, most multimedia experts do not have the time or budget available, and do not feel the direct need, to record measurements about their own project process[16]. Therefore, the metric to record measurements about Performer was set up as a separate tool, though using the same user interface concepts and technical infrastructure as Performer. Before describing the functional design of the metric, we once again stress the two different target groups for Performer and the ICOM metric:

- Multimedia experts use performer for their daily or weekly project tasks. The researcher does not use Performer for measurement activities, but does monitor its use by employees.
- The ICOM metric is used by the researcher(s) to monitor the quality of the process of projects that use Performer. Multimedia experts do not the use the metric for their daily tasks themselves.

Since this research project aims at improving the maturity of the project process of the educational multimedia experts involved, we feel that a metric should involve both quantitative (measurable in numbers) and qualitative (measurable in content) variables, because maturity has both a quantitative side as well as a qualitative side to it. An important requirement is that, for both the quantitative as well as the qualitative dimension, we measure factors that determine the maturity of the project process. An informal inventory of literature on process maturity [CON1986; GAR1996; GRE1996; JON1997; MUS1987; PUT1996; SOF1995; YEH1993] results in the following list of measurement units:

1. Household data such as the project name, customer name, product name, product type, potential risk, and available budget (this is for identification purposes rather than for measurement purposes);
2. Estimates on all quantitative variables involved, such as: calendar time, cost, effort (man hours), product size, function points, project risks, and defects found;
3. Actual values on the same quantitative variables, in order to be able to calculate the difference between estimations and actual values, and thus determine to some degree the quality of the estimates;

[16] This may say something about the level of professionalism of the company of study, but equally so about the level of professionalism of customers, since many of these are not willing to reserve budget for such activities.

4. Definition of available skills and structure of the project team; this is required to measure the amount of qualification of required skills or disciplines, in other words, is the project equipped with the (human) resources that are required for the project;
5. Project experiences about qualitative variables: project management, consultancy, (interdisciplinary) communication, technology, and political, psychological, and customer relationship issues.

Before explaining how to incorporate these measurement units into a metric, we now briefly explain each of the units.

Explanation of the metric measurement units

Ad. 1. *Basic project "household" data*

This project data is required to determine if the project meets the requirements and boundary conditions for analysis in the scope of this research project. This is because for very small or extremely large projects, different rules probably apply to the required level of maturity. Especially the budget part is an indicator for the expected complexity and size of the project, but the potential risk and description of product type also influence the suitability of the educational project for research analysis. Potential risk includes difficult-to-control factors such as hidden political agendas, customer unfamiliarity with IT, and software requirements that are hard to clarify. The potential risk can also be made quantitative in the shape of weighted factors; this is why risk is also mentioned on (ad 2). The product type should, of course, be in the field of educational multimedia applications. This measurement unit is actually more for identification purposes rather than measurement purposes.

Ad. 2. *Quantitative project variables*

The quantitative project variables are gathered mostly for prediction purposes. We have listed the following examples of quantitative project variables: calendar time, cost, effort (man hours), product size, function points, project risks, and defects. We eventually decided to measure a subset of these, according to the minimum set of required quantitative variables as reported by Putnam & Myers, and Greene [PUT1996; GRE1996]. This resulted in a quantitative measurement set of the following variables: calendar time, budget, effort, product size, and risk factor. Of all these variables, we gathered the initial estimated values (before the project had begun), and collected the values of the same variables in retrospect (i.e., when the project had finished). For each variable, we recorded multiple values (e.g., for the Time variable, we recorded estimates and actual values of several milestones, not just one). When comparing the actual values of all variables to their estimate counterparts, we can say something about the discrepancy between these two. We call this difference the *discrepancy factor* of the project. The discrepancy factor one part of the main experiment, described in section 6.1.

We list the five quantitative project variables used in our research in Table 13.

Table 13. *Quantitative project variables used in the scope of our research.*

Variable	Explanation
Time	Calendar time, usually in days. Project milestones are examples of time variables.
Cost	Project budget, consisting of money spent and money received from customer.
Effort	Working hours spent on the project
Size	An indication of the size of the educational multimedia application. This is the hardest of them all. Eventually we designed a formula to address this, which is given in appendix E.
Risk	A "risk" number that is generated from a questionnaire about possible project risks.

Quantitative Software Management ltd. (QSM) uses the same inputs, with their project estimation tool called SLIM[17]. QSM holds that, if any four out of five project estimates and actual values are given at any point in time for that project, SLIM can make reliable predictions about whether or not the project will meet the requirements in time and budget. This is published in [PUT1996, GRE1996]. In other words, take the estimates for size, effort, risk, and time, record the actual values of these variables and input them into SLIM, and a prediction for the flow of budget for the rest of the project can be generated. After that, the variables can be slightly adjusted to see what will happen in other cases. For example, we could say we will put more effort into production, which costs more money but can also result in shorter throughput times.

The most important variable for SLIM to achieve this, is the size of the application to be built. Of course, with educational multimedia, it is not easy to define one unit of measure for the size of the application. This is because an educational multimedia application includes much more than just lines of code, or function points, or GUI screens, which are widely used measurements for more traditional software applications. In the research world of software engineering, much work has been done to find unambiguous objective formulas for determining the size of a software application [PUT1996; JON1997]. Software process improvement tools have gratefully employed the results of that research to develop tools that can be used to improve the maturity of the software project process. However, these tools have up to now always failed to take into account the more complex highly interactive systems such as multimedia products (VAU1994, AFT1988; CON1986), although some do pay attention to the multidisciplinary character of measuring a multimedia product [MYE1994; CAR1990].

When first attempting to determine the size of a software application, researchers started to count non-breaking lines of code. This could not include comment statements or empty lines. Many complex rules have been devised to make this counting an objective and unambiguous task. However, it was not long after this method became commonplace, when people realized that the size of a software application could not be determined by just counting lines of code. Hence, more sophisticated methods were devised.

[17] Actually, SLIM uses defects instead of "risk". However, as we explain in section 6.2, we had no way of objectively determining a "defect" in educational multimedia applications. The methodology used, called PROMISE, did have a risk analysis for educational projects. Since defects are usually the outcome of risks, we substituted "risk" for defects, which was not a problem because the educational multimedia projects were not fit for SLIM anyway. For more information about QSM, the reader is referred to www.qsm.com.

The most widespread of these is probably Function Point Analysis (FPA) [SOF1995]. Though this method seems to work in a satisfactory way for many traditional software projects, the field of multimedia needed something in addition, because the effort that is put into the production of a multimedia product is far more complex than software engineering issues alone [ENG1996, VAA1998B]. More often than not, the heaviest components of effort are in visual and graphic design, video and audio production, or functional requirements analysis [VDM1995]. Therefore, we cannot suffice by just counting lines of code or function points. In many modern multimedia production tools, lines of code are completely absent. Function points are no good indicator either, because (for example) the megabytes of graphics that are used in multimedia product x, may well have taken about 50% of all the effort put into the project [VAA1998A].

This is why we came up with the notion of the SIUN, which is an abbreviation of Size Unit. A SIUN is a weighted average of the most important components of an educational multimedia product. Similar to other quantitative project variables, the size of an educational multimedia product in SIUNS can be estimated beforehand, as well as determined in retrospect. This has been published in [VAA2000A]. The exact definition of the SIUN is given in appendix E. We list the most important factors in Table 14.

Table 14. *The elements that make up the Multimedia size unit (SIUN)*

Multimedia product aspect	Weight	Unit	Complexity
Lines of html code (# characters div. by 40)	0.10	Count(Char)	{1,2}
Lines of dynamic html code (# characters div. by 40)	1.20	Count(Char)	{1,2}
Lines of script code	1.80	Count(Char)	{1,2}
Uncompiled lines of 3rd gen. language code	3.20	Count(Char)	{1,2}
Uncompiled lines of 2nd gen. language code	3.50	Count(Char)	{1,2}
Graphics sources	0.30	Kilobytes	{1,2}
Animated graphics sources	0.40	Kilobytes	{1,2}
Realtime audio	0.40	Kilobytes	{1,2}
(Streaming) Video	0.45	Kilobytes	{1,2}
Database structure: # tables + # fields + #rel.	3.00	Count()	{1,2}

The weight of each multimedia aspect not only depends on the amount of effort that is required to produce it, but also on the unit that is chosen for that aspect. For example, graphics sources may well take up several thousands of kilobytes, and if not properly weighted, their influence would incorrectly overwhelm the effort for the lines of code (if any). The unit column denotes what the result is of what is counted.

The complexity column is a separate matter. The fact that, for example, the graphics factor is not large does not mean that it did not take a large amount of effort to create these graphics. Therefore, we can multiply the value by a complexity factor that ranges from 1 (not complex) to 2 (very complex), with digits in-between possible (for example, 1.65). Please note that we can also include web-based multimedia products since html and other scripting languages may also be included.

Ad 3. *The actual values of the quantitative variables for the discrepancy factor*
The main reason why we have set up the metric is to be able to determine if the knowledge management solution employed helps in achieving a higher level of project maturity. In Ad 2 above, we have seen how we record estimates for the five quantitative project variables, plus the actual

values in retrospect, i.e. when the project is finished. We can now calculate the difference between these two points of measurements, for each quantitative variable. This results in the *discrepancy factor* for each of the quantitative variables. The discrepancy factor is a number in the unit of that particular variable unit, e.g., for time, the unit will be days; for cost, the unit is currency; for size, the unit is SIUNs, for effort, the unit is man hours, and for risk, the unit is an integer number. In Table 15 we provide an example of calculating the discrepancy factor for a typical project:

Table 15. *The discrepancy factor of the five quantitative variables*

Variable	Estimated	Actual	Discrepancy
Time	23 weeks	24 weeks	1 week
Cost	$42,000	$40,650	$1,350
Effort	1100 man hours	1014 man hours	86 man hours
Size	1834 SIUNs	2044 SIUNs	210 SIUNs
Risk	235 out of 1188	244 out of 1188	9 out of 1188

We can now keep track of the discrepancy factors of the projects in which we use the proposed knowledge management solution, and compare them to the discrepancy factors that did not use that solution. The hypothesis, then, is that the discrepancy factors for the projects were Performer was used become smaller, implying that estimates are better when using the knowledge management solution proposed. Additionally, we can use these numbers as input for QSM's tool SLIM, once generic units such as the SIUN are supported.

The quantitative data (both the estimates and the actual values of the quantitative variables for all projects analyzed) was arranged in a special "Metrics" sub-screen of the Project details screen; this is shown in Figure 26 (As can readily be identified, the actual values for this project were not yet known at the time of the screen grab). The discrepancy factor for each of the quantitative variables can be calculated automatically. In Figure 26, customer data is grayed out.

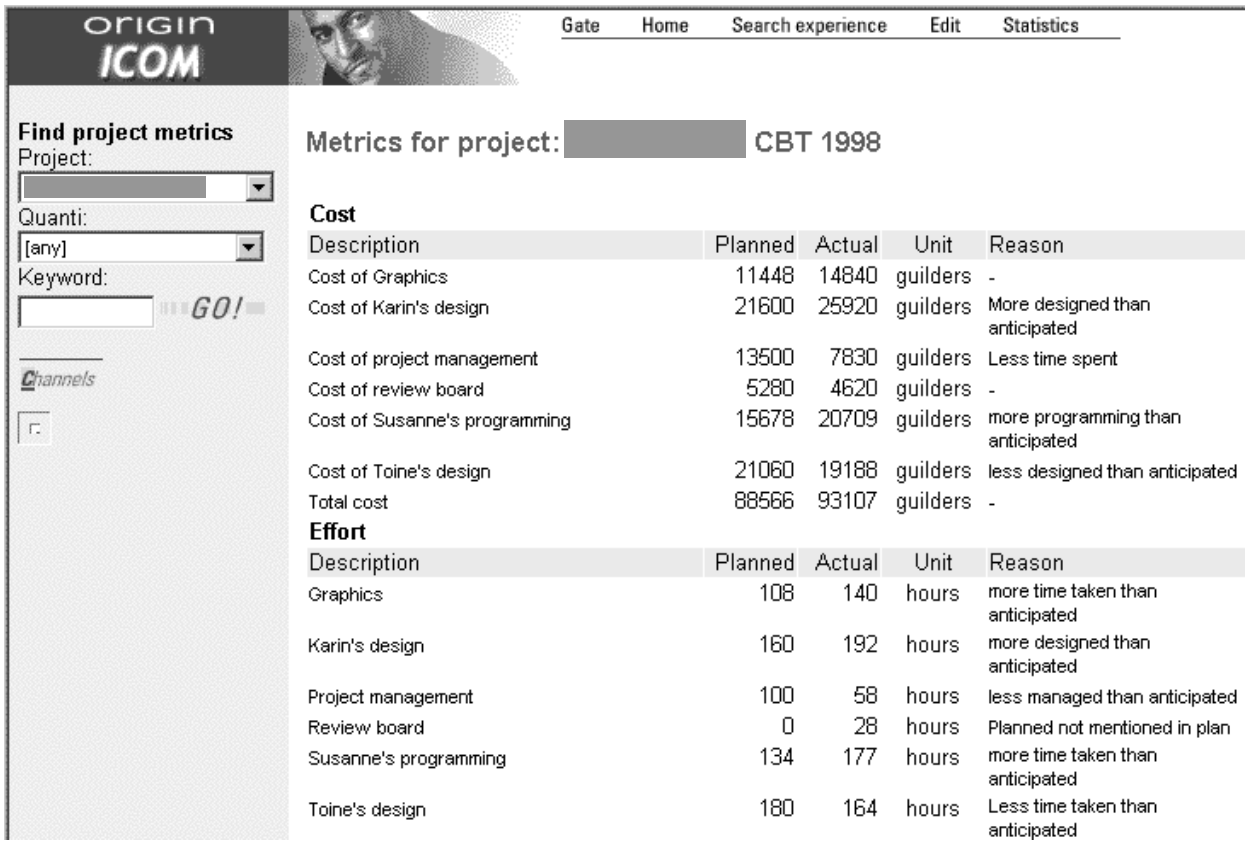


Figure 26. *Metrics data (quantitative variables) on an Educational project.*

Ad 4. *Definition of available skills within the project team*
For each educational multimedia project, we record the names of the project team members, and for each member, the role(s) that he or she fulfilled during that project. Sometimes one team member fulfilled more than one role, and conversely one role had to be covered by more than one team member. With this list available, we can later determine if any disciplines required for the educational multimedia project were missing, in other words, if the qualification of skills is sufficient. If it is not, this might point to one of the causes of process problems registered later. An example of a recorded list of available skills is given in Figure 27. This figure lists some of the basic household project data that is captured, in addition to the team structure. Please note that the ICOM metric must not be confused with a project management tool, as planning schemes such as Gantt-charts and prediction models are not supported.

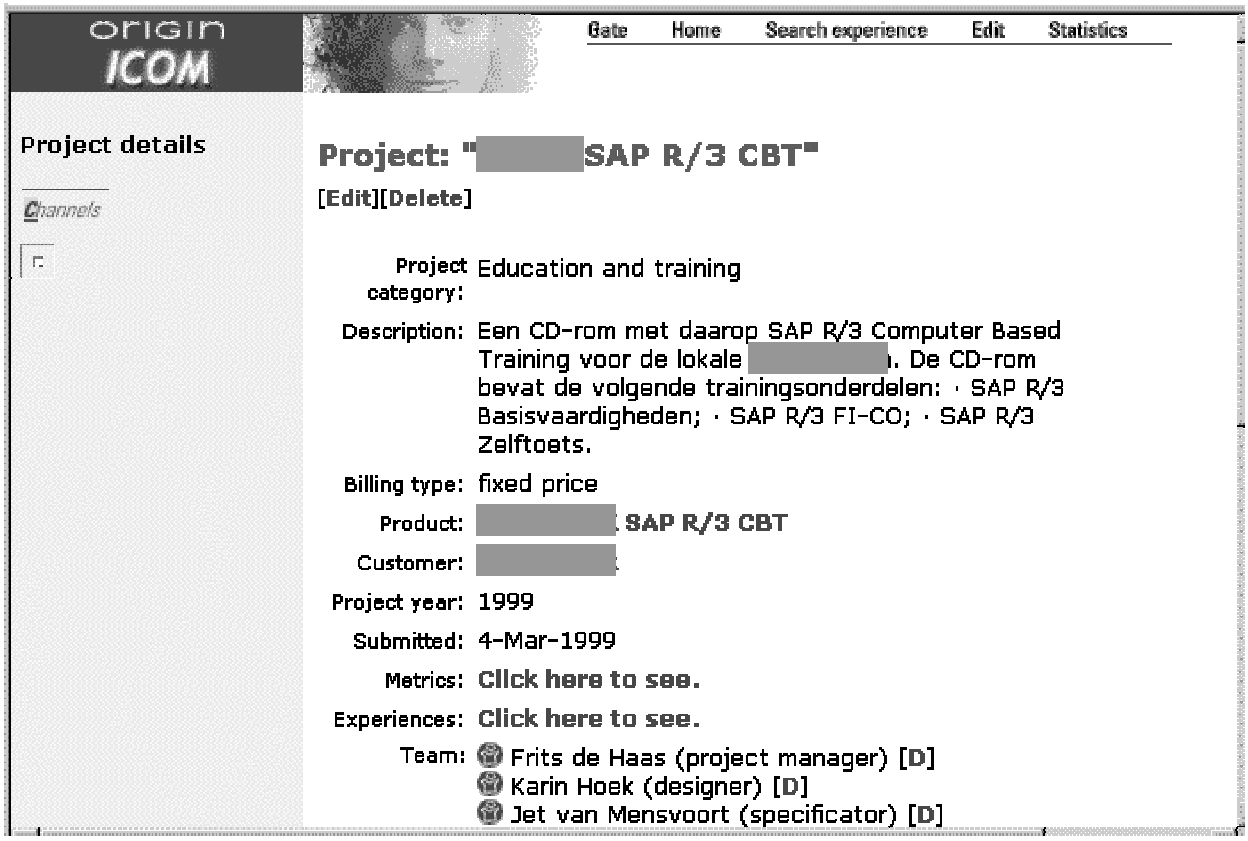


Figure 27. *Overview of a project within the ICOM metric and experience database (in Dutch)*

Ad 5. *Gathering project experiences*
With dimensions 1 and 4 combined, we were able to determine if an Educational Project met the requirements and constraints for analysis (e.g., within a certain budget range, with a certain required amount of project team members, etc.). In addition to the quantitative variables described above, we also felt the need to capture the more qualitative variables, which are harder to measure in numbers. These include aspects such as quality of interdisciplinary communication and psychological issues. Although these seem hard to make quantitative at first sight, research reports offer valuable methods for assigning numbers to these variables [BAA1995; RIF1998; OPP1992].

The qualitative variables were gathered through structured project interviews – using questionnaires – with the educational multimedia experts at the company of study. Sometimes these interviews were held on an individual basis, while at other times the project interview was organized as a group session, depending on the amount of time each employee had available for the project interview. The questionnaires that were used to gather this interview data are described in chapter 6, section 1; a more comprehensive listing of the questionnaires may be found in appendix A.

We used the identified list of project problems as variables for the qualitative dimension. Subsequently, we were able to categorize all interview entries according to these problem categories. At an early stage, employees themselves did this categorization process on an intuitive basis; later, three independent scientific researchers did this categorization process (this is described in chapter 6). An example of this early categorization is given in Figure 28. The advantage of modeling the categorization this way is that project experiences of a certain category are easier to trace. For example, in Figure 28, it is easy to filter all project experiences about the quality of communication,

for one particular project, or for all projects available. Also, it is now possible to query a list of all project experiences that contain a certain keyword, such as "stress".

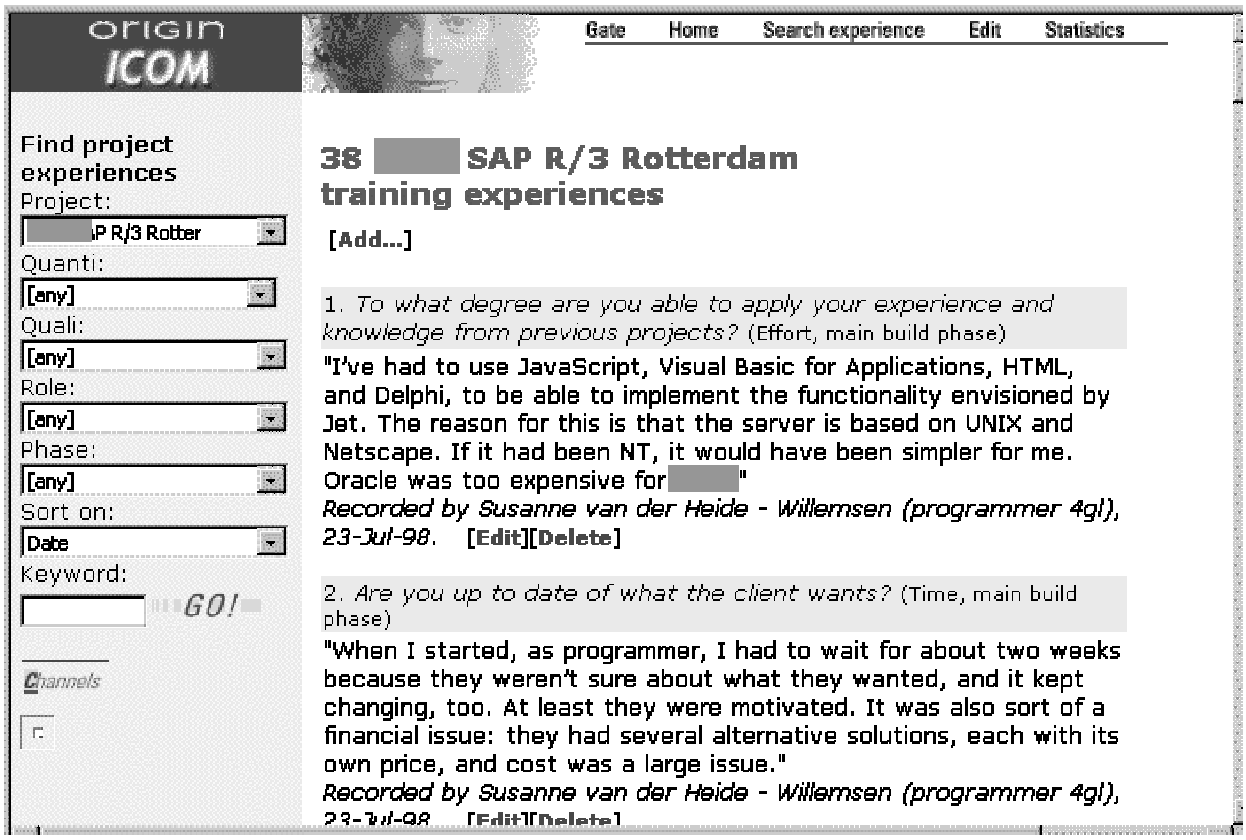


Figure 28. *The ICOM metric: Categorization of project problems and filtering options to find them.*

Contributing Experiences

The following procedure was employed for adding the project experiences to the metrics database described above:

1. Researcher makes meeting request with project team member (usually project manager) to conduct a project interview, with one or more employees.
2. Project team member sets date and location of interview. Often, the project interviews took place at the location site of the customer.
3. The Project interview is done, usually requiring 50 to 75 minutes if with one employee, 70-100 minutes if with a group of employees (more about this in chapter 6).
4. The Interview answers are sent back later by email to project employee for verification of correctness. Often, the project employee adds some slight corrections or remarks.
5. After confirmation of correctness, the interview answers are categorized and stored into the metrics experience database.

Interview sessions were structured using predefined questionnaires. The design of the early versions of the questionnaires is described in chapter 4; the design of the final version of the project questionnaires is described in chapter 6. All questions on the questionnaires were modeled in the web-based version of the metric, shown in Figure 28. This allowed for quick and easy retrieval of the answer to a particular question for a particular project. Interview answers were entered into the

experience database by the researcher rather than the users themselves. Initially, the researcher also categorized each interview answer given along exactly one of the project problem categories. The purpose of this was primarily for testing purposes in order to determine whether or not certain project experiences could be easily retrieved at a later stage. Later, three independent researchers did the analyses of the project interviews, thus ensuring a non-biased analysis of the projects. (See section 6.3). With this categorization available, we are now able to count the number of experiences reported on a certain problem category, and also the nature of the experience: positive or negative. This in turn allows us to determine if a positive trend in project experiences may be measured when a knowledge management solution such as Performer is in place (and used).

Conclusion

In this chapter, we have used the KnowledgeWorks methodology to design and implement a formal solution for knowledge management. The resulting instrument, called Performer, employs an objectives matrix to visualize the formal way of working in courseware projects. Knowledge elements and individual estimates can be connected to one or more appropriate objectives. Less time is required to find templates and best practices while working on projects. With the ICOM metric, we now have available a means to do measurements on the effectiveness of Performer.

This means that we have a knowledge management instrument in place that facilitates in addressing the needs for improvement as listed in chapter 4. Moreover, we can now test the hypothesis that this solution actually results in a more mature project process. We describe the main experiment to test the hypothesis in the next chapter.

6. Main experiment and results

In this chapter we report on the main experiment that we have conducted to test the hypothesis, using the solutions we describe in chapter 5. In chapter 4, we have conducted initial project interviews as part of the problem analysis, to determine the shape of our knowledge management solution (the pilot study); subsequently, we have conducted two types of measurements in a main experiment to test sub (*a*) and sub (*b*) of the hypothesis, respectively. These are reported on in sections 6.1 and 6.2. The results are further discussed in chapter 7.

6.1. Main experiment: measuring project management data

All research that we undertake is geared towards testing the hypothesis. A well-defined test requires a properly set up experiment; which means that it must be unambiguous and repeatable. Riffe *et al.* [RIF1998] states that "to [exactly] know what is needed before data are collected and analysed makes for a smoother, more efficient analysis and one that does not have to be redone".

The purpose of the main experiment that we describe in this chapter is to demonstrate the causal effect presumed in the hypothesis, namely the improvements that are caused by the use of our knowledge management solution in an educational multimedia project environment. One condition for the demonstration of a causal relationship is *time order*. This implies analyzing data that is collected both before the knowledge management solution is in place, as well as after the knowledge management solution is operational, in order to be able to determine the causal effect. A second condition necessary for a causal relationship is called *association*: varying degrees of employing knowledge management should be associated with similar degrees of the expected effect. The third requirement for demonstrating a causal relationship involves the discussion of "*rival explanations*" [RIF1998; OPP1992; BAA1995]. Rival explanations are all the potential alternative explanations for what has been observed as (apparently) a causal relationship. The list of rival explanations can be minimized by carefully defining boundaries, assumptions and constraints on the pilot study and main experiment. In our case, we discuss rival explanations for what we have observed chiefly in chapter 7. It is important to realize here, though, that few phenomena are themselves the results of a single cause, but rather of a set of causes, some of which are not controllable. In other words, it is virtually impossible to eliminate all rival explanations. The trick is to be able to make plausible the assertion that our solution is the most important cause of observed improvements.

When considering the hypothesis of more successful projects using a knowledge management solution, sub (*a*) of the hypothesis states that the estimates of basic project variables will be better. This statement in itself leaves many questions open. What are the "basic project variables" that are estimated in courseware projects? How can we determine estimates to be "better"? Clearly, we cannot set up an experiment that tests the hypothesis without answering these questions. We will use the same method for setting up the experiment as in section 6.1.

Step 1. Define the relevant content.
The basic question here is what will be needed to test the hypothesis, more specifically sub (*a*) of the hypothesis. In our case, these are educational multimedia projects that we have analyzed in the scope of this research project. Let us again define the sampling units, the recording units, the context units, and the analysis units.

The sampling units are those projects that have a sufficient amount of data (estimates and actual values) available to lend themselves for quantitative analysis. The recording units are those projects contained in the sampling units that satisfy the boundary conditions for projects described in chapter 3; thus, the recording units form a subset of the sampling units. In the scope of this research project, we were able to find ten courseware projects that meet these boundary conditions. In five of these, our knowledge management solution Performer was not used, while in the other five, Performer has been used. This distribution of Performer usage has been chosen specifically for this experiment, though it became increasingly difficult to withhold Performer from project teams when it is actually available to them; this trend is visible in Table 16, where we list the recording units.

Table 16. *Projects that were analyzed in the scope of the main experiment*

Rank	Project name	Brief description	Performer?	Year
1.	RapidLearn	Solutions for setting up CBT developmt by customer himself	*Yes*	*2000*
2.	Heavy ind. CBT	Training groups of employees of customer in using a device	*Yes*	*2000*
3.	I.B. CBT	Instructing sales people worldwide in a particular service	*No*	*1999*
4.	SAP R/3 CBT 1	Courseware as support for an SAP implementation	*Yes*	*1999*
5.	Mistral	Courseware as support for an SAP implementation, internally	*No*	*1998*
6.	Help facilities	For a particular banking service, offer usage help facilities	*Yes*	*1998*
7.	Alliance	Training software for employees in a sales situation (industry)	*No*	*1998*
8.	SAP R/3 CBT 2	Courseware as support for an SAP implementation	*Yes*	*1998*
9.	New Mail system	Courseware as support for a new email system	*No*	*1997*
10.	Logistics CBT	Courseware on CD-Rom to support logistics processes	*No*	*1996*

The context units are the factors that must be taken into account in order to retain a clear picture of what we are measuring. In this case, we need not worry about the context factors too much, because in this experiment, we are merely measuring basic recorded quantitative project variables. One context factor may be, for example, that the company of study spent a large amount of money on a particular project because they did that project for a first-time customer and they did not want to have the customer pay for everything the first time. If context factors like these were relevant in projects at all, then it is outside of our power to correct for these factors. We have the basic administered data to work with, including their possible context influences.

The analysis units are a quite different matter. By defining the analysis units we answer the question "what are the basic project variables that are used for estimates?" Project management literature [cf. DUN1996] lists at least three basic quantitative project variables that must be administered, namely calendar time, budget, and working hours (effort). Other researchers [GRE1996; MUS1987; PUT1996; SOF1995] suggest that two additional quantitative variables are required in order to be able to make calculations geared towards predicting project success. The most important of these additional quantitative variables is product size. As we have shown in section 5.4, for traditional software engineering there are various ways to calculate the product size, but for courseware, the situation is more complex. That is why we devised our own unit of size, the SIUN (see appendix E for the design of, and the rationale behind the SIUN).

The fifth required quantitative variable that is reported in the literature are the *defects* found in the product during the various stages of development. Again, in traditional software engineering, reliable estimates are now available on the expected number of defects to be found for a given type of software. This posed a problem for us, as defects were not estimated nor recorded at the company of study. To make matters worse, we could not agree upon what we should define as a defect, because there were multiple media involved. Should the fact that a navigational graphic is missing on screen X be regarded as a defect? We could, of course, define a severity scale of defects, ranging from cosmetic to system-critical, but such a scale is then defined from a technical, software development point of view. A defect that is cosmetic to a software developer may be experienced as very important to a customer or to an end user, and vice versa.

Since it turned out to be too difficult to define an agreed upon definition of a defect, we used the concept of risk instead. At the company of study, a risk analysis for courseware was available. This analysis consists of a questionnaire about various aspects of the development of the courseware product, and each question within the questionnaire had a weight assigned from zero up to six. One example of such a question is "the functionality requirements are likely to change from week to week". If this is the case for a particular project, we might rate this question a five or a six, while if this is not the case at all, we can rank it a zero. In a worst-case situation, where all the risks occur to a maximum extent, the risk factor of that project would be 1188, meaning 198 questions in the questionnaire. This worst case never happens in practice, of course. We can now assign a risk percentage value to each project to denote the risky-ness of that particular project, relative to the maximum value of 1188. With these two additional factors in place, we have now defined our analysis units: (1) calendar time, (2) budget, (3) working hours, (4) product size, and (5) risk factor.

Step 2. Specify the formal design.
In this step, we specify the procedure for carrying out the experiment, geared towards testing sub (*a*) of the hypothesis. For each of the projects defined in the set of recording units, we collected the quantitative project variables as described above. This was usually done in an informal meeting with the project administrator of the project (who was often project manager as well), where we discussed the various requirements of the project variables. We note here that the amount of quantitative data that was administered depended on the size and complexity of the project. Of all project variables that we registered, we must mention one exception, namely the risk factor; this was calculated for a few projects, but the eventual number of projects for which both an estimation and the actual outcome of the risk factor was available, turned out to be too small to be of any relevance to the experiment. This is one of the consequences of the empirical character of this research project: the researcher could not force project teams to co-operate on risk factor calculation if no budget was available to do this. That is why we had to settle for collecting four of the five quantitative project variables.

In Table 17, we provide a sample of one of the projects that we analysed, where we have collected both estimates and the actual outcomes of the four quantitative variables. This particular sample was about a project where Performer was not used (in fact, Performer did not even exist at that time). The project administrator provides the comments of the rightmost column, except for the "size" category where the comments are provided by the researcher (SIUN figure was calculated in retrospect).

Table 17. *Sample of one project whose quantitative variables were analyzed*

Calendar time	Estimate	Actual	Comment
Project kick-off	Week 11/96	Week 11/96	
Delivery of project plan	Week 11/96	Week 12/96	
Final delivery analysis report	Week 13/96	Week 15/96	
Delivery customer review comments	Week 12/96	Week 14/96	Had to wait for customer to deliver comments
Final delivery plan of approach	Week 12/96	Week 14/96	
Delivery of Help file comments	Week 16/96	Week 19/96	Had to wait for customer to deliver comments
Delivery of application structure	Week 17/96	Week 19/96	
Delivery of content specifications	Week 13/96	Week 16/96	
Delivery of user interface / layout	Week 13/96	Week 15/96	
Realization of C.B.T.	Week 17/96	Week 20/96	
Delivery of required assets	Week 16/96	Week 20/96	
Delivery of C.B.T. program	Week 18/96	Week 21/96	
Final delivery	Week 20/96	Week 22/96	
Cost	**Estimate**	**Actual**	**Comment**
Analysis phase	$6556	$6800	
Project management	$4625	$5008	
Realization phase	$18780	$19104	
Cost of assets	$3240	$3431	
Cost of meetings / formal communication	$3810	$3909	
Design phase	$12510	$12736	
Total project budget	$50000	$50000	
Effort (working hours)	**Estimate**	**Actual**	**Comment**
Functional analysis phase	72	89	Customer did not know what he wanted.
Production of assets	40	64	Much more than initially anticipated!
Design phase	152	178	Customer changed the requirements
Customer information analysis	24	40	More complicated than we thought.
Customer assets delivery effort	0	6	We thought we didn't need that, but…
Customer delivery activities	24	12	Customer didn't have much time for us.
Realization of the CBT	246	248	Neatly planned.
Product size (SIUNs)	**Estimate**	**Actual**	**Comment**
Computer Based Training	1022	1146	More assets than initially envisaged

We can calculate the *discrepancy* factor for each of these analysis categories separately; this is defined as the difference between the actual value and the estimated value. It is not possible to directly combine discrepancy factors of the four categories into one general discrepancy factor for each project, because each category employs a different *unit* (e.g., we cannot compare currency to calendar time). However, we can assign ranking numbers to each category if we want to create an aggregate overall view; we discuss this at the end of this section.

Step 3. Create analysis categories.
This step is about how analysts will know the data when they see it. We must ask ourselves what units of content we will place in the categories. We will place the units of content in the analysis categories as shown the second column of Table 18.

Table 18. *Units of content used in this part of the experiment (the discrepancy factor)*

Analysis category	Unit of content
Calendar time	Calendar weeks
Money	US Dollars
Effort	Working hours
Product size	SIUN (size unit, see section 5.4)
Risk	Risk ratio

Step 4. Operationalize.
Here, we explain how the variables in the study are to be measured and recorded. In our case, the way of measurement is fairly straightforward: for each of the four quantitative project variables, we record an estimate value and later on an actual value. The difference between these two is called a discrepancy. By averaging all records for a particular category, we get the discrepancy factor for that quantitative project variable. Initially, one would argue that we should take the sum of all records per category, but this is not realistic since usually only a subset of all quantitative values for each category is recorded, and records are often interrelated (not independent from each other). The project administrator of the given project initially does the recording of the records; later, the researcher copies relevant records into the ICOM project experience database (see section 5.4). The data that is available in the project experience database is used for analysis in the scope of this experiment. To test sub (*a*) of the hypothesis, we look at the discrepancy factors of each project, for both sets of projects. Our general expectation is to observe smaller discrepancy factors for the projects where Performer was used. If we can show statistically significant smaller discrepancy factors, we can then say that we cannot falsify sub (*a*) of the hypothesis.

Step 5. Specify population and sampling.
This step is about how much data will be needed to test the hypothesis. Analogous to the pilot study, we would like to analyze a minimum of six projects where Performer was not used, plus another minimum of six projects were Performer *was* used. In practice, we only managed to find five suitable projects for each of the two situations, resulting in ten projects in total; these are listed in Table 16. Although the theoretical minimum is only a rule of thumb, we may run the risk that our sampling size is too small to result in any statistical reliable outcome, depending on the unreliability threshold α that is chosen for the statistical analysis. In the main experiment we accepted that risk, with the argument that in this case it is better to conduct a quantitative experiment on a small sampling set than not to conduct an experiment at all.

Step 6. Pretest and establish reliability procedures.
This step is about how the quality of the data can be maximized. For this particular part of the main experiment, this step is not an elaborate one. The project administrator records the three basic quantitative project variables: calendar time, budget, and working hours; no reliability procedures are required to exert control over this; the PROMISE handbook provides for the definition of the activities involved. The SIUN variable is calculated through inspection of the product that is delivered at the end; if necessary, an informal meeting is held to identify exact numbers for kilobytes of graphics used and, if applicable, audio and video. Since the number of samples is clearly too small to assume a Gaussian distribution of the data, we will use a standard non-parametrical statistical analysis method to determine if there is a significant improvement. More specifically, we employ the Mann-Whitney non-parametric method [SPR2000] to determine if the distribution of the "No-Performer" set of projects differs significantly from the "Yes-Performer" set of projects. We shall choose the unreliability factor $\alpha = 0.05$ to assess proposition H_0: "*Discrepancy factors of post-Performer projects do not differ from discrepancy factors of pre-Performer projects*" to proposition H_1 "*Discrepancy factors of post-Performer projects are smaller than discrepancy factors of pre-Performer projects*". If the results indicate that we should reject H_0, we then take this decision with an unreliability factor $\alpha = 0.05$ (5%). If we subsequently decide to accept H_1, we do this with an unknown reliability factor β.

Step 7. Process the collected data, and interpret and report results.
The discrepancy factors of each of the ten projects that we have analyzed are given in Table 19. For each project, the four discrepancy factors that we have defined are listed. For the calendar time, the figure given is to be interpreted as calendar weeks. For the budget, the figure given means dollars. For the working hours, the figure given is to be interpreted as working hours. For the product size, the figure given is expressed in SIUNs. Please note that figures can only be negative if the average of all the records indicate that the actual values are under the estimates, e.g., less money was spent than estimated, and less hours were spent than estimated. Also note that the figures in Table 19 do not say anything about whether or not the project was regarded as a success by either the customer or the project team; this is the subject of the next part of the main experiment, described in section 6.2.

Table 19. *Discrepancy factors for ten projects analyzed in the main experiment, for units see table 18.*

Rank	Project name	Calendar time	Budget	Working hours	Product size	Performer
1.	"RapidLearn"	+0.33	+1028.44	+5.5	+412	*Yes*
2.	Zuigerkop	+3.0	+1530.56	+8.12	+78	*Yes*
3.	Inv.Banking CBT	+4.68	+8135.24	+36.32	-1016	*No*
4.	SAP R/3 CBT 1	+0.17	-6490.00	+27.21	+56	*Yes*
5.	Mistral	+5.2	+8640.46	+40.43	-227	*No*
6.	Help facilities	+1.16	+3554.24	+18.12	+553	*Yes*
7.	Alliance	+2.71	+1297.43	+12.83	+3391	*No*
8.	SAP R/3 CBT 2	+0.16	-485.17	-2.89	-131	*Yes*
9.	New Mail system	+1	+4661.00	+3.5	+736	*No*
10.	Logistics CBT[18]	+2.23	+209.57	+11.29	+124	*No*

We now conduct the Mann-Whitney test four times, one for each quantitative category. This allows us to determine for each of the categories if a significant change in the distribution may be concluded. For each quantitative category, we have two sample sets, namely the "No-Performer" sample set (called population set F_x) and the "Yes-Performer" sample set (called population set F_y).

We determine the actual value $\underline{v}_{ij}$ according to the following rules: $\underline{v}_{ij} = 2$ if $\underline{x}_i > \underline{y}_j$; $\underline{v}_{ij} = 1$ if $\underline{x}_i = \underline{y}_j$; $\underline{v}_{ij} = 0$ if $\underline{x}_i < \underline{y}_j$. The case that $\underline{x}_i = \underline{y}_j$ is very unlikely to happen, which means that our $\underline{v}_{ij}$ will probably always be an even number.

Calendar Time aspect
Ordered collection set x_i: { `1.00, 2.23, 2.71, 4.68, 5.20` } ($n = 5$)
Ordered collection set y_j: { `0.16, 0.17, 0.33, 1.16, 3.00` } ($m = 5$)

[18] This is the courseware project shown in Table 17.

$$\sum_{i=1}^{5}\sum_{j=1}^{5} \underline{v}_{ij} = (2+2+2) + (2+2+2+2) + (2+2+2+2) + (2+2+2+2+2) + (2+2+2+2+2) = 42$$

The statistical table of the Mann-Whitney test for $n, m \leq 20$ gives a critical c value of 8 for $\alpha = 0.05$, n=5 and m=5. The critical area range for our test lies in the area of $\{2.n.m - c, \ldots 2.n.m\}$ which translates to{42,…50}. We must therefore reject the calendar time part of H_0 if $\underline{v}_{ij} \geq 42$. Our $\underline{v}_{ij}$ is just within the critical range, which means that we reject the calendar time part of proposition H_0, with an unreliability factor of 5%. This means that we conclude that the discrepancy factor for calendar time is indeed statistically smaller with projects were Performer was used, meaning that estimates of calendar time did indeed improve.

Budget aspect

Ordered collection set x_i: { 209.57, 1297.43, 4661.00, 8135.24, 8640.46 } ($n = 5$)
Ordered collection set y_j: {-6490.00, -485.17, 1028.44, 1530.56, 3554.24 } ($m = 5$)

Analogous to the Calendar time aspect: $\underline{v}_{ij} = 40$
Please note that an implicit assumption here is that negative values of budget discrepancy (i.e., less money spent than estimated) are regarded as very positive; this is why we do not take the *absolute* values of the discrepancy records). Analogous to the previous calculation, the critical area range for our test lies in the area of $\{2.n.m - c, \ldots 2.n.m\}$ which again translates to{42…50}. We must therefore reject the budget part of H_0 if $\underline{v}_{ij} \geq 42$. Since or $\underline{v}_{ij}$ is not in the critical range, we cannot reject proposition H_0. In other words, the discrepancy factor of money spent does not become smaller, statistically speaking..

Working hours aspect

Collection set x_i: { 3.5, 11.29, 12.83, 36.32, 40.43} ($n = 5$)
Collection set y_j: {-2.89, 5.5, 8.12, 18.12, 27.21} ($m = 5$)
Thus, our $\underline{v}_{ij}$ =34. As in previous calculations, our critical area is in the range of {42…50}. We must therefore reject the working hours part of H_0 if $\underline{v}_{ij} \geq 42$. Since or $\underline{v}_{ij}$ is not in the critical range, we cannot reject proposition H_0. In other words, the discrepancy factor of the working hours involved, does not become smaller.

Product size aspect

For the SIUNs, we must take into account that we do not only want the discrepancy factor to become smaller, but also absolutely speaking smaller; we assume that to deliver less product than estimated is worse than delivering almost the product size that was promised. Therefore, we take the absolute values of the Siuns from Table 19.
Ordered collection set x_i: {124, 227, 736, 1016, 3391 } ($n = 5$)
Ordered collection set y_j: { 56, 78, 131, 412, 553 } ($m = 5$)
Our $\underline{v}_{ij}$ = 44. Similar to previous calculations, we must reject the product size part of H_0 if $\underline{v}_{ij} \geq 42$. Our $\underline{v}_{ij}$ is within the critical range, which means that we reject the product size part of proposition H_0. By definition, we then accept the product size part of proposition H_1 that the discrepancy factor for product size is indeed statistically smaller with projects were Performer was used, meaning that estimates of product size did indeed improve. One assumption that we need to make explicit here, is that the SIUN calculation itself contains a notable subjective-ness, namely that of the complexity

of each of the media involved. For example, our SIUN value for a courseware product gets higher if we rate the complexity of the production of graphics higher; but complexity is not an entirely objective notion.

Aggregation of measurement data

We can assign ranking numbers to the data records in Table 19 and then aggregate the ranking numbers to determine if there is an overall improvement for "Yes Performer" projects compared to "No Performer" projects. We assign the ranking numbers as follows: for each quantitative project variable, we assign the smallest discrepancy factor (the best result) the highest-ranking number in the range of {1…10} and the largest discrepancy factor (the worst result) the lowest ranking number in the same range. We then take the average of the ranking numbers for each project. If we group these ranking numbers for the two types of projects, we get table Table 20 below.

Table 20. *Aggregated ranked data for the discrepancy factors for the ten projects.*

Rank	Project name	Ranking number	Performer used?
1.	RapidLearn	7	*Yes*
2.	Zuigerkop	6	*Yes*
3.	Investment banking CBT	2	*No*
4.	SAP R/3 CBT 1	8	*Yes*
5.	Mistral	2.25	*No*
6.	Help facilities	4.5	*Yes*
7.	Alliance	4	*No*
8.	SAP R/3 CBT 2	9	*Yes*
9.	New email system	5.5	*No*
10.	Logistics CBT	6.75	*No*

We can now once again conduct the Mann-Whitney test on this aggregated set of data. Our collections become:
Ordered collection set x_i: {4.5, 6, 7, 8, 9 } ($n = 5$)
Ordered collection set y_j: { 2, 2.25, 4, 5.5, 6.75 } ($m = 5$)

Our $\underline{v}_{ij} = 44$, which is within the critical range of {42…50}. So we do observe smaller discrepancy factors in general in projects that used Performer. So on an aggregate level, we do observe a statistically significant improvement.

Summary of results of the first part of the main experiment

We have observed a positive shift in the discrepancy factors for the aggregated set of variables, meaning that the discrepancy factor indeed becomes smaller. If we summarize this part of the experiment from a statistical point of view, we can conclude that the discrepancy factors of calendar time and product size become smaller with projects that have used Performer, meaning that the quality of the estimates improves. As we will note in the discussion of chapter 7, especially the calendar time output result seems logical because the main advantage of Performer is felt to be the fact that it saves time for people in their activities.

Discrepancy factors of money spent and working hours made do not become statistically smaller with projects that have used Performer. On an aggregate level, however, we do note a statistically

significant improvement overall. Therefore, we cannot reject sub (*a*) of the hypothesis. We do note that estimates for projects depend on more than just an appropriate solution for knowledge management in place. We will discuss this in chapter 7. Let us now examine the "softer" aspect of process improvement, namely the project experiences.

6.2. Main experiment: project experience content analysis

Sub (*b*) of the hypothesis states that using our knowledge management solution in educational multimedia projects results in significantly more positive project experiences and significantly less negative project experiences. With the interview analyses, we have collected a large set of project experiences about a variety of issues in educational multimedia projects. We can analyze this project experience data set to test sub (*b*) of the hypothesis. Basically, we will be counting the project experiences of the entire set of projects that we analyze, and assessing the nature of each experience. Similar to section 6.1, there are some questions that we need to answer first. What do we mean by "*use* of our management solution"? How will we determine if there are more "positive" project experiences and less "negative" project experiences for our two groups of projects: without Performer and with Performer?

First of all, "Using" Performer means the accessing of descriptions of objectives and / or knowledge elements at least once a week by at least one project member of the project. This is a theoretical minimum value; in practice, we observed a much heavier use of Performer for all projects that we chose. At early stages of the project, it is likely be the project manager who consults Performer, for example for a standard project handbook and a quality assurance guide. At later stages of the project, it will likely be the development team members who consult Performer for design and analysis templates, links to relevant Internet sites for downloadable files, and other resources. The exact amount and kind of resources accessed in the ten projects was logged in the database in {`userid, date, element_id`} triples, but we did not analyze these logs because of the labor-intensive requirement. We did, however, ask the project team members to what degree they had used – or were using – Performer, as a standard part of our interview sessions.

We will use the same method for setting up this part of the main experiment as we have done in section 6.1, and we will again use the Mann-Whitney test to address sub (*b*) of the hypothesis, for each project issue category that we have defined. Let us now set up the measurement activities.

Step 1. Define the relevant content.
This step is about what content will be needed to test the hypothesis. We once again use Table 5 on page 51 to define the relevant content. Our *study units* consist of all project experiences that project team members in educational multimedia projects have provided to the experience database. The *sampling units* are those experiences that can be classified as either "complaining" about a problem (negative), or alternatively "enjoying" a success story (positive). Neutral experiences such as *"the customer's project manager has left the company, so we now hope that we can continue working for the customer if our new project bid is good"* are not part of the sampling set. On the other hand, reported experiences such as *"Poor network facilities caused significant slowdown of work progress during the realization phase"* would be part of the sampling set.

The *recording units* are the elements of the sampling units in projects that meet the boundary conditions as defined in chapter 3. Thus, the recording units are a subset of the sampling units. The

context units are all factors that must be taken into account for the recording unit to be unambiguous. This is because project experiences can depend on a variety of contextual (environmental) factors. An example of a context unit is given in the following experience: *"Communication with Mr. X from company Y was very troublesome; however, we've dealt with this particular customer before and he has a reputation of being very picky about details, so we were able to take that into account"*. The context unit in this case is the known reputation of Mr. X of previous projects.

The analysis units, finally, are those content units that are used as the basis for statistical analysis. Since we are interested in recording both negative and positive experiences, the analysis units consist of the spectrum of very negative project experiences to highly positive project experiences. In order to reduce complexity for the interview analyst(s), we defined the analysis units for both the pilot study and the main experiment as shown in Table 6 on page 52.

In Table 21, we list the projects that satisfy the conditions we set in chapter 3. The list is sorted on the column "*year*" (descending). All project experiences reported for these projects together make up the recording units; it is these experiences that we shall analyze to test sub (*b*) of the hypothesis.

Table 21. *Projects analyzed using the ten interview questionnaires.*

Rank	Project name	Brief description	Year
1.	"RapidLearn"	Solutions for setting up CBT development by customer himself	*2000*
2.	Heavy industry CBT	Training groups of employees of customer in using a device	*2000*
3.	Inv.Banking CBT	Instructing sales staff worldwide in providing a particular service	*1999*
4.	SAP R/3 CBT 1	Courseware as support for SAP implementation	*1999*
5.	Mistral	Courseware as support for SAP implementation, internally	*1998*
6.	Help facilities	For a particular banking service, offer usage help facilities	*1998*
7.	Alliance	Training software for employees in a sales situation (industry)	*1998*
8.	SAP R/3 CBT 2	Courseware as support for SAP implementation	*1998*
10.	New Mail system	Courseware as support for a new email system being introduced	*1997*
11.	Logistics CBT	Courseware on CD-Rom as support for internal logistics	*1996*

Step 2. Specify the formal design.
This step is about how the hypothesis shall be tested in a reliable and repeatable manner. One issue we must address is the objectiveness of the assessment of project experiences. In order to ensure a non-biased, non-subjective analysis of the project experiences, we decided to create a *project experience analysis group* consisting of three independent senior researchers, who would – again independently, without seeing each other – analyze each of the project interviews. The word "independent" is to be interpreted as: "not in any organizational way related to the company of study, namely Atos Origin. The three researchers use a specific, agreed-upon *codebook* as a guideline for analyzing the interviews.

The codebook, described in step 4 below, is agreed upon by all researchers involved, and contains definitions and examples of all project problem categories as investigated in the pilot study, described in section 6.1. In addition, it defines the analysis units scale as described in Table 6 on page 52. Using the codebook, the following procedure is then employed for each project:

1. Author conducts interviews for project *x,* and collects these into one document. Company names are made anonymous, but context of the project is explained.

2. Author sends interview to each independent researcher for analysis. This document contains four columns: (1) the question; (2) the experience that was reported; (3) assessment of the main problem category that this experience is about, plus positive/negative assessment; (4) assessment of an optional second problem category that this experience is about[19]. A sample of such a document is shown in Figure 29 (where assessments are already entered, for illustrative purposes).
3. Each researcher analyzes the project experiences. At least the third column and optionally the fourth column are entered for each experience.
4. Author collects the analyses of the three researchers and combines them into one document. A sample of such a collected document is given in appendix B on page 159.
5. Author determines the inter-reliability factor for this collective analysis (see step 6 below) and summarizes the results, which are stored for use in this part of the experiment.

Question	Answer	Cat.1		Cat.2	
Are there any problems involving team communication?	Relational aspects of communication in the entire team were okay. Process-aspects were okay, but the procedural communication caused some problems. It was not clear what the programming team wanted, and vice versa what the specification team wanted.	**CO**	**1**	**PR**	**1**
Are there any problems regarding communication with the customer?	Communication to customer was okay. A frustrating point was that we just had to make the CBT, and that's it. So we didn't really have any communication to the customer, that was organised through Frits. For the customer, we're welcome, but that's all.	**CU**	**1**	**PS**	**-1**
Were there any problems regarding project meetings?	Project meetings were okay. In general, they were brief because the specification team demanded it because of time pressure. We got the information that we needed (very brief, concise), a bit like a factory. This because of high pressure.	**PR**	**2**	**CO**	**2**
Can you say something about the project pressure?	Project pressure is very high. Cause: a wrong estimation of the number of specifications of work instructions that could be made in a given period of time. Origin was cheap, because Origin also payed some of it themselves to so they can keep the source.	**PS**	**1**		

Figure 29. *Sample of project interview sent to each independent researcher.*

Step 3. Create analysis categories.
In a session dedicated towards creating the final set of analysis categories, the three researchers mentioned above, plus the author, constructed a set of project problem categories, called "themes", that would serve as main input for the codebook to be used during the project interview analyses. These themes were derived from the problem analysis and the outcome of the pilot study described in section 6.1. For the experiment, we will use the word "theme" as a synonym for agreed-upon project problem category. The definitive list of themes is:

1. Project management (Short code: `PR`)
Often, the project management is seen as a task solely for the designated project manager of a (courseware) project. Other definitions see project management as a group responsibility: the success of the project is a result of the effort of the whole team, and this explicitly includes project management tasks from that whole team. Our theme agrees to the latter definition.

[19] The optional fourth column was added because a project experience is often about more than one problem category.

2. Qualification of skills (Short code: QU)
This theme is about the skillfulness of the project team members (including the project manager, excluding the customer). Often, project process problems can occur because one or more team members are simply not yet experienced enough to have foreseen the problem. Most often, they will learn from their mistakes and avoid that pitfall in projects later on. Still, almost every multimedia project has junior employees, and therefore this theme is a valid one. The knowledge management solution Performer is especially good at tackling this problem, as it offers handy knowledge elements and individual estimates.

3. Communication (Short code: CO)
In the end, one might say that almost all problems are due to poor communication: if we would communicate in a perfect way all the time, we would not have any problems. But that does not hold true, because even if we could communicate perfectly, we still cannot foresee all pitfalls on the way ahead. With this theme we mean problems that have to do with communication in the most direct way; so if a problem is only vaguely attributable to communication, then this may not be the proper theme for that problem.

4. Technological (Short code: TE)
With this theme, we refer to problems that are of a direct technological nature in the sense of technology: a network that does not function properly; development tools that are far from robust; very slow connections to the Internet; conversion utilities that do not work well, and so on. In other words, you should be able to "touch" the cause of the problem, as it were. Examples of these are given in the description of the codebook below.

5. Psychological (Short code: PS)
The problems in this category are of a much harder to define nature than most other themes. When a project is in danger of running out of budget or calendar time, a certain amount of stress occurs, not just within most team members, but for the entire group, too. Perhaps some team member feels that things are going completely the wrong way but is afraid to speak up for it. These kinds of things are categorized as psychological problems. Their impact on the success of a multimedia project is often not objectively determinable, but is very present and real nonetheless.

6. Customer/user. (Short code: CU)
This theme is mentioned last, but it is certainly not the least important. The classic example of a problem of this category is the customer who fails to deliver content on time, even if that deadline was formally agreed upon. A difficult point is here is communication aspect with the customer: if a problem is mainly due to poor communication, then a problem may be categorized as Communication. However, if, for that particular problem, there were also aspects that are due to the acting of the customer, that problem would be considered to be Customer/user.

Step 4. Operationalize.
The heart of a content analysis is the codebook that explains how the variables in the study are to be measured and recorded. We use the themes agreed upon above, plus our analysis units {-2, -1, 1, 2}, as the basis for our codebook. The purpose of the codebook is to maximize the inter-reliability between the three researchers. If they all agree upon when to dub an experience as "technological", or "communication", for example, this will increase the quality of the analysis. To better illustrate the various project issue "themes" the codebook provides some examples for each

theme, and we reprint these here for clarity. The examples given are taken either from interviews that were already analyzed by the researchers, or otherwise from projects that the researchers, for a variety of reasons, will not analyze.

Project management (PR)
Q: Are there problems regarding the working environment right now?
A: "Well, the office space is rather poor; there are too many people in a small room. Four is a lot, and at times there were five. Three would be acceptable."
Explanation: This is clearly a project management issue: the team should be able to work in an acceptable office environment if we want a successful project process. Obviously, this has not been taken care of in a satisfactory way. Also note that since we do not know whether or not the office space is at customer location and the customer was not able to supply the team with a better room, we have no reason to dub this problem Customer/User (CU).

Q: What are current problems in managing this project?
A: "The role of project management is not always overly clear. Normally, the project team leader is someone from the customer, but the project administrator sometimes also plays this particular role."
Explanation: we dub this one "PR" because the main cause of the problem is poor project management. There are also aspects of customer (CU) and communication (CO) here

Qualification of Skills (QU)
Q: What would you like to see in an experience database for future use?
A: "It was hard to construct the planning because I'm not skilled in estimating high-end offset tasks. What I would like to know is how long a professional offset agency would need to do their job, so that we can incorporate these things in our planning."
Explanation: while this answer seems to be about planning, implicating project management, the answer really is more about qualification of skills, namely the planning skills on required effort of offset tasks.

Q: What disciplines do you feel are missing on the project right now?
A: "A higher degree of availability of SAP content matter expertise would have been most welcome. Plus… a layout-interaction designer; now we're all doing user interface things ourselves."
Explanation: this is a tricky one. You would probably say that this is about Qualification of Skills (QU) because the team is in need of some disciplines that are not on the team. However, if it is made explicitly clear that better project management could have solved this one, it would be more appropriate to dub it "PR". Since this is not the case here, we dub it "QU".

Communication (CO)
Q: What were important problems regarding project communication?
A: "Communication aspects. Project management did not always communicate sufficiently; each time I had to start something new I did not have the required content available."
Explanation: although communication may be seen as an aspect of project management, it is really an aspect of all organizational aspects of a project. The cause of the problem here is mainly poor communication; therefore we dub this question "CO".

Q: What are communication problems within the team?
A: "The project team itself has always been able to work without annoying miscommunications, fortunately, and I think that is a critical success factor for multimedia systems, because if you've got that problem, you lose a heck of a lot of time."
Explanation: Again, though good communication sometimes seems to be a matter of managing the project well, some miscommunications just cannot always be avoided. So we dub this problem CO. Also note that this would be a positive occurrence of this problem.

*Technological (*TE*)*
Q: What are technological problems during this project?
A: "I am quite frequently bothered by a full hard-disk, the one advantage of that is that it forces you to clean up!"
Explanation: This is clearly a problem that was caused by technology. You might say that this could have been prevented by good project planning, but it is not always foreseeable how much hard-disk space is required for a project. Also note that although there is a problem, the experience is not wholly negative. So this could either be classified as a negative technological experience, but also as a positive one. This has no influence on the reliability number as long as the category "TE" is chosen.

Q: Are there problems regarding technical infrastructure of this project right now?
A: "Email was quite slow, though. I think I've been waiting for email for about 1 hour per week on average. For the fax, this is more like one and a half hour of waiting. Of course, you don't do just nothing during that time, but still…"
Explanation: This is clearly a technological problem that was probably not foreseeable. In other words, we dub this "TE" instead of "PR".

*Psychological (*PS*)*
Q: How do you experience project pressure? Are there things that you think should be said aloud?
A: "At one point I've felt that too high demands have been placed upon me in the project, when we were at the point of waiting for a lot of assets which we didn't seem to get, and we needed to go on."
Explanation: when someone says that "too high demands" were placed, a certain threshold has been passed, and we then dub such a problem as "psychological".

Q: What's the current project pressure for this project?
A: "There are conflicts on opinions between project members, sure, and why ever not - I think it's good for the project."
Explanation: Although this clearly has to do with psychological problems, and we dub this one as PS, this is an example of a positive experience (at least in the eyes of the person who said this – if somebody else experiences that same problem as negative, we would probably categorize this as negative.)

*Customer/User (*CU*)*
Q: Is the project still going according to project planning? If not, why not?
A: "No, we're one week late, but we've communicated that and the customer understands. The subject matter expert is not available all the time, even though he was scheduled to be."

Explanation: although this seems to be a project management issue, the cause of the problem is really not the fault of the project manager or any other team member. The customer simply did not have the required expertise available, even though he committed to it. Therefore, we dub this problem "CU".

We have now provided definitions and examples for the categorization of project issues on certain themes. Together with the definition of our analysis units, the researchers are now fully equipped to analyze project experiences.

Step 5. Specify population and sampling.
This step is about how much data will be needed to test the hypothesis. In Table 22 we list the frequency of use of each of the interview questionnaires defined in the pilot study. The role of the customer is not listed for frequency reasons. Although a few interviews with customer representatives were conducted, the total number of interviews with role {C} was obviously too small to be of any significance to the research results. The main reason for this lack of customer reviews was a shortage of time on both the customer's side and the researcher's side. Therefore, Table 22 is restricted to roles {A} and {B}.

The set of projects includes all projects that were analyzed after the information analysis was complete. The fact that the role of project manager is clearly less well represented than role {B} is caused by project interviews with the entire project team. The project manager was often present with these interview sessions, too.

Table 22. *Frequency of use of the nine main project interview questionnaires.*

Role	1: At project start	2: During project	3: In retrospect
A: *Project manager*	*A1*: **5**	*A2*: **7**	*A3*: **3**
B: *Project team*	*B1*: **9**	*B2*: **26**	*B3*: **8**

The 58 interviews collectively resulted in over 1900 project experiences that were analyzed by the researchers during two and a half years. The researchers assign experiences to at most two themes.

Step 6. Pretest and establish reliability procedures. How can the quality of the data be maximized? We have already mentioned the inter-reliability factor as a means to maximize the quality of the analyses. It should be determined for each pair of researchers and can be calculated for each project according to a description by Riffe *et al.* [RIF1998]. The basic formula for calculating inter-reliability is as follows:

$$Pi = \frac{\%OA - \%EA}{1 - \%EA}$$

P_i is calculated for each pair of researchers; this means that for each project interview, we did three calculations. The expected agreement (EA) is calculated by multiplying each of the percentages occurrence of each project category for a project. The observed agreement (OA) is calculated by counting the number of experience records for which the two researchers agreed upon at least one theme. The formula implies that the inter-reliability is always in the interval {0…1} or from 0% to 100%. Riffe reports that a minimum inter-reliability level of 80% is usually regarded as an accepted boundary.

In addition to defining inter-reliability, the researchers have conducted a pre-test for one of the projects involved; this was the first time that the codebook was used. With these activities, we have now prepared this part of the main experiment according to the required steps.

Step 7. Process the collected data and interpret and report results.
In this step, we address the question: "Has the research hypothesis been tested successfully?"

The time available for a project interview was often limited by the amount of time each person being interviewed had available for an interview. Since there was no formal budget reserved for interviews for project team members, the hours that were spent on being interviewed had to be charged to the customer. Fortunately, this has never prevented any interviews from taking place (although in most cases the customer *did* want to know why the interviews were held in the first place, and the customer usually also wanted the final results of the interview). The budget restriction mentioned above *did* have influence on the time that the interviews lasted: most of the time the project manager tried to agree on a maximum interview time of one hour. Despite of this restriction, most project interviews were done in an easy-going setting and could last for up to a hundred minutes. In Figure 30, we list the frequencies of duration of 45 of the 58 project interviews (the duration of the remaining project interviews was not recorded).

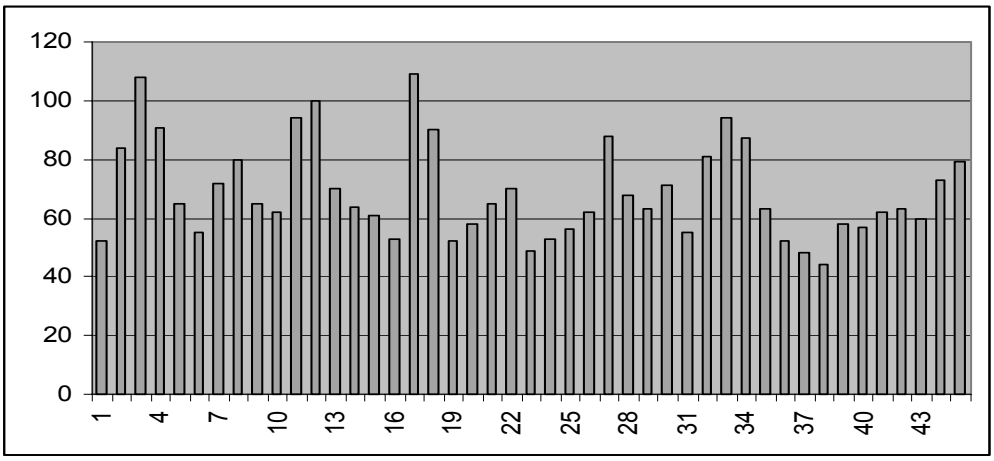


Figure 30. *Interview durations in minutes of 45 project interviews.*

In appendix B, we list a sample of a project analysis by the three independent researchers. We will not list all of the analyses here but we will discuss the outcome. In Table 23, we summarize the balance of negative and positive project experiences for our sample set of ten projects. The *balance* column is an absolute summation of the assessment of the analysis units ranging from –2 to +2. The *negative experiences* column lists the absolute number of negative experiences that are reported; similarly, the *positive experiences* column lists the absolute number of positive experiences. Finally, the *percentage* columns indicate the relative position of the respective column to the absolute number of experiences reported. Note that it is possible for a balance to be positive, while there are more negative experiences than positive experiences: in such a case, the positive experiences were apparently more positive than the negative character of the negative experiences.

Table 23. *Positive and negative project experiences.*

Project	Performer used?	Balance	Negative experiences	neg%	Positive experiences	pos%
Project 1	No	1	123	0.51	120	0.49
Project 2	No	-18	164	0.68	78	0.32
Project 3	No	-44	171	0.96	7	0.04
Project 4	No	-12	29	0.24	90	0.74
Project 5	No	-57	225	0.93	18	0.07
Project 6	Yes	2	143	0.51	135	0.49
Project 7	Yes	-14	116	0.69	52	0.31
Project 8	Yes	12	63	0.39	98	0.61
Project 9	Yes	18	37	0.27	101	0.73
Project 10	Yes	-30	88	0.56	69	0.44

When summarizing Table 23 for the two groups of projects, in other words: grouping the results by the column "Performer used?", we get Figure 31. The numbers given are in percentages, derived from the percentage columns above. The shift in positive project experiences for the "Yes Performer" ("Post") case is clearly evident.

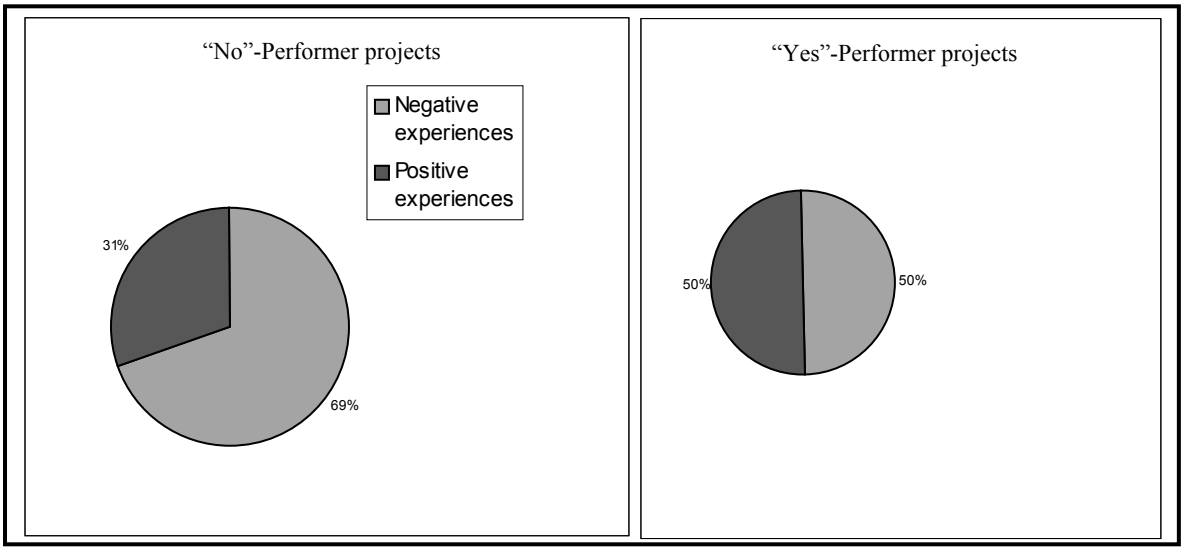
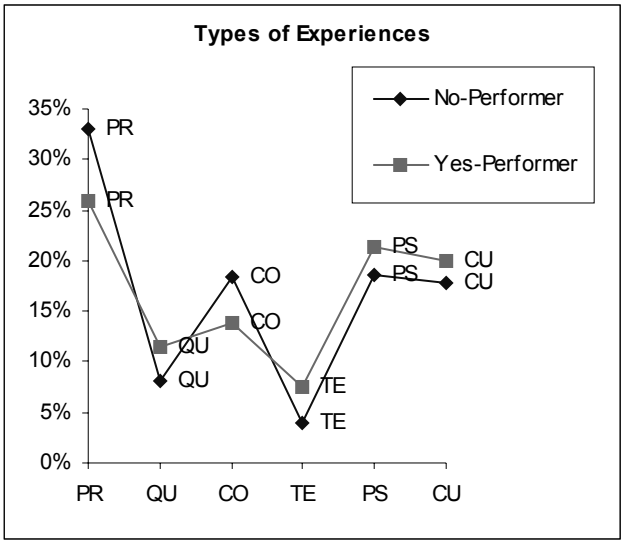


Figure 31. *Shift in nature of project experiences without regard to themes.*

In Figure 32, we list the relative occurrences of the various themes for the "No Performer" projects and the "Yes Performer" projects, without regard to the negative / positive character. As may be expected, there are no obvious shifts in the distribution of the themes. Most problems that are reported are about project management, psychological issues, and customer/user relationship.



Explanation:
The relative occurrence of each theme is shown here in two lines. The "No-Performer" line summarizes all experiences of the projects that did not use Performer, while the "Yes-Performer" line summarizes all experiences of the projects were Performer was used. The percentage values are printed for each category / theme.

Figure 32. *Distribution of themes for "No-Performer" and "Yes-performer" projects.*

Finally, we have counted the number of positive and negative project experiences per project category (theme), which resulted in two graphs, both shown in Figure 33. The top graph denotes the positive project experiences for each theme, while the bottom graph denotes the negative project experiences per theme. For each theme, the left bar denotes the "No Performer" projects, while the right bar denotes the "Yes Performer" projects. We see that the bars for all themes show a shift in the positive direction, except for the Technological theme [TE]. We will use Figure 33 as the basic data with which to test sub (*b*) of the hypothesis. We will not reject sub (*b*) of the hypothesis if the aggregated ranked data shows a statistically significant improvement.

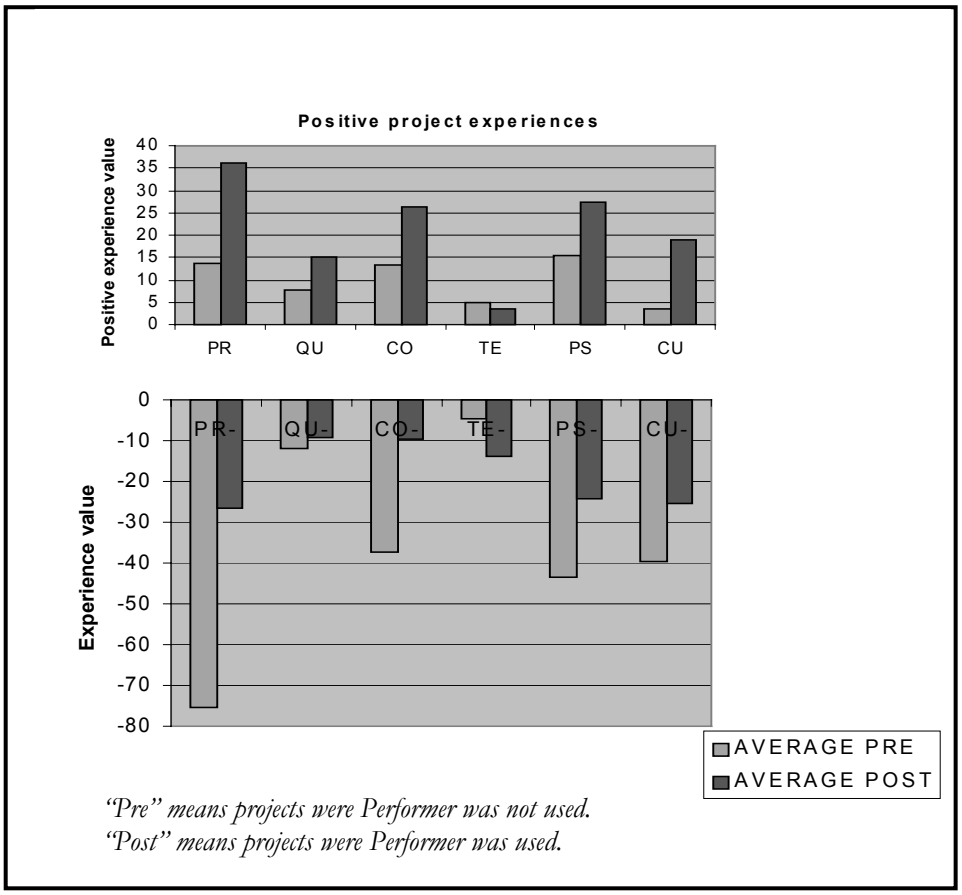


Figure 33. *Assessment of experiences per project problem category (theme).*

First of all, we conduct the Mann-Whitney test for each of the six themes separately to determine a change in distribution of the various experience summations. More formally speaking: we assess for each theme the proposition H_0: "post-Performer projects do not have significantly more positive project experiences than pre-Performer projects", to proposition H_1: "post-Performer projects do result in significantly more positive project experiences than pre-Performer projects". Since the hypothesis also talks about less negative experiences, we will conduct the same test for the negative experiences. We start by looking at positive project experiences

We will again employ an unreliability threshold $\alpha = 0.05$ to assess H_0. Since for both situations we have five projects, both the *n* and the *m* are 5, analogous to section 6.1. Since the hypothesis assumes higher values for the "Yes-Performer" projects, x_i is the collection of "Yes-Performer" experiences, while y_j is the collection of "No-Performer" experiences. The numbers in the collections denote the summation of all "positivism" for each of the projects involved. Wherever a "0" (zero) occurs, this means that there were no positive project experiences for that theme for that project.

Project management – positive experiences
Ordered collection set x_i: {17, 23, 39, 41, 61} ($n = 5$)
Ordered collection set y_j: { 1, 5, 7, 10, 45} ($m = 5$)

$$\sum_{i=1}^{5}\sum_{j=1}^{5} \underline{v}_{ij} = (2+2+2+2) + (2+2+2+2) + (2+2+2+2) + (2+2+2+2) + (2+2+2+2+2) = 42$$

The statistical table of the Mann-Whitney test for $n, m \leq 20$ gives a critical $\boldsymbol{c}$ value of 8 for $\alpha = 0.05$, n=5 and m=5. The critical area range for our test lies in the area of $\{2.n.m - \boldsymbol{c}, \ldots, 2.n.m\}$ which translates to{42…50}. We must therefore reject the "project management – positive experiences" part of H_0 if $\underline{v}_{ij} \geq 42$. Our $\underline{v}_{ij}$ is just within the critical range. We accept the "project management – positive experiences" part of proposition H_1, meaning that experiences about project management are indeed significantly more positive.

Qualification of skills – positive experiences
Ordered collection set x_i: {3, 12, 13, 23, 25} (n = 5)
Ordered collection set y_j: {0, 5, 6, 12, 15} (m = 5)
Our $\underline{v}_{ij}$ = 37. The critical area range for our test is the interval {42…50}. We must reject the "qualification of skills – positive experiences" part of H_0 if $\underline{v}_{ij} \geq 42$. Our $\underline{v}_{ij}$ is outside the critical range. We therefore cannot reject the Qualification of skills – positive experiences part of H_0. In other words, project experiences about qualification of skills are not significantly more positive.

Communication – positive experiences
Ordered collection set x_i: {17, 17, 20, 37, 40} (n = 5)
Ordered collection set y_j: { 1, 4, 5, 23, 34} (m = 5)
Our $\underline{v}_{ij}$ = 38. The critical area range for our test is the interval {42…50}. We therefore cannot reject the communication – positive experiences part of H_0. In other words, project experiences about communication are not significantly more positive.

Technological issues – positive experiences
Ordered collection set x_i: {0, 1, 5, 6, 6} (n = 5)
Ordered collection set y_j: {0, 0, 8, 8, 9} (m = 5)
Our $\underline{v}_{ij}$ = 18. The critical area range for our test is the interval {42…50}. We therefore cannot reject the technological issues – positive experiences part of H_0. In other words, project experiences about technological issues are not significantly more positive (but also not significantly more negative).

Psychological issues – positive experiences
Ordered collection set x_i: {16, 16, 29, 35, 41} (n = 5)
Ordered collection set y_j: { 0, 4, 11, 24, 38} (m = 5)
Our $\underline{v}_{ij}$ = 38. The critical area range for our test is the interval {42…50}. We therefore cannot reject the psychological issues – positive experiences part of H_0. Project experiences about psychological issues are not significantly more positive.

Customer/User relationship – positive experiences
Ordered collection set x_i: {12, 13, 15, 23, 32} (n = 5)
Ordered collection set y_j: { 0, 3, 4, 4, 7} (m = 5)
Our $\underline{v}_{ij}$ = 50. The critical area range for our test is the interval {42…50}. Our $\underline{v}_{ij}$ is within the critical range. We accept the "customer/user relationship – positive experiences" part of proposition H_1, meaning that the use of Performer results in significantly more positive project experiences about the customer/user relationship.

The inter-reliability level for the ten projects that were analyzed was 0.807 (or 80.7%) on average, which is just above the required minimum level of 80%. We did notice that the inter-reliability level slowly increased as additional interviews were analyzed.

The calculation for the negative project experiences is done in an analogous way and we shall not fully reprint these calculations here. We summarize the results of this set of tests in Table 24. The actual values of our $\underline{v}_{ij}$ are reprinted in brackets behind the conclusion for each theme.

Table 24. *Quick reference table to test sub (b) of the hypothesis.*

Experiences	PR	QU	CO	TE	PS	CU
Positive exp.	Yes (42)	No (37)	No (38)	No (18)	No (38)	Yes (50)
Negative exp.	Yes (48)	No (31)	Yes (46)	No (43)*	No (35)	No (32)

Aggregation of data

Similar to section 6.1, we can assign ranking numbers to the project experience records for each project, and then aggregate the ranking numbers for all themes to determine if there is an overall improvement for "Yes Performer" projects compared to "No Performer" projects. We assign the ranking numbers as follows: for each theme, we assign the project with most positive average project experience (the best result) the highest-ranking number in the range of {1…10} and the most negative average project experience (the worst result) the lowest ranking number in the same range. To this end, we consider the average project experience ratio for each theme as one number. In other words, we do not distinguish between negative and positive values any longer; we take the average experience ratio. We then calculate the average of the six themes for each project. If we group these ranking numbers for the two types of projects, we get Table 25 below.

Table 25. *Aggregated ranked data for project experiences of the ten projects.*

Rank	Project name	Ranking number	Performer used?
1.	RapidLearn	5.8	*Yes*
2.	Zuigerkop	9.2	*Yes*
3.	Investment banking CBT	2	*No*
4.	SAP R/3 CBT 1	8.3	*Yes*
5.	Mistral	5.3	*No*
6.	Help facilities	4.2	*Yes*
7.	Alliance	2	*No*
8.	SAP R/3 CBT 2	7	*Yes*
9.	New email system	6.7	*No*
10.	Logistics CBT	4.5	*No*

We now conduct the non-parametric Mann-Whitney test to determine if the overall project experiences are indeed significantly higher with "Yes Performer" projects. The collection set $\underline{x}$ consists of the projects that did use Performer, while collection set $\underline{y}$ consists of the projects were Performer was not used.

Ordered collection set x_i: `{4.2, 5.8, 7, 8.3, 9.2}` ($n = 5$)

Ordered collection set y_j: `{2, 2, 4.5, 5.3, 6.7}` ($m = 5$)

Our $\underline{v}_{ij} = 42$. The critical area range for our test is the interval {42…50}. Our $\underline{v}_{ij}$ lies just within the critical range. We therefore cannot reject sub (*b*) of our hypothesis and we conclude that project experiences as a whole do indeed become significantly more positive with courseware projects that use Performer. Looking at Figure 31, we see that positive experiences were 19% more positive for "Yes Performer" projects compared to "No-Performer" projects.

Summary of the results of this experiment

We have said to reject sub (*b*) of the hypothesis if the aggregated data of the themes for the two Performer usage cases do not result in significantly more positive project experiences. According to Table 25 and the analysis that follows, we cannot reject sub (*b*) of the hypothesis. We therefore conclude that project experiences do become significantly more positive for projects that use Performer, and our experiment indicates a shift in the positive direction of 19 percent. On an individual theme level, we observed that the effectiveness of our knowledge management solution is especially evident for experiences about project management and the customer/user relationship (more positive), and about project management and communication (less negative). There is one peculiar outcome, denoted with the asterisk in Table 24: the use of our knowledge management solution apparently results in significantly more *negative* project experiences for technical issues (TE). We note, however, that of all experiences reported the technological issues were the least, and therefore the relevance of this outcome may be doubted. Additional research should determine whether the number of technological problems does indeed increase, or that it is merely an aspect that employees now feel more keenly because all other aspects have improved.

On a quite different level, namely when asking the opinions of those that use Performer for their daily project tasks, Performer is also experienced as a success because it allow employees to do much more in less time. This is also exemplified in the number of active Performer users (over a thousand at the time of publication of this thesis). We will further discuss this in chapter 7.

7. Discussion

In this chapter we discuss the outcomes of the pilot study and the main experiment, and the implications that the results have on our research hypothesis. Then, taking the synthesis on a more abstract level, we discuss whether or not the implemented solution has actually solved the problems identified in the problem analysis described in chapter 4. On a still higher abstraction level, we discuss whether or not knowledge management is a suitable instrument for improving the maturity of the process of producing educational multimedia projects. The discussions described in this section eventually culminate in the general conclusions about this research project, which we present in chapter 8.

7.1. Pilot study: problems specific to educational multimedia projects

The pilot study was conducted in the scope of the problem analysis (chapter 4). The objective of the pilot study was to identify the most important problems that caused the relatively low level of maturity of the courseware development project process. A team of three independent researchers identified six project problem categories, which we call themes. The most appropriate instrument for addressing the themes appeared to be knowledge management (in the sense of 2.4) because of the nature of the needs for improvement. In section 7.4, we discuss the suitability of knowledge management as an instrument to improve process maturity.

Process issues: general or really educational multimedia specific?
When taking one step back, we may ask ourselves if the problem categories that were found are really specific to courseware development projects. Even though there may not be a definitive "yes" or "no" as answer to this question, we will argue here that the answer is both "yes" and "no", only seen from different angles when looking at these project problems.

Let us briefly recall the project problem categories that we identified: Project management, qualification of skills, communication, psychological issues, technological issues, and customer/user relationship issues. First of all, the answer to whether we are talking about educational multimedia-specific categories is "no", because when looking at the words themselves, there is really nothing educational multimedia specific about them. We have project management in all types of software development; the same goes for communication issues and the relationship with the customer. Psychological issues are usually not made explicit in traditional software projects (although there are exceptions, see [GIB1994]), but they shall surely occur; finally, the qualification of skills is also an issue that certainly applies to traditional software development.

The answer to whether we are talking about educational multimedia-specific categories only becomes "yes" when we look beyond the mere wording used for the categorization. When we dub "communication" as one of our project issue categories, we refer to the communication between disciplines that are as different as fire and water. We mean communication between an artist and an engineer on a product that is supposed to convey some new knowledge to a learner. This is a wholly different type of communication than that usually found in the development of traditional software. Similarly, when we talk about "Qualification of skills", we must take into account the various media that are involved: when looking at audio, for example, we could be talking about skills such as music composition and arrangement, music recording, voice-over recording, mixing, mastering, and editing in the final product. When we talk about "Customer-related issues" we refer to a product

that has an instructional goal, instead of aiming to allow a user to achieve some set of tasks more easily, as is often the case in traditional software development. For technical issues, we talk about video footage, animations, high-resolution photographs, music and speech tracks, in addition to programming languages, scripting tools, database management tools, and network issues, which are most often found in traditional software development. So yes, we do feel that the problem categories that we have identified do contain many educational multimedia-specific elements; they just do not surface at the level of the wording.

Why this particular categorization?
One might also argue that the categories of project problems found are rather arbitrary; many other categorizations would probably suffice as well. This is probably true for most of the categories, although some of our categories found are rather distinctive from the others, particularly the categories on technical issues and on project management. Categories such as communication and customer-related issues are more vague: communication is wrought in almost all other categories, and customer-related issues are often solely about communication or project management.

Still, the target for the pilot study was to identify a set of categories that would allow us to view the courseware development process problems from various angles. We required the categorization to be able to determine the relative weight of each of them and to determine changes in each of the categories as the knowledge management solution was deployed. In this sense, other candidate categorizations would probably have sufficed as well; however, our categorization was fully agreed upon by three independent researchers, which was a prerequisite for achieving the required amount of inter-reliability for analyzing content in project interviews.

Soft versus hard process issues
Although the creation of courseware seems a rather technical challenge at first, only one of the reported project problems addressed technological issues. All other categories were on the "soft" side of the software development spectrum. Although this seems rather strange for a process that eventually delivers a product that runs on a computer, even in traditional software development, practitioners and researchers now realize that technical difficulties are only a small part of all issues that must be managed. A defect network printer is usually not a project stopper, while a customer that continues to change the requirements is many times more bothersome. Recent software development methodologies such as D.S.D.M. explicitly take into account the importance of the "softer" issues with principles such as (1) teams that are empowered to make decisions on their own; (2) a no-blame project culture; (3) a strong emphasis on co-operation between all parties [STA1997]. We even feel that the "softer" issues become more important as the level of interactivity of the final product increases. Additional research would be required to test this presumption, but if true, this would explain the small amount of project experiences that were reported in section 6.2.

Recent research (e.g., [VDM1995A; ALB1996; BOY1997; LYN1993]) indicates that the role of the project manager in courseware projects is by far the most important because the employee in that role must be able to oversee and communicate with all the disciplines that are involved. Our study confirms these findings; moreover, we see that project experiences about the multidisciplinary project management become significantly more positive when an instrument for formal knowledge management, in the sense that we have defined it, is applied to this process. We conclude that our instrument for facilitating the knowledge needs of courseware development teams can facilitate

multidisciplinary project management in such a way that the project team experiences it as more positive.

7.2. Main experiment: the effectiveness of implemented solutions

Several thousand employees involved in projects in various departments of the entire range of ICT services that the company offers, now use the knowledge management solution that we have designed and implemented. In other words, the knowledge management solution that we have devised to address process related issues in educational multimedia systems, also seems to work for other types of software activities. We mention the Data Warehousing Performer (some 200 frequent users), the Desktop Migration Services Performer (over 500 active users), the SAP Performer (some 400 frequent users), and the Knowledge Management Performer (over 100 active users). At the company of study there is a general consensus that to employ knowledge management this way (that is, according to the KnowledgeWorks methodology), really does help in improving the maturity of the project process. We note here, however, that this observation is by no means backed up by scientific analysis.

We observed that the use of Performer is especially successful in the following usage patterns:

1. The formal definition of the way of working. The objectives matrix shows the activity flow of carrying out educational multimedia projects in a clear and concise way. Well-defined descriptions of each objective are directly accessible by clicking on the appropriate cell in the matrix. This has observably resulted in more clarity in the way of working of the entire project team.
2. The various ways of quickly accessing required knowledge. One of the main problems reported in the problem analysis was the constant "re-invention of the wheel", and the time it took people to find handy templates and best practices. Especially these searching times have been greatly reduced (by about a third, according to one impromptu questionnaire for the Data Warehousing Performer), and experts do not create new versions of documents that they can quickly find in the Knowledge Elements matrix.
3. The use of the individual estimates for constructing project bids. One of the most difficult points in constructing the project bids was the way in which estimates for the effort requirement of the various activities was done. As a large set of individual estimates is now quickly accessible for a variety of experience questions for all project roles, the quality of the project bids is now felt to have improved.

The main experiment has shown that the quality of project estimates is significantly better with "Yes Performer" projects, and project experiences are significantly more positive (19%) with those projects as well. However, we can still ask ourselves if the solution that we have successfully implemented really did help in improving the maturity of the project process of educational multimedia projects. We address this question by looking at the effect of the solution in a broader sense than just the statistical validation. When considering Figure 33, all themes but one show a shift in a positive vertical direction. When showing these graphical results to the experts involved, the majority agrees that the figure correctly visualizes the trend that they experience themselves. When asked to describe the advantages of Performer themselves, most experts remarked that (1) it allowed them to find knowledge and experience much faster, leaving time for other tasks, and (2) the fact that the entire way of working was now formally defined and agreed upon, made them feel they were working in a more professional environment. This was the goal of this research project

from the start: since we connect maturity to levels of professionalism, we aimed to improve the level of professionalism of the way of working of the project team, and thereby increasing the maturity of the project process.

The Capability Maturity Model revisited

Now that we have touched upon the issue of maturity, let us look again at the Capability Maturity Model. When looking at the CMM scale on levels 1, 2, and 3 (summarized in Table 26), because this is where our research project is situated, we see the following definitions and prerequisites for improvement (from [HUM1995]):

Table 26. *Summary of levels 1-3 of the Capability Maturity Model*

CMM level	Typical characteristics	Actions needed to reach higher level
3: Defined	Defined software process for both management and engineering activities is documented, standardized, and integrated into the development process. Projects use a documented and tailored version of the organization's process to develop/maintain software. Cost is reliable, though quality is still unpredictable.	Quantitative quality goals for software products through establishing and tracking process measurements and quantitative quality goals, plans, and cost/performance. Calculate cost of poor quality and compare to costs of achieving quality goals.
2: Repeatable	Basic project management processes are in place to track cost, schedule, and functionality. Necessary process discipline is in place to repeat earlier successes on projects with similar applications. Product quality is variable.	Org. standard process for developing and maintaining software, through documenting process standards and definitions, assigning process resources. Measurements of group coordination and project actions.
1: Ad hoc	Professionals driven from crisis to crisis by unplanned priorities and unmanaged change. Surprises cause unpredictable schedule, cost, and quality performance. Few processes are defined; success depends on individuals' heroic actions.	Process must become stable and repeatable through estimation, measurement, and planning (size, costs, risk, and schedule); requirements mgt; quality assurance; ability to manage sub-contracts

We shall not formally assess each key process area defined by the CMM for reaching a next higher level but instead employ Table 26 to assess the effectiveness of our solutions for the maturity of the project process. When considering levels 1 and 2, the problem analysis shows that the initial situation of the company of study was approximately on level 2 in terms of CMM. The PROMISE handbook could be regarded as a definition of a process that supported the tracking of cost, schedule, and functionality. Most of the disciplines were in place to repeat earlier successes, although the quality of products was not satisfactory controllable. Level 2 also defines the needs for reaching the next higher level. With Performer, we have indeed delivered an agreed-upon definition of a standard working process for developing courseware (although maintenance is still a neglected issue), and process standards have been formally made available through template knowledge elements. Moreover, we have set up a metric that allows for both quantitative and qualitative process variables to be measured. With the quantitative variables, we can say that we measure the "project actions", while using the qualitative "themes" we can say that we measure "group co-ordination". A formal assessment by a certified CMM assessor would have to determine if the process of the educational multimedia project team has indeed fully reached level 3. Except for maintenance and deployment of the courseware, this appears to be the case.

7.3. Validity of the results of the experiment

This section chiefly addresses the "rival explanations" that we mentioned in chapter 6. Some of the questions we ask ourselves are: (1) How do we argue that the effects that we have observed are indeed due to the solutions that we have implemented; (2) Are the results also applicable to, and repeatable with, similar environments outside the company of study; (3) Have we indeed observed the causal effect that the use of our knowledge management instrument results in a more mature process?

Are observed effects due to Performer?
First of all, one may argue that the improvements that we have observed may be due to entirely different reasons. Courseware experts, for example, may just have become more experienced in the course of the five-year time span of the research project. This would be a rival explanation for our observations. It would be hard to reject this rival explanation were it not for the attrition rate (personnel turnover) of the local unit of the company of study. The research project was carried out from 1996 until 2000; in some of these years, there has been a personnel turnover of more than 20 percent. At the end of the year 2000, only a fifth of the employees employed at the department in 1996 were still working at the same department. The average size of the department remained about the same because new staff was employed on all levels of seniority. The argument of improved seniority of existing employees becomes unlikely when taking this development into account. We do note that because of the objectives matrix of Performer, new employees were brought up-to-speed within our courseware projects much faster, because the way of working of the company of study was clear much quicker.

Applicability of the solutions outside the company of study
One might also argue that the fact that our company of study has experienced improvements in the maturity of the courseware development process is nice for Atos Origin, but does this also apply to similar companies? We have realized this issue from the start and have attempted to repeat our experiment at two similar companies in The Netherlands, which shall remain nameless. Both alternative companies were involved in the production of courseware from an IT-point of view, for similar types of customers. Both had a project-type approach and neither of them had a solution for formal knowledge management (in the sense that we used it) in place. In other words, these two candidate companies appeared to offer adequate empirical environments for repeating our experiment outside the company of study. In the startup phase at both alternative companies, though, we had to abandon the effort because our company of study considered it too much a risk to freely give away the structure of the knowledge management instrument developed in the course of this research project, even without the actual data. They felt that they had generated themselves a competitive advantage with Performer, the more so because various other organizational units were using Performer as well. Giving away the Performer concepts would potentially harm this competitive advantage too much. Since Atos Origin has acted as the financial sponsor of this research project, the research activities had to be restricted to the company of study only. Of course, for other companies this thesis provides a useful instrument for repeating the experiment. In chapter 8 we mention this point as one of the questions that remain to be answered.

Applicability of the solutions outside the boundaries
In chapter 3 we have defined the project boundaries of the projects that we analyzed. We may ask ourselves if the boundaries may be expanded to include larger courseware projects with a larger

budget and more functionality involved. Our company of study did not involve itself into courseware projects of such sizes (with a multi-million dollar budget, for example) because they felt that the department that was involved in courseware projects was not sufficiently large enough to tackle such a project. It would be an interesting research topic to try to apply the solutions and experiment of thesis to these kinds of projects. We expect more problems reported on all project problem categories, because of the increased number of people involved, with results in more complexity (the mythical man-month principle, see [BRO1995]).

Then we should also briefly consider the fast pace of technological developments. Is the experiment repeatable a few years from now? We have attempted to avoid as much as possible to depend on technology-driven constraints or assumptions. We have observed that by far most of the process problems experienced by courseware teams are of a non-technical nature. Only one of the six themes identified dealt with technological issues, and this particular theme was the least reported upon. This theme was also the only one that showed statistically significant more negative project experiences. We may ask ourselves if the ever-faster pace of new technologies, new development tools, and new technological means for communication causes this. This is an interesting research topic to pursue and will be recalled in chapter 8, with the recommendations for further research.

Finally, we note here that we observed that the knowledge management solution that we implemented for courseware projects at the company of study is adequately applicable to other types of IT services as well. Especially the Data Warehousing Performer, the SAP performer and the Desktop Migration Services Performer are used by a large number of experts in many departments throughout the company. It seems, therefore, that the educational multimedia-specific issues that caused process maturity problems can be solved by a general formal solution for knowledge management. Additional research is required to measure the effectiveness of a Performer-like solution for other types of software development.

Causality

Let us briefly recall the three prerequisites for determining a causal effect: (1) Time order. This implies analyzing data that is collected both *without* having the knowledge management solution in place, as well as having the knowledge management solution operational, in order to be able to determine the causal effect. We have certainly satisfied this condition. A second condition necessary for a causal relationship is called *association*: varying degrees of employing knowledge management should be associated with similar degrees of the expected effect. We have not been able to sufficiently test this prerequisite for causality. On the one hand, we may suggest more research to test this; on the other hand, it is hard to apply "more Performer" to an organizational environment. We can talk about "levels of formalism" of knowledge management, but it is unrealistic to talk of "more Performer" for a company of study. (Although not entirely impossible: for example, one could aim at facilitating the project manager even more in such a way that the person in this role can oversee all other disciplines even better.)

The third requirement for demonstrating a causal relationship involves the discussion of "*rival explanations*". These have been discussed earlier in this chapter; we have at least made credible that the rival explanations, such as increased levels of seniority of existing employees, are not likely to have influenced the experiment in a significant way.

7.4. Knowledge management as an instrument to improve process maturity

The main objective of this research project was to investigate ways to improve the maturity of the process of educational multimedia projects, and then to choose and implement the most appropriate of these. We chose to employ an instrument geared towards our working definition of knowledge management because the problem analysis of chapter 4 identified the lack of any structured form of knowledge management as one of the key problems to be addressed. We have conducted, as careful as possible given the empirical context, a main experiment to determine the effectiveness of the knowledge management instrument that we devised. Although the outcome of the experiment does not indicate a major breakthrough for the case for the power of knowledge management in general, it is surprising to find the lack of thorough quantitative evaluations of the use of knowledge management in process improvement. This is especially true for process improvement in courseware development.

Why has Performer become a success? What have been the key success factors? Answering these questions will be of value to other companies and professional institutes who aim to implement a formal solution for knowledge management. This thesis does not provide a statistically validated way of determining why Performer has become a success, but we can provide the reader with observations about why Performer seems to continue to work for an increased number of people.

First of all, it is important to note that a formal solution for our definition of knowledge management (which Performer is, with its defined way of working and storage of formal approved templates and best practices) cannot work without its counterpart, informal knowledge management. In chapter 5 we have discussed three levels of formality of knowledge management, in order of increasing formality level: (1) communities of practice, (2) networks of professionals, and (3) formal knowledge management. The company of study has each of these three in place. It is generally perceived that Performer clearly cannot work without colleagues finding each other in informal ways too, for example through networks of professionals with similar interests.

Secondly, even though this may be obvious to any reader, we feel that it cannot be over-emphasized: the commitment of the top-level management of the company. Their worry is one of the utilization of intellectual capital: the most valuable asset of an ICT services company consists of their employees, including their expertise. The main challenge lies in finding clever ways of implementing knowledge management to leverage this intellectual capital. Since there is no general prescriptive way of successfully implementing knowledge management in a project-type organization, it is generally very hard to get full-blown commitment from the top management: why should they if they have no guarantee of any positive effect? We should convince them by presenting quantitative figures that prove the actual added value of a well-prepared knowledge management implementation. This requires more formal evaluations of knowledge management implementation methods. On the other hand, there seems to be a circular problem: it is only when a formal budget is allocated on the highest level, that such a knowledge management implementation has any chance of succeeding. Those who are trying to convince those at the top-level management may use this thesis as argument for formal budget allocation. In addition, [BOO1997], [DAV1998] and [WEG1997] are valuable references to consult in this regard.

One of the main key success factors for the success of our Performer implementation was perceived to be the balanced implementation of the four aspects (1) Organization, (2) Culture, (3) Content, and (4) Infrastructure. The fact that merely installing a technical "tool" is not sufficient is now commonly accepted. For the "Organization" aspect, we have defined several roles with corresponding tasks and responsibilities, such as the content owner, the moderator, and the champion. In addition, we have set up procedures for the deployment of the solution: adding new knowledge, assessing the quality of existing knowledge, and granting access to the right part of the knowledge to the right employees. For the "Culture" aspect, we have set up an elaborate communication plan to inform people about the advantages of our approach to knowledge management; we have conducted knowledge-sharing sessions and hands-on workshops, and most important of all, we have convinced people that it is in their own interest to contribute to knowledge management. The culture aspect required the most budget of all, as much as 40%, more than we had expected. For the "Content" aspect, we made sure we had a minimum critical mass of knowledge and experiences already available by the time we formally operationalized our knowledge management solution. This was important because employees could immediately find what they were looking for, which made them more susceptive to contributing themselves. For the "Infrastructure" component, finally, we made sure that the objectives, the knowledge and experiences were available online regardless of geographic location, on the intranet. We claim that to ignore or neglect any one of these four factors quickly minimizes the chance of a successful knowledge management implementation.

Since it can be hard to convince employees that the sharing of knowledge is in the end more profitable than keeping one's expertise to oneself, employees must not be disappointed by the time they decide to "try it". We realized that we would have to provide for a low threshold for contributing and extracting knowledge and experiences. This caused a dilemma because we clearly did not want our online knowledge and experience to become available to unwanted persons. By formally restricting access to the corporate intranet, but providing free access to Performer once access to the intranet is gained, we hoped to provide for this low threshold. We did pose one restriction: physically downloading knowledge elements from any given Performer is allowed only if that person is actively involved in that particular field. In addition, a formal online "code of conduct" had to be signed in order to gain download permission. In practice, this has turned out to work in a satisfactory way.

A final prerequisite for successful knowledge management that we mention here is a formal deployment process for maintenance on both a functional and technical level. The technical level seems obvious, although this, too, is often neglected. A growing number of employees contributing to a growing collection of knowledge sets more stringent requirements on the accessibility and performance of the network. On a functional level of deployment, things are even more complex. By formally defining a user group and a change control board, including procedures for requesting functional changes, we can control the effectiveness of the knowledge management solution in such a way that the concept of formal knowledge management remains focused.

We conclude by mentioning that a knowledge management implementation should always be targeted towards improving the maturity of the process involved; a framework such as the Capability Maturity Model offers valuable advice for targeting the knowledge management implementation efforts.

Summarizing, we once again see that the prerequisites for successful knowledge management on an operational level have very little to do with technological requirements. We are convinced that knowledge management implementation failures are chiefly due to the aspects described in this section, in addition to a certain required minimum level of process maturity: an organization that already operates on level 2 of the CMM model is more likely to succeed in setting up knowledge management than a similar organization that operates on level 1.

8. Conclusions

8.1. Project process improvement in courseware projects

In this research project, we have investigated the effectiveness of a formal knowledge management approach to improve the process maturity of courseware projects. The development of courseware is a special, generally more complex case of traditional software development because of its multidisciplinary and multimedia nature. The multidisciplinary nature makes it complex because the characteristics of the various disciplines involved are so fundamentally different (e.g., the "artist" and the "engineer") that this can easily cause communication problems. The use of multimedia makes it complex because the use of multimedia requires tools that are often not robust (due to their innovative nature), media interaction is complex to design, and because customer's expectations are hard to manage.

Courseware is developed by a project team, often including the following disciplines: project management, instructional design, interaction design, graphics design, programming, audio realization, and video realization. At the company of study of our research project, one department is involved in producing courseware. While carrying out their projects, they experience a range of problems related to the level of professionalism, or maturity, of their way of working. An initial set of project interviews resulted in the identification of the following project problem categories: (1) Project management, (2) qualification of skills, (3) communication, (4) technological issues, (5) psychological issues, and (6) customer/user relationship issues. The department of the company of study experienced the following needs for improvement:

1. Easy access to all available knowledge and expertise of all employees;
2. Project experiences easily available for later re-use;
3. Standardization of the educational multimedia project process and support when working this way;
4. Availability of consensus estimates, for use in project bids.

These needs are chiefly of a knowledge management nature. We have therefore constructed a formal knowledge management instrument to address them. The framework used for designing our knowledge management instrument, called Performer, is the Capability Maturity Model; the steps required for moving to maturity level 3 have been leading for the design of the functionality. The framework used for implementing Performer into the organization is called KnowledgeWorks, a knowledge management methodology designed by the company of study. KnowledgeWorks holds that a knowledge management implementation can only be successful if there is a careful balance between the four aspects (1) Organization, (2) Culture, (3) Content, and (4) Infrastructure (more about this in section 8.3). Performer formally defines the way of carrying out a courseware project in terms of phases, roles, and objectives for each phase-role combination; templates and best practices, called knowledge elements, can be connected to their proper objectives. The company of study considers our implementation of Performer a success. In addition to the Education Performer for courseware projects, an additional fifteen Performers are now operational on a variety of services, including Data Warehousing and Enterprise Resource Planning.

Educational multimedia experts now work according to the agreed-upon way of working visualized in the Performer objectives matrix, and they can find required knowledge much quicker than

before. New employees are brought up to speed much quicker than before. Project bids can now become of a higher quality with the individual estimates that we have recorded. A formal deployment process is in place to guard the concept of the functionality of Performer. A metric has been constructed to record measurements about the project actions, e.g. calendar time, budget spent, and working hours recorded; in addition, project experiences about the six project problem categories can be recorded and analyzed. With these steps, we have addressed the required actions for reaching maturity level 3 on the Capability Maturity Model scale.

8.2. Research hypothesis

The research hypothesis states that the use of our knowledge management solution in educational multimedia projects within given boundary conditions, in a project oriented IT solutions provider, results in: (*a*) a better estimation of basic quantitative project variables of projects and (*b*) in statistically significant fewer occurrences of negative process experiences, and significantly more occurrences of positive process experiences. To test this hypothesis, we have set up a main experiment with two parts, one to test sub (*a*) of the hypothesis, and the second to test sub (*b*) of the hypothesis.

The first part of the main experiment, described in section 6.2, addresses the quality of the estimates of the quantitative project variables (1) calendar time, (2) budget, (3) working hours, and (4) product size. For courseware products, calculating product size is not a trivial matter because of the various media involved. We calculate the size of a courseware product using our own dedicated formula described in appendix E; we express the product size in the SIUN unit. We analyzed both the estimates and the actual values of these quantitative project variables for ten courseware projects, five of which did not use Performer. We call the average difference between the estimates and actual values of each quantitative variable the discrepancy factor for that variable. We have conducted the nonparametric Mann-Whitney test to test sub (*a*) of the hypothesis. We decided to reject the hypothesis if the aggregated data of the quantitative project variables, for each of the two Performer usage cases do not show a significant improvement. With the Mann-Whitney test and an unreliability threshold $\alpha = 0.05$, we find that we cannot reject sub (*a*) of our hypothesis, and we conclude that the discrepancy factor becomes indeed significantly smaller, in other words that project estimates do improve for projects were Performer was used.

The second part of the experiment, described in section 6.3, addresses the nature of the project experiences that are reported for the same ten courseware projects. Three independent researchers analyzed the interviews that we conducted with the aim of capturing the process problems of these projects. Each interview answer was assigned at most two of the six project problem categories ("themes"), described in section 8.1, with an assessment in the range of {-2,-1,1,2} where –2 is very negative and +2 is very positive. We once again conducted the Mann-Whitney test (with the same unreliability factor $\alpha = 0.05$) to determine if, for projects that did use Performer, there are significantly more positive project experiences for project and significantly less negative project experiences. The inter-reliability between the three researchers was found to just exceed the required minimum of 80%. We decided to reject the hypothesis if the aggregated data of all themes, for each of the two Performer usage cases, do not show a significant improvement. The results indicate that overall (with aggregated themes), the project experiences are indeed significantly more

positive. Especially with project management and customer/relationship issues, we found that project experiences were much higher. We conclude that we cannot reject sub (*b*) of the hypothesis.

One notable observation is that we probably require a larger number of projects to analyze, to determine the causal effect of our knowledge management solution with higher reliability thresholds. Still, the company of study regards the implementation of our knowledge management solution as a success, as an increasing number of employees active in a variety of services now use Performer for their project tasks. We can therefore conclude that to employ a formal knowledge management solution such as Performer, implemented using the KnowledgeWorks methodology results in significantly more positive project experiences and significantly better quantitative project variable estimates.

8.3. Successfully implementing knowledge management

What are key success factors for a knowledge management implementation? Understanding this will be of value to other companies who aim to implement a formal solution for knowledge management. We present here some observations on an operational level, based on section 7.4, about why Performer seems to continue to be of value for an increased number of employees.

First of all, we distinguish between three levels of formalism in knowledge management, namely (1) communities of interest, (2) communities of practice (networks of professionals), and (3) communities for purpose (formal knowledge management). All levels of formalism are required for a successful knowledge management implementation, with knowledge management in the sense that we have defined it in section 2.4. In case (1), employees with similar interests start informal knowledge sharing. In case (2), these employees organize themselves with some formal budget allocation, and hold regular meetings and online discussions. In case (3), the way of working is formally agreed upon and templates, best practices and consensus estimates are collected and made available to all who are interested in that area.

Secondly, a very important requirement is the commitment of the top-level management of the company. Their worry is one of the utilization of intellectual capital: the most valuable asset of an ICT services company consists of their employees, including their expertise. The main challenge lies in finding clever ways of implementing knowledge management to leverage this intellectual capital. Since there is no general prescriptive way of successfully implementing knowledge management in a project-type organization, it is generally very hard to get full-blown commitment from the top management: why should they if they have no guarantee of any positive effect? We should convince them by presenting quantitative figures that prove the actual added value of a well-prepared knowledge management implementation. This thesis may help in that regard.

Thirdly, according to the KnowledgeWorks methodology for knowledge management, there should be a balanced implementation of the four aspects (1) Organization, (2) Culture, (3) Content, and (4) Infrastructure. The fact that the merely installing of a technical "tool" is not sufficient is commonly accepted by now. Organization is about defining procedures with their roles, tasks and responsibilities. We require at least a champion, a content owner and a moderator. Culture is about convincing employees that sharing their knowledge is in the end much more advantageous than keeping the knowledge to themselves. Content is about creating an initial critical mass of valuable knowledge and expertise by the time the implementation is finished. Infrastructure is about having

the knowledge and experience available online always and anywhere. Neglecting any four of these aspects quickly reduces the chance of success. In our experience, the culture aspect requires some 40% of the total implementation budget, while the three other aspects each require some 20%.

Employees must not be disappointed by the time they are finally persuaded to "try it". We should provide for a low threshold for contributing and extracting knowledge and experiences. This can cause a dilemma because we clearly do not want online knowledge and experience to become available to unwanted persons. By formally restricting access to the corporate intranet, but providing free access to the knowledge management solution once access to the intranet is gained, we can provide some safety level on the one hand, and a low threshold for contributing and extracting knowledge on the other hand.

Finally, we mention the importance of a formal deployment process for maintenance on both a functional and technical level. On the technical level, a growing number of employees contributing to a growing collection of knowledge sets more stringent requirements on the accessibility and performance of the network. On a functional level of deployment, things are even more complex. By formally defining a user group and a change control board, including procedures for requesting functional changes, the effectiveness of the knowledge management solution can be controlled.

8.4. Recommendations for further research

We have observed a negative tendency in project experiences about technological issues. We may ask ourselves if the increasing rapid pace of new technologies, as discussed in section 7.3, can cause this. An interesting experiment would be to conduct measurements on project experiences for a wide range of project types and then try to determine if experiences become more negative in the scope of five years. Other recommendations for additional research are:

- We have addressed the effectiveness of a formal solution for knowledge management for courseware development projects. We should investigate the other levels of formality of knowledge management as well. Especially the effectiveness of networks of professionals in relationship to formal knowledge management is not well understood and could be investigated further.
- We should investigate the applicability of our research to a university environment as opposed to a commercial company environment. More generally, we should test our hypothesis in environments where courseware is developed not strictly in teams.
- We recommend repeating the experiment at other companies of study to determine if the results are generally applicable to our type of organizations. The company of study was an ICT services company. However, companies that are more media-oriented, and in which IT services play a less significant part deliver a large percentage of available courseware. It is not clear whether or not the needs for improvement are similar, and whether or not a similar knowledge management solution implementation would work.
- We recommend repeating the experiment at project-type organizations that are involved in areas other than educational multimedia. We observed that our knowledge management instrument appeared to work for other departments at the company of study as well. Since many of these departments feel the need for improved process maturity too, we could investigate how this thesis applies to those types of services.

- We could conduct a more elaborate evaluation of the process maturity improvement that we have observed, connected to the formal key process areas of the Capability Maturity Model.
- Our research on employing knowledge management has been mostly on an operational level, i.e., a hands-on solution that employees in project teams can use. In order to get highest-level management commitment, we could look into quantitative ways of determining the added value of knowledge management solutions for the entire company, on a business-level point of view. In general, how the use of knowledge management translates to utilizing the intellectual capital of organizations is not well understood.
- We could investigate the possibilities of employing the possibilities of applying formal knowledge management such as Performer for initial student training to develop courseware. We have now used Performer to help experts in the field, but it may also help for students to learn about developing educational multimedia. In an educational setting, we could experiment with various degrees of applying knowledge management to assess the effect of the level of knowledge management for the learning curve. We could measure groups of students that do not use Performer versus groups of students that do use Performer.
- We should investigate just what the effect of the use of our instrument is on the role of the project manager – in what way are the expertise and level of professionalism of that role influenced?
- A solution for formal knowledge management such as Performer requires a defined way of working (modeled in the objectives matrix). We may ask ourselves the question if we can use a Performer-like solution for processes that are harder to define, such as the dissemination and use of knowledge about the usage of ICT in secondary schools.
- An interesting question is whether or not our knowledge management approach is suitable for attempting to reach even higher levels on the CMM scale. We should investigate whether or not we can use more of the same sort of solution, or if an entirely different method is required.

With this list and the results of our research, we hope to have contributed to the young but growing body of knowledge about using knowledge management for software process improvement.

References

[AFT1983] Aft, L.S. (1983), *"Productivity, Measurement, and Improvement"*. Reston Publishing Company, Viriginia, U.S.A.

[ALB1996] Alber, A.F. (1996), *"Multimedia: a management perspective"*. Wadsworth Publishing Company, California, USA. ISBN 0-534-21312-X

[ALB1997] Albert, S., and K. Bradley (1997), "*Managing knowledge*". Cambridge University Press, Cambridge, Mass., U.S.A.

[ALL1997] Allee, V. (1997): "*The knowledge evolution: Expanding Organizational Intelligence*". Butterworth-Heinemann, Boston, U.S.A.

[AND1990] Anderson, C.J., and M.D. Veljkov (1990), *"Creating interactive Multimedia"*. Scott, Foresmann, Glenview, Illinois, U.S.A.

[BAA1995] Baarda, D.B., M.P.M. de Goede, and J. Teunissen (1995), *"Kwalitatief onderzoek. Praktische handleiding voor het opzetten en uitvoeren van kwalitatief onderzoek"*. Stenfert Kroese, Leiden, Netherlands (in Dutch).

[BLO1983] Block, R. (1983), *"The politics of projects"*. Yourdon Press computing series, New York, U.S.A.

[BOE1988] Boehm, B.W. (1988), "A spiral model for software development and enhancement" In: *IEEE Computer*, pp 61-72.

[BOO1997] Boone, P.F. (1997), "*Managing Intracorporate Knowledge Sharing*": Ph.D. thesis Erasmus University, Rotterdam. Eubron Publishers. ISBN 90-5166-577-6.

[BOY1997] Boyle, T. (1997), *"Design for multimedia learning"*. Prentice-Hall Europe, U.K. ISBN 0-13-242215-8.

[BRO1995] Brooks, F.P. jr. (1995), *"The mythical man-month. Essays on software engineering"*. Addison-Wesley, Reading, Massachussetts, U.S.A.

[CAR1990] Card, D.N., and R.L. Glass (1990), *"Measuring software design quality"*. Prentice-Hall, New Jersey, U.S.A. ISBN 0-13-568593-1.

[CER1990] Cerri, S.A., and J. Whiting (Eds.) (1990), "Learning technology in the European Communities". *Proceedings of the DELTA conference on research and development.* Kluwer Academic publishers, London, U.K.

[CHA1990] Chambers, J.A., and J.W. Sprecher (1990), "Computer Assisted Instruction: current trends and critical issues". *Communications of the ACM*, Vol. 23, no. 6, pp. 332-342.

[CLA1994] Clarke, A. (1994), *"Human factors guidelines for multimedia. European Commission RACE ISSUE project 1065"*. HUSAT Research Institute, Loughborough institute of technology, United Kingdom.

[CON1986] Conte, S.D., H.E. Dunsmore, and V.Y. Shen. (1986), *"Software Engineering Metrics and Models"*. Benjamin/Cummings, California, U.S.A.

[DAV1998] Davenport, T., and L. Prusak (1998), *"Working knowledge: How Organizations Manage What They Know"*. HBS Press, Boston, MA, USA. ISBN 0-8758-4655-6.

[DEM1987] DeMarco, T., and T. Lister (1987), *"Peopleware: productive projects and teams"*. Dorset House Publishing co., New York, U.S.A.

[DIX1993] Dix, A. and J Finlay. (1993), "*Human-Computer Interaction*". Prentice-Hall Europe, U.K. ISBN 0-13-458266-7.

[DUN1996] Duncan, W.R. (ed.) (1996), *"A guide to the project management body of knowledge"*. By the PMI standards committee. Project Management Institute, Upper Darby, PA, USA. ISBN 1-88041-013-3.

[EBE1988] Eberts., R.E., and J.F. Brock (1988), "Computer Based Instruction". In: *Handbook of Human-Computer Interaction*, Elsevier Science Publishers B.V. (North Holland), pp. 599-627.

[ENG1996] England, E., and A. Finney (1996), *"Managing multimedia"*. Addison-Wesley, Reading, Massachussetts, U.S.A.

[FAL1995] Fallenstein Hellman, M., and W.R. James (1995), *"The multimedia casebook"*. Van Nostrand Reinhold, New York, U.S.A. ISBN 0-442-01819-3.

[FIS1988] Fisher, R., and S. Brown (1988) , *"Getting together – Building Relationships as we negotiate"*. Penguin Books, New York, U.S.A.

[FOG1999] Fogg, B.J. (ed.) (1999), "Persuasive technologies". In: *Communications of the ACM*, May 1999, Vol.42, no. 5, pp. 7-16.

[GAR1996] Garmus, D., and D. Herron (1996) , *"Measuring the software process: A practical guide to functional measurements"*. Prentice Hall, London, United Kingdom.

[GER1987] Gery, G. (1987) , *"Making CBT happen"*. Weingarten Publications, Boston, U.S.A. ISBN 0-9617968-0-4.

[GIB1994] Gibbs, W. W. (1994), "Software's Chronic Crisis," In: *Scientific American*, September, pp. 86-95.

[GIL1988] Gilb, T. (1988) , *"Principles of Software Engineering Management"*. Addison-Wesley, ISBN 0-201-19246-2.

[GLA1983] Glass, R.L.. (1983), *"Computing Catastrophes"*. Computing Trends, PA, U.S.A. ISBN 0-686-35783-3.

[GRE1996] Greene, J. (1996), *"Management measures for excellence: the software control office"*. Published by Quantitative Software Management ltd., www.qsm.com.

[HAA2000] Haas, R., W. Aulbur, A. Statuz, (2000), "A web based system for distributing and sharing knowledge". In: Proceedings of *Euromedia 2000 conference, Antwerp*. SCS, Ghent, Belgium. Pp. 69-73.

[HAR1988] Hartemink, F.J.A. (ed.) (1988), "*Handbook for the design of educational software*". Version 3.0. PMI series 12, C.O.I., Enschede, Netherlands (in Dutch).

[HAT1995] Hatchuel, A., and B. Weil, "*Experts in organizations: a knowledge-based perspective on organizational change*". De Gruyter Publishers, Berlin, Germany. ISBN 3-11-014119-1 (translated by L. Libbrecht).

[HEB1977] Hebenstreit, J. (1977), "New trends and related problems in Computer-Based education". In: B. Gilchrist (ed.), *1977 IFIP Conference proceedings*, North-Holland, pp. 201-208.

[HEN1991] Henderson, A. (1991), "A development perspective on interface design and theory". In: J.M. Carroll (ed.), *Designing Interaction*, Cambridge University Press, pp. 254-268.

[HUM1995] Humphrey, W.S. (1995), "*A discipline for software engineering*". Addison-Wesley, Reading, Massachussetts, U.S.A.

[HUM1997] Humphrey, W.S. (1997), *"Managing the software process"*. Addison-Wesley, Reading, Massachussetts, U.S.A.

[HSI1993] Hsia, P. (1993), "Learning to Put Lessons Into Practice". *IEEE Software*. September 1993, pp. 14-17.

[HUN1991] Huckin, T.N., and L.A. Olsen (1991), "*Technical writing and professional communication for nonnative speakers of English*". Second edition. McGraw-Hill, New York.

[JAC1996] Jacobs, D. (1996), *"Het kennisoffensief. Slim concurreren in de kenniseconomie"*. Samsom Bedrijfsinformatie, Alphen a/d Rijn, Netherlands (in Dutch).

[JAC1997] Jacques, R. (1997), *"Engagement as a design concept for multimedia"*. Ph.D. thesis, Southbank University, London, U.K.

[JOH2000] Johnson, D.L., and J.G. Brodman (2000), "Applying CMM project planning practices to diverse environments" *IEEE Software, Vol. 17 nr 4*, special issue on process diversity, pp. 40-47.

[JON1997] Jones, C. (1997), *"Applied software measurement: assuring productivity and quality"*. Second edition. McGraw-Hill, New York, U.S.A.

[JON1994] Jong, Ton de, and L. Sarti (eds.) (1994), *"Design and Production of Multimedia and Simulation-based Learning Material"*. Kluwer, Dordrecht, The Netherlands. ISBN 0-7923-3020-X.

[KEE2000] Keeni, G. (2000), "The evolution of quality processes at Tata Consultancy Services". In: *IEEE Software, Vol. 17 nr 4*, special issue on process diversity, pp. 79-88.

[LAI1994] Lai, R. (1994), "*The move to mature processes*". IEEE Software publication, July 1994, pp. 14-17.

[LAU1993] Laurel, B. (1993), *"Computers as theatre"*. Addison-Wesley, Reading, Massachussetts, U.S.A.

[LEE1994] Leeuw, A.C.J. de (1994), "*Besturen van veranderingsprocessen*". Van Gorcum & Comp, Assen, Netherlands. ISBN 90-232-2781-6.

[LEO1995] Leonard-Barton, D. (1995), "*Wellsprings of knowledge*". HBS Press, Boston, MA, U.S.A.

[LU2000] Lu, S.C.J., and J. Cai (2000), "STARS: a socio-technical framework for integrating design knowledge over the Internet". In: *KIEEE Internet Computing, special issue on knowledge networking*, Sept-Oct 2000, pp. 54–62.

[LYN1993] Lynch, P.J. (1993), "Developing Multimedia: considerations for a successful project". In: *Higher Education Product Companion 3 (3),* U.S.A.

[MON1981] Monaco, J. (1981), *"How to read a film. The art, technology, language, history and theory of film and media"*. Oxford University Press, New York, U.S.A.

[MOO1987] Moonen, J. (1987), "Educational Software Development: The pedagogical design". In: Tj. Plomp, K. van Deursen, J. Moonen (Eds.), *CAL for Europe*, pp 51-64. North Holland Publishers, Amsterdam.

[MOR1996] Moran, T.P., and J.M. Carroll (1996), *"Design Rationale: Concept, techniques, and use"* Lawrence Erlbaum associates, Mahwah, New Jersey, U.S.A.

[MUS1987] Musa, J.D., I. Iannino, And K. Okumoto (1987), "*Software reliability – Measurement, Prediction, Application*". McGraw-Hill, New York, U.S.A.

[MYE1994] Myers, B.A. (1994), "Challenges of HCI Design and Implementation", In: *Interactions*, Vol. 1, No 1, pp. 73-83.

[NEJ1995] Nejmeh, B. A. (1995). "Process Cost and Value Analysis". In: *Communications of the ACM*, Vol. 38, No. 6, June 1995.

[NON1995] Nonaka, I., and H. Taceuchi (1995), "*The knowledge-creating company*". Oxford Press, New York, U.S.A

[NOR1988] Norman, D. (1988), *"The psychology of everyday things"*. Basic books, inc., New York, U.S.A.

[ODE1998] O'Dell, C.S., and C. Grayson (1998), *"If only we knew what we know: the transfer of internal knowledge and best practices"*. Free Press, U.S.A.

[OMA1990] Oman, P.W., and T.G. Lewis (1990), *"Milestones in Software Evolution"*. IEEE Computer Society Press. ISBN 0-8186-9033-X.

[OPP1992] Oppenheim, A.N. (1992), *"Questionnaire design, interviewing and attitude measurement"*. Pinter publishers, 1992.

[PFL1996] Pfleeger, S.L. (1996), "Integrating Process and Measurement". In: A. Melton (ed.), *"Software measurement"*. International Thomson Computer Press, London, United Kingdom, pp. 53-74.

[PRE1994] Preece, J. (ed) (1994), *"Human-Computer interaction"*. Addison-Wesley, Reading, Massachussetts, U.S.A.

[PRU1994] Prusak, L. (1994), *"Knowledge in organizations"*. Butterworth-Heinemann, Massachussetts, U.S.A. ISBN 0-7506-9718-0

[PUT1996] Putnam, L.H., and W. Myers (1996), *"Controlling software development"*. IEEE Computer Society Press, Los Alamitos, California, U.S.A.

[REI1996] Reijnders, E. (1997), "Interne Communicatie". Van Gorcum, Assen, Netherlands (in Dutch). ISBN:

[RIF1998] Riffe, D., S. Lacy, and F.G. Fico (1998), *"Analyzing Media Messages. Using quantitative Content Analysis in research"*. Lawrence Erlbaum Associates, Mahwah, New Jersey, U.S.A.

[ROB1994] Robillard, P. N., J. Mayrand, and J. Drouin. (1994), Process Self-Assessment in an Educational Context. In J. L.Dìaz-Herrera (ed.), *Software Engineering Education: 7th SEI CSEE Conference.* New York: Springer-Verlag. p. 211-225.

[ROU1992] Rouse, W.B. (1992), "*Strategies for innovation: Creating successful products, systems, and organizations*". John Wiley, New York, U.S.A.

[RUG1997] Ruggles, R.L. (ed.) (1997), "*Knowledge Management tools*". A collection of articles from journals and books published between 1964 and 1995. Butterworth-Heinemann, Massachusetts, U.S.A. ISBN 0-7506-9849-7

[RUS1983] Rushby, N.J. (ed.) (1983), "*Computer Based Learning. State of the Art Report 11:4*" Pergamon, Maidenhead, U.K.,

[SAN1998] Sanders, A.F. (1998), *"Elements of human performance: reaction processes and attention in human skill"*. Lawrence Erlbaum associates and publishers, New Jersey, U.S.A. ISBN 0-8058-2051-5.

[SOF1995] Software productivity consortium (1995), *"The software measurement guidebook"*. International Thomson Computer press, U.S.A.

[SHN1992] Shneiderman, B. (1992), *"Developing the user interface: strategies for effective human-computer interaction"*. Addison-Wesley Publishing Company, U.S.A. ISBN 0-201-57286-9.

[SOL1988] Sol, H.G. (1988): "Information Systems Development: a problem solving approach". *INTEC*, Atlanta, Georgia, U.S.A.

[SPR2000] Sprent, P., and N.C. Smeeton (2000): "*Applied nonparametric statistical methods*". Chapman and Hall / CRC, U.S.A. ISBN 1-58488-145-3.

[STA1997] Stapleton, J. (1997): "*Dynamic Systems Development Methodology: working with DSDM.*" Addison-Wesley Publishing Company, U.S.A. ISBN 0-2011-7889-3.

[STE1986] Stevens, A.L. (1986): "The next generation of AI-Based Teaching Systems". *Machine-Mediated Learning*, Vol 1, No. 4, pp. 313-326.

[STE1985] Sternberg, R. (1985): "*How to complete and survive a doctoral dissertation*". St. Martin's Publishers, New York, U.S.A.

[SZE2000] Szentiványi, G. (2000) : "*Open, reusable and distributed components for multimedia information management: Design, implementation and integration*". Ph.D. thesis, Delft University of Technology, The Netherlands.

[TJO1991] Tjosvold, D. (1991), *"Team organization: an enduring competitive advantage"*. Wiley, Chichester, United Kingdom.

[VAA1994] Van Aalst, J.W., T.T. Carey, and D.L. McKerlie: "Design Space Analysis as a training wheels for user interface design*"*. In: *ACM CHI'94 Proceedings*, Boston, Mass., U.S.A.

[VAA1996] Van Aalst, J.W., and C.A.P.G. van der Mast (1996), "Analysis of problems during multimedia development projects". In: *EuroMedia Conference Proceedings*, London, U.K., 1996. Published by S.C.S., Ghent, Belgium, p. 78-82.

[VAA1997] Van Aalst, J.W., and F. de Haas (1997), "Kennismanagement concreet gemaakt." *In: Multimedia management*, issue October 1997, Kluwer, Netherlands (in Dutch).

[VAA1998A] Van Aalst, J.W., and J. Vader (1998), "Between concept and product: problems experienced by multimedia project teams". *In: Trends in Communication*, special issue, 1998, pp. 9-20. Boom Publishers, the Netherlands.

[VAA1998B] Van Aalst, J.W., and C.A.P.G. van der Mast (1998), "*Media experts share their expertise and project experience*". In: *EuroMedia Conference Proceedings*, Leicester, U.K., 1998. Published by S.C.S., Ghent, Belgium.

[VAA1998C] Van Aalst, J.W., and C.A.P.G. van der Mast (1998), "Creating the multimedia experience database." in: A. Sutcliffe, J. Ziegler, P. Johnson (Eds.) *Designing Effective and Usable Multimedia Systems, Proceedings of the IFIP Working Group 13.2 Conference*, Stuttgart, Germany, September 1998, Kluwer Academic Publishers, pp. 117-129.

[VAA2000A] Van Aalst, J.W., and C.A.P.G. van der Mast (2000), "Consistently measuring the size of a multimedia application" In: *Proceedings Euromedia 2000 conference,* Society for Computer Simulation (SCS) *,* pp. 181-187

[VAA2000B] Van Aalst, J.W., and C.A.P.G. van der Mast (2000), "Improving the CBT development process through knowledge sharing: a case study" In: *Interactive Learning Environments*, vol. ?, 2001.

[VBE1992A] Van Beckum (1992), "Management issues for large scale multimedia CBT projects" In: *Proceedings ETTE92 Conference, by J.Fricker (ed.),* pp. 181-187.

[VBE1992B] Van Beckum (1992), "Multimedia for flexibility in training and education: experiences from large scale multimedia projects". In: *TIME Europe 1992 Conference Proceedings*. EPCO International, Eindhoven, the Netherlands.

[VDM1983] Van der Mast, C.A.P.G. (1983), "Computer-based learning" In: *Computer-based learning*, State of the Art report 11:4, N.J. Rushby (Ed.) Pergamon, Maidenhead, U.K., 1983, pp. 111-124.

[VDM1991] Van der Mast, C.A.P.G. (1991), "Trends in Computer Assisted Instruction" In: *COO, State of the art*, Hogeschool voor Economische Studies, Rotterdam, The Netherlands. In Dutch.

[VDM1992] Van der Mast, C.A.P.G., and J. Rantanen (1992), "Next Generation Authoring Systems: integration of multiple methodologies and tools" In: S.A. Cerri & J. Whiting (Eds.): *Learning Technology in the European Communities*, Kluwer Academic Publishers, Dordrecht, Netherlands, pp. 519-533.

[VDM1995A] Van der Mast, C.A.P.G. (1995), "Developing Educational Software: Integrating disciplines and media". Ph.D. Dissertation, Delft University of Technology. ISBN 90-9007776-6.

[VDM1995B] Van der Mast, C.A.P.G. (1995), "Professional Development of Multimedia Courseware". In: *Machine-Mediated Learning*, vol. 5 (3 & 4), pp. 269-292.

[VAU1994] Vaughan, T. (1994), *"Multimedia: Making it work"*. Osborne McGraw-Hill, Berkeley, California, U.S.A. ISBN 0-07-882035-9.

[VER1994] Verreck, W.A., and H.G. Weges (1994), "Towards a Common Training Architecture for Flexible Distance Learning: a Description and Definition". *In: Design and Production of Multimedia and Simulation-based Learning Material.* Kluwer Academic publishers*, 1994, pp. 219-243*. ISBN 0-7923-3020-X

[WEG1997] Weggeman, M. (1997), "*Kennismanagement: Inrichting en besturing van kennisintensieve organisaties*"*:* Scriptum books, Netherlands. (In Dutch)

[WEG2000] Weggeman, M. (2000), "*Kennismanagement: de praktijk*"*:* Scriptum books, Netherlands. ISBN 90-5594-180-8 (In Dutch)

[WER1994] Werth, L. (1994), "An Adventure in Software Process Improvement". In J. L.Diaz-Herrera (ed.), (1994) *Software Engineering Education: 7th SEI CSEE Conference*. New York: Springer-Verlag. pp. 191-210.

[WII1995] Wiig, K.M. (1995), "*Knowledge management methods: practical approaches to managing knowledge*". Schema Press, Arlington, TX, USA. ISBN 0-9638925-2-5

[WIN1996] Winograd, T. (ed.) (1996), *"Bringing design to software"*. ACM Press books and Addison-Wesley, U.S.A.

[WIJ1994] Wijnen, G., W. Renes and P. Storm (1994), *"Projectmatig werken"*. Het Spectrum publishers / Marka, The Netherlands. ISBN 90-274-4489-7 (In Dutch).

[YEH1993] Yeh, Hsiang-Tao. (1993), *"Software Process Quality"*. McGraw-Hill, New York, U.S.A. ISBN 0-07-072272-2.

Summary

Of the Ph.D. thesis of Jan-Willem van Aalst, entitled "Knowledge Management in Courseware Development", published in April 2001.

Research setting
Multimedia is increasingly used as an integral part of educational systems. The multimedia factor is often partly responsible for the engagement of the user towards the educational product; additionally, multimedia can also substantially improve the learning effect of the user. Educational multimedia systems consist of hardware, software, procedures, data, and people. The courseware, then, consists of the hardware, the software, and the data. The people who are involved in courseware projects can be categorized as users, customers, and multidisciplinary project teams that produce the courseware product. Customer organizations are often large companies in the industry and (financial) services sectors. The project organizations, on the other hand, are often smaller service providers; they usually consist of small multidisciplinary teams who work according to a set of defined roles, tasks, procedures and responsibilities, and optionally a defined way of working. In these teams, disciplines such as project management, didactical design, functional design, technical design, interaction design, graphical design, programming, and video and audio production are found.

In development of courseware there are product issues and process issues to address. The quality of a courseware product is partly dependent on the level of maturity of the development process. Our research focuses on the process issues of courseware development; more specifically, we examine the maturity of the courseware development process, with a particular emphasis on the multidisciplinary and multiple media aspects. To this end, we choose a working definition for the term maturity, in order to make it measurable. We choose the framework of the Capability Maturity Model (CMM) because it is now accepted and widely in use in general software process improvement. In other words, our research aims to investigate ways in which the process of developing courseware can be more mature in terms of the CMM. We follow the interpretive research approach: we observe the maturity of the courseware development process in an empirical environment, and then in an inductive way propose a solution that we implement in the courseware development process. We then measure the effectiveness of our solution. The research hypothesis holds that the availability of an instrument to facilitate courseware development teams in their knowledge needs results in better project estimates, more positive and less negative project experiences on a variety of project themes. To this end, we choose a working definition for the fuzzy terms knowledge and knowledge management (defined in section 2.4).

To identify the needs for improvement in the courseware development process, we conduct a problem analysis at a commercial ICT services company, our company of study. In a formal pilot study, we carried out a set of sixty structured interviews. The results indicate that the reported needs for improvement fit well in the five-level maturity scale of the CMM. Characteristics on maturity level 2, the *repeatable* level, include the lack of formal procedures for controlling time and costs, and absence of a defined and agreed upon way of working. We observed that the process issues are related to areas such as multidisciplinary group project management, interdisciplinary communication, technology, stress, qualification of the many required skills, and the relationship with the customer and the end users.

The observed needs for improvement show that a possible solution may be found in facilitating courseware development teams in their knowledge needs. In this research project, we designed and implemented an instrument that supports the sharing of knowledge and expertise in an educational project environment. The effectiveness of this instrument, called Performer, on project estimates and on the general experiences of the team members has been measured at our company of study.

Proposed solutions

To be able to precisely identify the functional requirements of the required solutions, we have taken the CMM suggestions as a reference. According to CMM, a key prerequisite is that the project process is defined and agreed upon in an unambiguous way. All project phases, roles, objectives, tasks and activities must be defined. In addition, a metric must be in place to be able to measure quantitative project variables such as time, cost, and effort. Moreover, the metric should allow for an analysis of whether or not the implemented solutions are indeed being effective or not.

Within the company of study this has led to the design, realization and implementation of a knowledge management solution called Performer. Within Performer, the project process is unambiguously defined in terms of project phases, project roles, and objectives for each combination of project phase and project role. Each objective in the resulting matrix becomes a placeholder for knowledge elements. A knowledge element is a template or a best practice; this can take the shape of a presentation, a software executable, (written) documents, references to publications or references to sites on the World Wide Web, or courseware demos. In addition, Performer allows for experience questions to be connected to one or more objectives. Experts answer experience questions; these individual estimates can, combined, make up a consensus estimate. The use of the knowledge elements and consensus estimates potentially increases the effectiveness of the way of working of the entire project team. Performer is set up as an intranet application that generates html pages from a database that models the solution described above.

The implementation and deployment of a solution such as Performer in the project organization encompasses much more than just setting up a technical environment and granting access to employees. According to KnowledgeWorks, a method for implementing knowledge management, a careful balance is required between the four aspects of organization, culture, content, and infrastructure. The organization aspect includes procedures for keeping alive the knowledge management organization; also, it includes a deployment process with roles, tasks, and responsibilities to guarantee a minimum amount of quality. The content aspect includes all knowledge, expertise, and individual estimates of the employees; a certain amount (the "critical" amount) of knowledge must be ready in the system by the time it is used by employees. The cultural aspect usually requires the largest budget for implementation; some forty percent. This is about getting people motivated to share knowledge, and to keep people enthusiastic. The implementation of Performer at the company of study has been a success chiefly because the four aspects above were carefully taken into account. Over a thousand employees now use Performer actively.

In addition to Performer, a metric has been designed to allow us to measure whether or not the use of Performer in projects results in (*a*) improved estimates of the five quantitative project variables (Cost, Time, Effort, Size, and Risk), and (*b*) significantly more positive project experiences. This metric, called the ICOM metric, facilitates the control over all the categories of project problems that had been identified in the problem analysis. In addition, the qualitative project variables may be

administered and analyzed in the form of project experiences. This collection of data served as the basic instrument for testing the hypothesis through a main experiment. The Performer application is clearly an instrument for the project teams themselves, while the ICOM metric was used as a measurement instrument for the researcher.

Experiment and results

We observed that the need for a more mature project process was born out of a number of process problems at our company of study. We conducted a pilot study and a main experiment to test the hypothesis. The pilot study consisted of sixty interviews distributed over ten courseware projects, using a set of structured questionnaires. An analysis of these interviewing sessions revealed the following set of process problem categories: Project Management (PR); Qualification of required skills (QU); Communication (CO); Technology (TE); Stress and psychological problems (PS); and the relationship with the customer and user (CU). We call these the qualitative project variables, or *project themes*. A final category of project problems, "political problems", has not been included in the research project because we cannot exert a sufficiently large amount of influence over these.

Within the main experiment we looked in a quantitative way to the difference between estimates and actual values of the quantitative project variables: time, cost, effort, product size, and a risk factor. We name the difference between the estimated and the actual values the *discrepancy factor* of the project. The use of exactly these quantitative variables is also found elsewhere in existing literature on software process improvement. However, with educational multimedia, some specific issues must be addressed. To determine the size of a multimedia application, we have constructed a special formula with a new unit of measurement, the SIUN (short for Size Unit); this is derived from function point analysis theory. The risk factor is determined using a special checklist that was already available at the company of study. We found that on an aggregated level the discrepancy factor for projects where Performer was used did become significantly smaller. In other words, project estimates are better if Performer is used.

With the second part of the main experiment, we examined the same set of ten courseware projects. Five of these projects did not use Performer in any way, while another five of these projects did use Performer to some degree. All ten projects were done in the period spanning 1996 to 2000. The interviews that belong to these ten projects have been analyzed by a group of three independent researchers, to ensure undependability of the experiment. The researchers categorized each interview answer on at most two qualitative project variables (project themes). The inter-reliability of these three researchers was found to be overall higher than the required minimum level of eighty percent.

Most project experiences were reported on project management, interdisciplinary communication issues, and the relationship with the customer. We found that not all individual project experience themes were significantly more positive, but overall speaking, the aggregated project experiences did become significantly more positive. We observed that while only 33% percent of all experiences turned out to be positive with the "No-Performer" projects, as much as 50% of the experiences were positive with the "Yes-Performer" projects. A surprising observation was that the "technology" category was mentioned least of all categories. This was also the only category that showed a tendency towards more negative experiences.

Discussion of results

An important objection to the results that we found may be that the educational multimedia experts simply became more experienced during the five years that the investigation took, automatically resulting in more positive project experiences. However, this is not very likely because the attrition rate (personnel turnover) within the company of study was about twenty percent on average during those five years; in addition, new employees started on all levels of seniority. Moreover, several projects that did not use Performer were also conducted later on in the research project, even though Performer was already fully operational. It must be noted, though, that the definition of "using Performer" must not be taken too strict: we have not defined what it means to "use" Performer, although we do know that this includes searching for knowledge elements in all project phases by several roles/disciplines, several times a week.

A second important objection may be that the results found in this empirical experiment may well be valid for this particular company of study, but the experiment may very well have a different outcome at other organizations. Regrettably, it turned out to be impossible to repeat the experiment in additional organizations, for political reasons, even though two strong efforts have been undertaken to do this. This is why we have attempted to define the boundaries, assumptions, constraints and other "noise" factors as precise as possible.

The combination of a knowledge management solution that is strongly geared towards the way of working of the project organization, combined with a project experience database that serves as a thermometer for the satisfaction level of the project team turns out to work well to improve the overall process maturity of educational multimedia project teams. We must note here, too, that a successful implementation of knowledge management is dependent on a very careful balance between the factors Organization (procedures, roles, tasks, responsibilities), Culture (motivation, enthusiasm, keeping the spirit alive), Content (knowledge elements, individual estimates) and Infrastructure (technical aids, databases, network).

Conclusions

The results of the pilot study and our main experiment show that the development process of courseware is experienced as more positive using Performer than with projects in which such a solution was not at hand. We conclude that the problems in educational multimedia applications are mostly caused by a lack of experience in managing multidisciplinary teams, poor internal team communication, and problems that have their cause at the customer's side, with the project team failing to manage the customer's expectations in an adequate way. The results confirm the usefulness of a knowledge management instrument that is geared towards the way of working of the project team. The use of our knowledge management instrument has measurably improved the process of realizing courseware applications.

The analysis of the quantitative project data showed that the estimates of time, cost, effort, and product size do significantly improve if the project team uses the knowledge management instrument Performer. Our analysis also shows that the project estimates are dependent on more factors than just the usage of a knowledge management instrument. Especially the level of seniority of the project manager, the uncertainty of the functional requirements, and hidden political agendas can have a large amount of influence on the discrepancy factor.

The general conclusion is that to formally employ knowledge management within educational multimedia project organizations does have the potential to positive contribute to the maturity of the project process, on the condition that a sufficient amount of attention is being paid to the implementation aspects organization, culture, content, and infrastructure. As a consequence, this means that the entire project process, especially the way of working, must be defined in an unambiguous manner; moreover, formal procedures are required for the gathering, categorizing, disseminating and keeping alive of the most important formal knowledge and experience within the project organization. In addition, a metric program must be initiated and managed to be able to determine whether or not the project process becomes more mature indeed. We also observed that the same knowledge management instrument may be applied to other types of software development. Additional research is required to determine if the results of this empirical research are applicable to other organizations and cultures as well.

Samenvatting (Summary in Dutch)

Van het proefschrift van Jan-Willem van Aalst, getiteld "Knowledge Management in Courseware Development" ("Kennismanagement in educatieve softwareontwikkeling"), publicatie april 2001.

Context van het onderzoek
Multimedia wordt steeds vaker een standaard element in educatieve systemen. De multimedia factor is vaak mede bepalend voor de belevingswaarde van het product voor de gebruiker, en heeft invloed op het leereffect van de applicatie voor de gebruiker. Educatieve multimedia systemen bestaan uit hardware, software, procedures, data, en mensen. De educatieve multimedia applicatie (courseware) is dan de hardware, software, en de data. De mensen die te maken hebben met zo'n applicatie zijn te onderscheiden in gebruikers, klantorganisaties, en projectorganisaties die de applicatie ontwikkelen. De klantorganisaties zijn vaak grote commerciële bedrijven in de industrie en dienstensector. De projectorganisaties zijn vaak kleinere dienstverlenende ondernemingen, en bestaan veelal uit multi-disciplinaire teams met rollen, taken, procedures en bevoegdheden, en een vastgestelde manier van werken. In deze teams vindt men disciplines zoals projectmanagement, didactisch ontwerp, functioneel ontwerp, technisch ontwerp, interactieontwerp, grafisch ontwerp, video en audio ontwerp, en realisatie van genoemde onderdelen.

In courseware ontwikkeling zijn productvraagstukken en procesvraagstukken te onderscheiden. De kwaliteit van een courseware product is gedeeltelijk afhankelijk van het volwassenheidsniveau van het ontwikkelproces. Ons onderzoek richt zich op de procesvraagstukken in courseware ontwikkeling. Ons onderzoek richt zich op de verbetering van de volwassenheid van het courseware ontwikkelproces, met de nadruk op het multidisciplinaire en multimedia karakter. Hiertoe kiezen we een werkdefinitie van het begrip volwassenheid, om het meetbaar te maken. We kiezen het raamwerk van het Capability Maturity Model (cmm) omdat dit nu geaccepteerd en veelgebruikt is in algemene software procesverbetering. Met andere woorden, ons onderzoek richt zich op wijzen om het proces van courseware ontwikkeling meer volwassen te krijgen in termen van het CMM. We volgen hierbij de interpretieve onderzoeksmethode: we observeren de volwassenheid van het courseware ontwikkelproces in een empirische omgeving, en stellen vervolgens op een inductieve wijze een oplossing voor, die we implementeren in het courseware ontwikkelproces. Vervolgens meten we de effectiviteit van onze oplossing. De onderzoekshypothese veronderstelt dat het beschikbaar hebben en gebruiken van een instrument dat courseware ontwikkelteams faciliteert in hun kennisbehoeften resulteert in betere projectschattingen, meer positieve en minder negatieve projectervaringen voor verscheidene projectthema's. Hiertoe kiezen we een werkdefinitie van de begrippen kennis en kennismanagement (zie paragraaf 2.4).

Om de behoeften voor verbetering van het courseware ontwikkelproces helder te krijgen, voeren we een probleemanalyse uit bij een commerciële ICT services organisatie, ons "studiebedrijf". In de vorm van een pilot study voeren we zestig projectinterviews uit. De resultaten laten zien dat de gerapporteerde verbeteringsbehoeften goed passen in de vijf niveau schaal van het CMM. Karakteristieken van volwassenheidsniveau 2, het *herhaalbare* niveau omvatten het ontbreken van formele procedures voor het beheersen van doorlooptijd en kosten, en het niet aanwezig zijn van een geïntegreerde en alom geaccepteerde manier van werken. We observeren dat de procesvraagstukken gaan over gebieden als multidisciplinair groep projectmanagement, interdisciplinaire communicatie, technologie, stress, kwalificatie van de vele benodigde vaardigheden, en de relatie met de klant en eindgebruikers.

Uit de geobserveerde verbeteringsbehoeften concluderen we dat een mogelijke oplossing gezocht moet worden in het faciliteren van de kennisbehoeften van de ontwikkelteams. In dit onderzoeks-project ontwerpen en implementeren we een instrument dat het delen van kennis en expertise in een educatieve projectomgeving ondersteunt. De effectiviteit van dit instrument, genaamd Performer, met betrekking tot projectschattingen en algemene projectervaringen is gemeten bij ons studiebedrijf.

Voorgestelde oplossingen
Om het pakket van eisen aan de benodigde oplossing te kunnen identificeren kiezen we het CMM. volwassenheidsraamwerk Volgens CMM moet in ieder geval het projectproces ondubbelzinnig en formeel gedefinieerd zijn, en alle taken, verantwoordelijkheden en activiteiten voor alle rollen moeten gedefinieerd zijn. Daarnaast moeten er methoden zijn om kwantitatieve projectvariabelen zoals tijd, kosten en manuren goed te kunnen administreren en in de hand te houden. Er moet bovendien een meetinstrument worden opgezet om te kunnen vaststellen of de beoogde verbeteringen inderdaad gerealiseerd worden.

Deze adviezen leidden binnen het studiebedrijf tot het ontwerp, de realisatie en de implementatie van het kennismanagement instrument genaamd Performer. Binnen Performer wordt het projectproces beschreven in termen van projectfasen, projectrollen, en doelen die voor elke combinatie van projectfase en projectrol moeten worden bereikt. Aan elk van die doelen kunnen dan weer kenniselementen worden geplaatst. Een kenniselement is een sjabloon of een voorbeeld van een succesproduct; die kunnen in de vorm zijn van presentaties, software, geschreven documenten, referenties naar literatuur of naar websites op het Internet, en opleidingen. Ook kunnen ervaringsvragen aan de genoemde doelen worden gekoppeld. Op zo'n ervaringsvraag worden meerdere ervaringsantwoorden gegeven, die gezamenlijk een kengetal kunnen vormen. Het gebruik van de kenniselementen en kengetallen versnelt de taken van de projectmedewerkers aanzienlijk (in potentie). Performer is opgezet als intranet applicatie die html pagina's genereert uit een database waarin bovengenoemde oplossing is gemodelleerd.

Het invoeren van Performer in de projectorganisatie omvat meer dan het realiseren van de technische omgeving en het toegang verschaffen van medewerkers tot Performer. Volgens KnowledgeWorks, een methode voor het implementeren van kennismanagement, dient er bij de invoering van een kennismanagement oplossing een goede balans te zijn tussen de vier aspecten organisatie, cultuur, inhoud, en infrastructuur. De organisatie omvat procedures voor het levend krijgen en houden van kennismanagement, en het opzetten van een beheerproces met rollen, taken, en bevoegdheden om de infrastructuur en de inhoud "levend" te houden en van een voldoende hoog kwalitatief niveau. Het inhoudsaspect omvat alle kenniselementen en de ervaringscijfers van de medewerkers; er moet een zekere kritische massa aan inhoud aanwezig zijn om de oplossing initieel waarde te laten hebben. Het cultuuraspect vraagt in het algemeen het grootste gedeelte van het budget voor invoering, zo'n 40%. Hier gaat het om het motiveren van medewerkers om hun kennis te delen, en ook om medewerkers enthousiast te houden. De implementatie van Performer is bij het studiebedrijf, met name door voldoende aandacht te besteden aan deze vier aspecten, een succes geworden.

Naast Performer is er een meetinstrument ontwikkeld om te kunnen bepalen of het gebruik van Performer door projectteams resulteert in (*a*) betere schattingen van de vijf kwantitatieve

projectvariabelen, en (*b*) significant positievere projectervaringen. Dit meetinstrument, de ICOM metriek, faciliteert het beheren van alle categorieën projectproblemen die in de probleemanalyses waren geïdentificeerd; bovendien kunnen de vijf belangrijkste kwantitatieve projectvariabelen worden geadministreerd in de vorm van projectervaringen. Deze verzameling van data diende als basisinstrument om de hypothese te kunnen testen door middel van een formeel experiment. De Performer applicatie is duidelijk als instrument voor de projectteams gebruikt, en de metriek met ervaringsdatabase als instrument voor de onderzoeker.

Experiment en resultaten
De behoefte aan een meer volwassen projectproces kwam voort uit een aantal procesproblemen waarmee de educatieve multimedia afdeling binnen het bedrijf van studie te maken had. Er is een voorstudie en een formeel experiment uitgevoerd om de hypothese te toetsen. Hiertoe zijn zestig interviews in tien courseware projecten gehouden met behulp van een set van gestructureerde vragenlijsten. Uit deze interviews kwamen de volgende procescategorieën naar voren: Projectmanagement (PR), kwalificatie van vaardigheden (QU), communicatie (CO), technologie (TE), stress (PS), en relatie met de klant (en gebruikers) (CU). Dit noemen we de kwalitatieve projectvariabelen. Verreweg de meeste projectervaringen zijn onder te brengen bij één of meerdere van deze categorieën. Een laatste categorie, "politieke problemen" is niet meegenomen in het onderzoek omdat hier te weinig invloed op uit te oefenen is.

In het formele experiment werd op een kwantitatieve wijze gekeken naar het verschil tussen de schattingen van de kwantitatieve projectvariabelen: doorlooptijd, kosten, manuren, en product-grootte, en de uiteindelijke waarden van deze variabelen. Het verschil hiertussen noemen we de discrepantiefactor van een project. Het gebruik van deze kwantitatieve variabelen komt overeen met algemene benodigdheden voor projectbeheersing in de literatuur. Echter, bij educatieve multimedia speelt een aantal specifieke zaken. Ter bepaling van de grootte van een educatieve multimedia applicatie is een speciale formule ontwikkeld met een eigen resultaateenheid, de SIUN (voor Size Unit), afgeleid van functiepunt analyse theorie. De discrepantiefactor bleek inderdaad significant kleiner te zijn bij courseware projecten waarbij Performer is gebruikt.

In het tweede deel van het experiment zijn dezelfde tien projecten geanalyseerd, waarvan vijf gebruik maakten van Performer en nog eens vijf géén gebruik maakten van Performer. Deze projecten speelden alle in de periode 1996-2000, door elkaar heen. De interviews die bij deze tien projecten horen, zijn door drie onafhankelijke interview analysatoren gecategoriseerd op ten hoogste twee van de kwalitatieve projectvariabelen die hierboven zijn genoemd, om de onafhankelijkheid van het onderzoek te waarborgen. De onderlinge betrouwbaarheid tussen deze drie analysatoren lag gemiddeld boven het vereiste minimum van 80%.

De verdeling van procesproblemen bleek niet significant te veranderen tussen de twee sets van projecten. De meeste projectervaringen gingen over projectmanagement, communicatie, en de relatie met de klant. Wel bleek de aard van de projectervaringen duidelijk positief toe te nemen: 33% positieve ervaringen bij de Pre projecten, tegen 50% positieve ervaringen bij de Post projecten. Hoewel niet iedere afzonderlijke projectprobleem categorie significant positiever werd, bleken de projectervaringen als geheel wel significant positiever. Met name bij projectmanagement en relatie met de klant werden méér positieve projectervaringen geregistreerd. Tegen de verwachting in werd in alle projecten de categorie "technologie" het minst genoemd. Ook was deze categorie de enige die een negatieve verschuiving in projectervaringen liet zien.

Interpretatie van resultaten
Een belangrijk bezwaar tegen deze resultaten kan zijn dat de multimedia experts meer ervaren zijn geworden in het uitvoeren van hun projecten gedurende de vijf jaar dat het onderzoek duurde, en dat daardoor de projectervaringen positiever werden. Dit is echter niet waarschijnlijk omdat het verloop binnen het onderzoeksobject soms tot 20% opliep, en er nieuwe medewerkers op alle senioriteits-niveaus binnenkomen. Ook werden er in latere stadia nog projecten geanalyseerd die géén gebruik maakten van Performer, terwijl Performer wel al bestond. Wel is het zo dat "gebruik maken van Performer" ruim geïnterpreteerd moet worden: wat *gebruik* precies inhoudt, is gedefinieerd als het opzoeken van benodigde kennis in alle projectfasen door meerdere disciplines/rollen, en dit tenminste eens per week. In de praktijk zien we een veel zwaarder gebruik van Performer.

Een tweede belangrijk bezwaar kan zijn dat de resultaten wellicht valide zijn binnen één onderneming, maar dat het experiment in een andere onderneming tot andere resultaten kan leiden. Het bleek echter om politieke redenen niet mogelijk te zijn om het experiment te herhalen in soortgelijke ondernemingen, hoewel twee pogingen hiertoe zijn ondernomen. Daarom is in het experiment getracht zo duidelijk mogelijk de randvoorwaarden, aannames en beperkingen te benoemen die als "ruis" factor konden worden geïdentificeerd.

De analyse van de discrepantiefactor van de geanalyseerde projecten laat zien dat de schattingen afhankelijk zijn van andere factoren dan alleen maar het gebruik van een kennismanagement oplossing. Met name de senioriteit van de projectmanager, de onzekerheid over het pakket van eisen dat aan het product wordt gesteld, en verborgen politieke agenda's blijken van invloed te zijn op de discrepantiefactor.

De combinatie van een kennismanagement oplossing die sterk gericht is op de manier van werken van de projectorganisatie, in combinatie met een project ervaringsdatabase die als "barometer" dient voor de tevredenheid van het projectteam, lijkt een goede combinatie om de algehele procesvolwassenheid van educatieve multimedia projectteams te verhogen. Wel lijkt de succesvolle invoering van kennismanagement sterk afhankelijk van een juiste balans tussen de factoren Organisatie (procedures, rollen, taken, bevoegdheden), Cultuur (motivatie, enthousiasme, levend houden), Inhoud (kenniselementen, kengetallen), en Infrastructuur (Technisch gereedschap, database, netwerk).

Conclusies
Het ontwikkelproces van projecten waar Performer is gebruikt werd algemeen als significant positiever ervaren dan projecten waarin deze kennismanagement oplossing niet voorhanden was. Eén van de categorieën, technologie, gaf een kleine negatieve verschuiving te zien; echter, het aantal ervaringen op deze categorie was, verrassend, op alle projecten de minst genoemde categorie.

De resultaten van het onderzoek lijken erop te duiden dat problemen in educatieve multimedia projectteams voornamelijk worden veroorzaakt door gebrek aan ervaring in het managen van multidisciplinaire teams, onvolwassen interne communicatie, en problemen die aan de klantzijde worden veroorzaakt, en waar het projectteam te vriendelijk op inspeelt. De resultaten lijken er ook op te duiden dat het gebruik van een kennismanagement gereedschap, gericht op de manier van

werken van het projectteam, een stimulerende en ontladende werking heeft op het proces van het ontwikkelen van courseware producten.

De algemene conclusie is dat het inzetten van een formeel kennismanagement instrument in educatieve multimedia projectorganisaties positief kan bijdragen aan de volwassenheid van het courseware ontwikkelproces, mits voldoende aandacht wordt besteed aan de aspecten organisatie, cultuur, inhoud, en infrastructuur, d.w.z. volgens de KnowledgeWorks methode. Dit houdt in dat het projectproces, en de manier van werken, ondubbelzinnig moet worden vastgelegd, en formele procedures worden opgezet voor het verzamelen, categoriseren, beschikbaar stellen, en levend houden van de belangrijkste formele kennis en ervaring van de projectorganisatie. Bovendien moet een meetprogramma worden opgezet om vast te kunnen stellen of het eigen projectproces inderdaad significant verbeterd wordt. Een belangrijke observatie was dat onze manier van kennismanagement ook bruikbaar is voor andere typen softwareontwikkeling. Verder onderzoek zal moeten uitwijzen of de resultaten van dit onderzoek ook toepasbaar zijn in andere organisaties en in andere landen.

Acknowledgments

This thesis presents the results of a five-year research project on improving process maturity of educational IT project teams using a knowledge management solution. The thesis aims to contribute to the young but growing body of knowledge on ways to successfully use knowledge management for software process maturity improvement. The research stakeholders that are involved are, in no particular order:

- Atos Origin. This is an ICT services company acting as the financial sponsor and as company of study. More information about Atos Origin may be found on www.atosorigin.com. Notable Origin departments that have been involved are:
    - Atos Origin Netherlands Holding. Representative: Toon Witkam (1996-1999), Richard Thoben (1999-2001).
    - Atos Origin eBS. Representatives: Frans Hartemink, Paul Westeneng, Johan Vader.
- Delft University of Technology. The department that was involved is called ITS (Information Technology and Systems). The role of Delft University was mostly a scientific one, and a quality-assuring one. Representatives: prof. Jan Dietz and Charles van der Mast. More information on www.its.tudelft.nl.
- Leiden University. The role of Leiden University was mostly a scientific one, from a non-technical point of view. Representative: Prof. Gerard Kempen.
- BAAN. Representative: Johan Versendaal, a member of the research steering committee who switched from Atos Origin to BAAN halfway during the research project, but committed to following the research project throughout.

I would like to specifically thank the following people, without whose help this thesis would never have reached its publication date:
Charles van der Mast. Advisor on an almost weekly basis, Charles has helped to assure a high level of quality of the research and has often pointed out aspects that perhaps would otherwise have been neglected. His extensive expertise on developing educational software has been of great value to this work.
Toon Witkam. Toon has been enthusiastic about the possibilities that this research may offer from the start, and in the role of research budget holder of Atos Origin he has provided a five-year commitment to sponsoring the research. Throughout the reorganization turmoil, this commitment held its ground.
Jan Dietz. Despite an ever-busy time schedule, Jan has always found the time to attend the steering committee meetings and to make sure the research project kept its focus. His constructive comments on concept versions of this thesis have improved its quality substantially.
Gerard Kempen. On the difficult point of quantitative content analysis, Gerard has helped out in a highly constructive way, and his contributions are woven throughout this thesis.
Richard Thoben. His view on research, namely that it must ultimately add value for those whom the research is concerned about, has constructively improved the focus of this work.
Johan Vader. Involved from the beginning at the department of study, Johan has always provided useful advice and has identified various nuances for several issues.
Frans Hartemink. Director of the department of study throughout the research process, Frans has believed in the value of this research project from the start and has always provided constructive

suggestions in an enthusiastic way. Furthermore, he has substantially improved the marketing possibilities for this work.
Jan den Heijer. Jan has spent many hours analyzing project interviews of a company he has never worked at, for a person he didn't really know. His trust in the research project is admirable.
Johan Versendaal. Next steering committee meeting, perhaps?
Tom Carey. Followed the research from a great distance (Guelph, Canada) and has always provided constructive suggestions.
Marcel Theunissen. Followed from an even greater distance (Melbourne, Australia) and the same applies to him.
Jenny Preece. Provided constructive feedback at the early stages of the research project.

I conclude by mentioning all multimedia experts at Atos Origin who contributed to the research by providing the time for interviews: Sjaak van den Akker, Chagiet Altmann, Corinne Barents, Janneke van den Berg, Peter Berger, Marion Blinksma, Paul de Boer, Simone Boezewinkel, Paul van Buuren, Janet Derksen, Joery van Druten, Ton van Dijk, Nicole van Eerdenburg, Frits de Haas, Susanne van der Heide – Willemsen, Jeroen Hermkens, Erica Heinsen, Esther van den Heuvel, Hein van den Heuvel, Marcel Heijmans, Dorien Hein, Karin Hoek, Martin Koekenberg, Judith Kruijdenberg, Désirée Kwant, Jo Martens, Jet van Mensvoort, Emmy Mudde, Vincent Perquin, Natasja Schipper, Berno van Soest, Pieter van der Spiegel, Eef Stavenuiter, Bennie Vaasen, Maurice Verhalle, Dirk Weber, Paul Westeneng, Piet van Wier. An Additional thank-you to Bas van Dijk, Peter de Vos, Jules Ernst, Martin van Oostende and Adrie Keur. Without the support of all these people, this thesis would not have been possible.

A. Selected Interviews

In this appendix, we provide three samples of interviews that were conducted for the content analysis that we described in section 6.2. The text is extracted from the online ICOM project experience database.

Table 27. *Sample 1 of 3 of selected project interviews*

20. *To what degree are you able to apply your experience and knowledge from previous projects?* (main build phase)
"To me: making a bid this way was new; learned a lot: for example, how to write for such a specific customer. Specifications were too easy for me. Organising an evaluation session wasn't new to me either. Making a report, either." *Recorded by Maurice Verhalle (quality assurance manager), 26-Apr-99.*
21. *Are there problems regarding the project management? What may be improved?* (main build phase)
"to customer, don't know, was covered by the project manager (as should be, I think, he arranged everything for the project team so they didn't have to worry about that)." *Recorded by Maurice Verhalle (quality assurance manager), 26-Apr-99.*
22. *Are there problems regarding the project management? What may be improved?* (main build phase)
"Team PM: is related to project bid. Project pressure was high at both locations, because we did fixed price within a fixed time limit. Estimation was too tight, was based on our experiences with basic components. " *Recorded by Maurice Verhalle (quality assurance manager), 26-Apr-99.*
23. *Are there problems regarding the working environment right now?* (main build phase)
"The chairs and office desks are horrible. Also, there was a refurbishing project going on. Geographical distance from Amsterdam to Eindhoven was a problem. I had to go by car, and I spent some 3,5 hours each day." *Recorded by Erica Heinsen (specificator), 26-Apr-99.*
24. *Are there problems regarding the project management? What may be improved?* (main build phase)
"if you know what transactions you're talking about, use that as input for the training beforehand. Because you can fingerpoint the transactions and ask for explanation. We had two Sap experts at the customer site, from which we've learned a lot. Very useful." *Recorded by Maurice Verhalle (quality assurance manager), 26-Apr-99.*

Table 28. *Sample 2 of 3 of selected project interviews*

12. *What are problems regarding communication during this project?* (main build phase)
"The project planning was not very clear, and much of it still needed to be made when we started. Informal planning was communicated orally to the project team, though." *Recorded by Jan-Willem van Aalst (specificator), 10-Oct-97.*
13. *What do you think of the quality of the working environment of for the project team?* (evaluation phase)
"We were propped in one space with our team and people from the customer, which makes it hard to discuss things with eachother without disturbing others. Atmosphere is also important, the psychological distance between customer and team (for example, a review asks for a business-like atmosphere), which can be difficult sometimes." *Recorded by Jan-Willem van Aalst (specificator), 27-Sep-97.*
14. *What did you learn regarding the role of the customer (participation, testing, delivery etc)?* (evaluation phase)
"The customer likes to have lots of extras but doesn't want to pay additionally for it." *Recorded by Jan-Willem van Aalst (specificator), 26-Sep-97.*

15. *How do you experience project pressure? Are there things that you think should be said aloud?* (main build phase)

"The geographical distance to the working location is about 80 km, a one-hour drive for me, which is okay for me, although I would mind a two or three hour drive. Oh, and by the way, the customer doesn't care."
Recorded by Jan-Willem van Aalst (specificator), 25-Sep-97.

16. *What are emotional problems for this project?* (main build phase)

"The project team ventilated large problems to the project administrator and the customer. We all felt the work pressure, and we all tried to convince the project leaders that nothing could be allowed to go wrong, because of the short time span."
Recorded by Jan-Willem van Aalst (programmer 4gl), 24-Sep-97.

17. *Can you usually convince the customer of a better solution on your part?* (main build phase)

"In a way we can convince the customer of a better solution on our part, but we're talking about these little extra things then. It's not as if they're an easily controllable group, we don't have that much influence."
Recorded by Jan-Willem van Aalst (specificator), 23-Sep-97.

Table 29. *Sample 3 of 3 of selected project interviews*

1. *What do you think of project pressure that you experienced during this project?* (evaluation phase)

"Project pressure wasn't overly high for me, especially because Toine joined the project. Some other project members did suffer from very high project pressure, but it's hard for me to take over some of their tasks."
Recorded by Karin Hoek (project administrator), 16-Jul-99.

2. *What were important problems regarding team communication for this project?* (evaluation phase)

"Communication aspects. Project management did not always communicate sufficiently; each time I had to start something new I did not have the required content available."
Recorded by Natasja Schipper (designer), 16-Jul-99.

3. *What's the number of errors found for each evaluation iteration? Severity?* (evaluation phase)

"The errors that were found were not very serious, most of them were cosmetic. Although, on the last testing day one of the graphic artists had already been put on another job when we still needed him, so he was brought back to our project for that day."
Recorded by Natasja Schipper (designer), 16-Jul-99.

4. *Did the customer come up with surprises, things you had not expected beforehand?* (evaluation phase)

"One. At one implementation site, the operating system that is used turns out to be and old version, so the quick reference cards don't show the right screen dumps over there, and the training material is not 100% correct anymore either."
Recorded by Natasja Schipper (designer), 16-Jul-99.

5. *Can you describe the review process that the customer did?* (evaluation phase)

"The customer was very enthusiastic, but had only a quick look at it."
Recorded by Natasja Schipper (project administrator), 16-Jul-99.

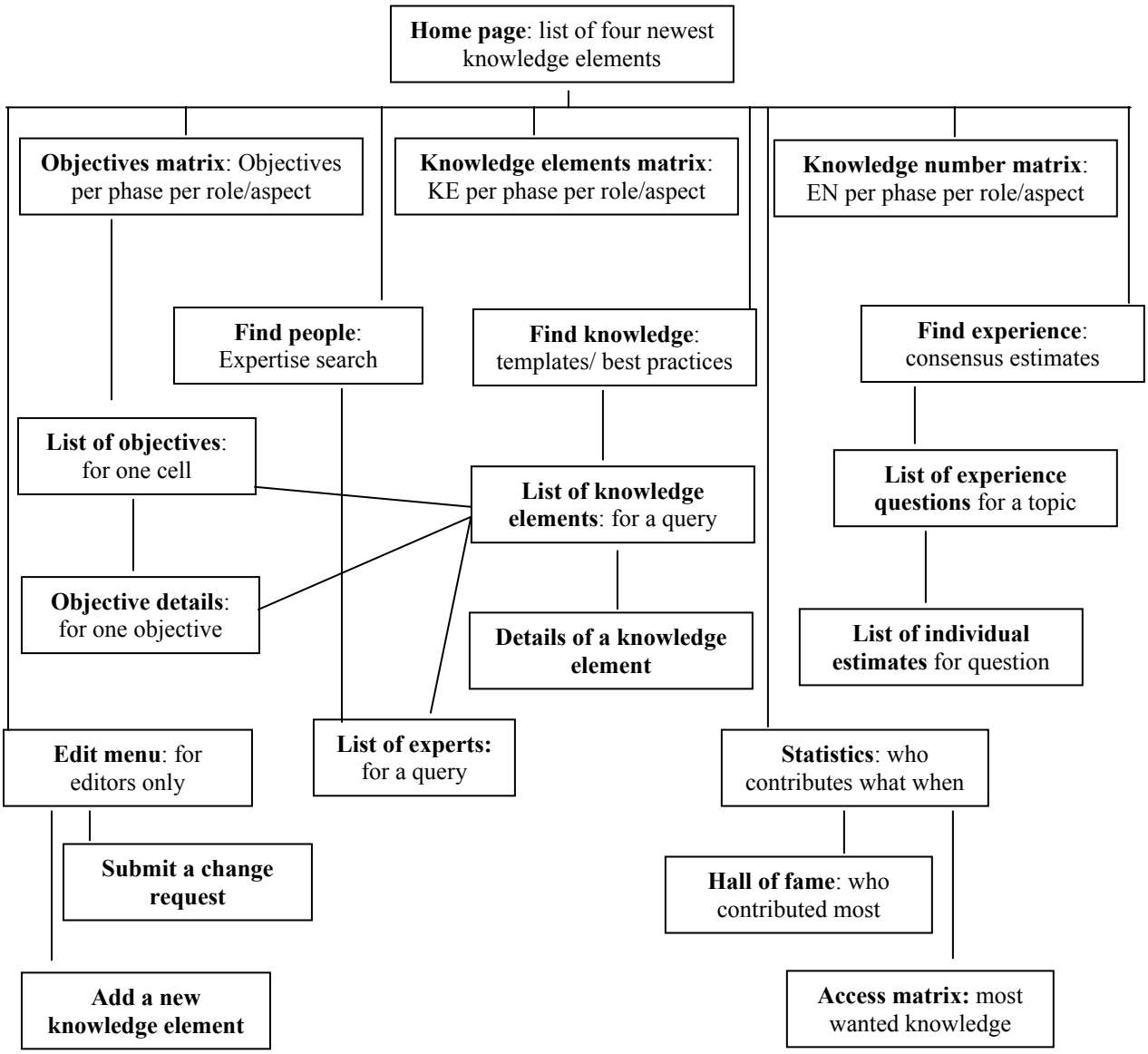
B. Selected Interview analysis

In this appendix, we present a sample of an interview that was analysed by the group of three independent researchers. They used the interview codebook described in chapter 6 to assign each interview answer to one or two project problem categories. This is shown at the right side of the picture, where each researcher was allotted two columns. Each column was made up of two cells, namely one cell for the actual project issue category, and a second cell to denote the nature of the experience: -2 for very negative, up to +2 for very positive. The researchers did their analysis completely independent of each other.

Are there any problems involving team communication?	There have been problems between the specification team and the programming team. For example, the testing was done in Eindhoven, so the technical team was not involved in that.	**CO**	**-1**	**PR**	**-1**	**CO**	**-1**	**PR**	**-1**	**CO**	**-1**		
What would you like to see in an experience database for later re-use?	Lesson to learn: make a knowledge number of the number of SAP work instructions that is possible to create in one day and use that in your next project. This number also changes in the course of time: first things are slow because of changes in work.	**QU**	**2**			**PR**	**-1**	**PS**	**-1**	**PR**	**-1**		
What are critical succes factors for this project?	After one or two weeks, get the project team together (technical team + specification team) without customer, to find out about any uncertain matters or preferences in the user interface. Now we did things that the customer didn't seem to want later on.	**CU**	**-1**	**CO**	**-1**	**PR**	**1**	**CU**	**-1**	**CO**	**-1**	**CU**	**-1**
Are there any problems involving team communication?	Relational aspects of communication in the entire team were okay. Process-aspects were okay, but the procedural communication caused some problems. It was not clear what the programming team wanted, and vice versa what the specification team wanted.	**CO**	**1**	**PR**	**1**	**PS**	**1**	**CO**	**-1**	**CO**	**-1**	**PS**	**1**
Are there any problems regarding communication with the customer?	Communication to customer was okay. A frustrating point was that we just had to make the CBT, and that's it. So we didn't really have any communication to the customer, that was organised through Frits. For the customer, we're welcome, but that's all.	**CU**	**1**	**PS**	**-1**	**PR**	**1**	**CO**	**2**	**CU**	**1**		
Were there any problems regarding project meetings?	Project meetings were okay. In general, they were brief because the specification team demanded it because of time pressure. We got the information that we needed (very brief, concise), a bit like a factory. This because of high pressure.	**PR**	**2**	**CO**	**2**	**CO**	**2**	**PS**	**1**	**CO**	**1**		
Can you say something about the project pressure?	Project pressure is very high. Cause: a wrong estimation of the number of specifications of work instructions that could be made in a given period of time. Origin was cheap, because Origin also payed some of it themselves to so they can keep the source.	**PS**	**1**			**PR**	**-1**	**PS**	**-1**	**PR**	**-1**	**PS**	**-1**
Did you miss any expertise on the project team?	We could have used more people on the project team, and these people were available. They should have been used on this project, they could have learned something of it.	**QU**	**1**	**PR**	**-1**	**PR**	**-1**			**PR**	**-1**		
What are critical succes factors for this project?	Organise a training session (one day) in the SAP FiCo content matter, this should have happened. Now we had two hour training, which was too short. Bring the project team up to date on the content matter.	**PR**	**-1**			**QU**	**-1**	**PR**	**-1**	**PR**	**-1**		
How is the communication between team members and the customer going?	Mr. J. of [redacted] was available to us to deliver assets to us, but that was informal. Formal communication was via Frits.	**CU**	**2**	**PR**	**2**	**CU**	**2**	**PR**	**2**	**CU**	**0**		

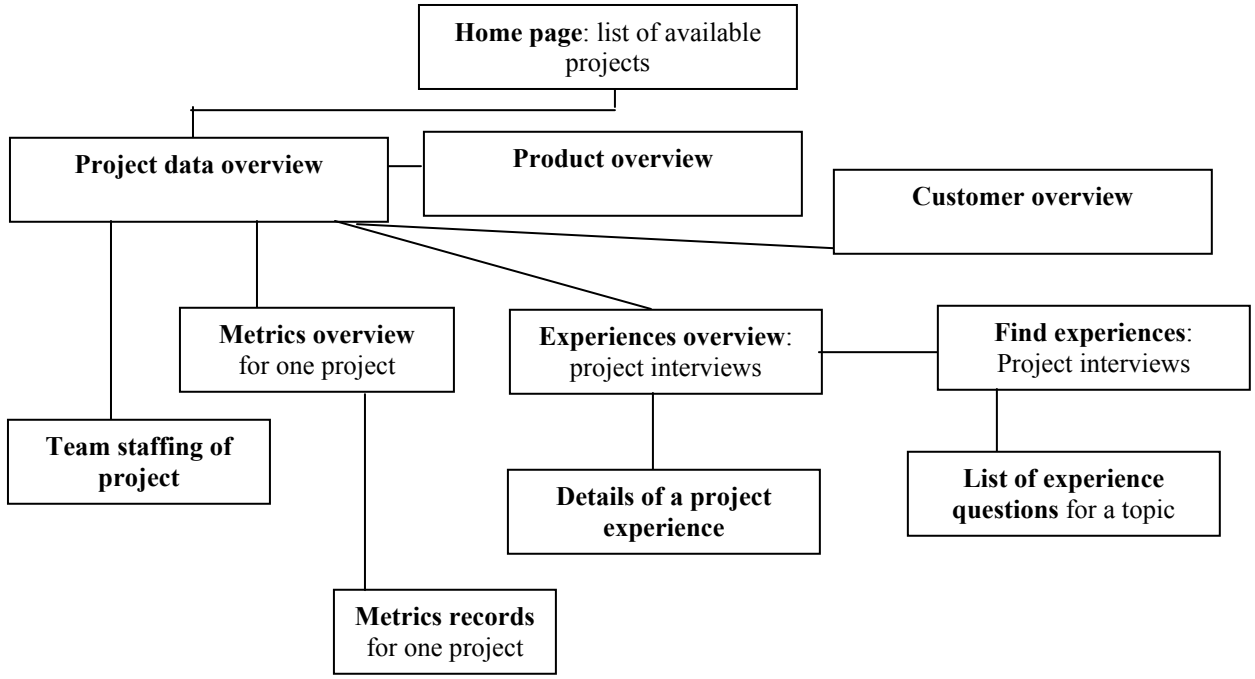
C. Performer documentation

In this appendix, we present the sitemap of the Performer application version 2.1, as running on the intranet of Atos Origin.



D. Experience database documentation

In this appendix, we present the site map of the ICOM project experience database.



E. Calculating the size of a multimedia application

The text in this appendix is edited from a publication found in [VAA2000].

In the research world of software engineering, much work has been done to find unambiguous objective formulas for determining the size of a software application [PUT1996]. Software process improvement tools have gratefully employed the results of that research to develop tools that can be used to improve the maturity of the software project process. However, these tools have up to now always failed to take into account the more complex highly interactive systems such as multimedia products (VAU1994). Multimedia products are not only more complex because of their highly interactive nature, but also because of the many disciplines involved and because of the various media involved. We present here a relatively easy way to objectively determine the size of a multimedia applications. We call the unit of this size the SIUN (for Size Unit).

Problem definition
When first attempting to determine the size of a software application, researchers started to count non-breaking lines of code. This could or could not include comment statements or empty lines. Many complex rules have been devised to make this counting an objective and unambiguous task [CON1986]. However, it was not long after the various methods for counting lines of code became commonplace, when people realized that the size of a software application could not be determined by just counting lines of code. Hence, more sophisticated methods were devised.

The most widespread of these by far is probably the method Function Point Analysis (FPA) [SOF1995]. Although this method seems to work in a satisfactory way for many "traditional" software projects, especially administrative software and process control software, the fields of Computer Based Training and multimedia needed something in addition, because the effort that is put into the production of a multimedia product cannot be expressed in lines of code or function points alone [ENG1996; VAA1998B].

More often than not, the heaviest components of effort are in visual and graphic design, video and audio production, or functional requirements analysis [VDM1995A]. For example, the graphics of some multimedia application may have taken up to 30% of all required man hours, while they can be traced back to only a relatively small number of function points, and not at all to lines of code. Therefore, we cannot suffice by just counting lines of code or function points. In many modern multimedia production tools, lines of code are completely absent. The field of multimedia production needs a way of determining the size of a product that takes into account these effort factors.

The question that props up immediately, then, is why do we need to know the size of a multimedia application after all? The answer is: to be able to supply the software improvement researchers or organizations with a scaffold for their solutions. Since these companies claim to be able to realize serious process maturity improvement [SOF1995] as long as we only have some basic variables in place (especially size!), we can now supply this size factor. Hence, multimedia project teams are able to supply all necessary variables for their own process maturity improvement.

Method and experiment

To construct our formula, we use – on an abstract level – the basic principle of Function Point Analysis, namely to take important relevant aspects (though not functions) of the product and to assign a weight to each of these aspects. This weight is determined according to the average amount of effort that is required to produce that aspect of the product. The average amount of effort, in turn, is determined from a sufficiently large number of projects that result in these multimedia products.

Thus, the research method is as follows:

1. Conceive a list of relevant effort-consuming aspects of a product, in such a way that they do not overlap in project tasks;
2. Determine a first estimation for the weight of each of these factors.
3. Analyze five to six multimedia products on these aspects;
4. Fine tune the weight factors according to one-to-one comparison of these product analyses.

In analyzing six multimedia products that were all realized in either 1997, 1998 or 1999, we conceived the list of aspects of table 1.

MULTIMEDIA PRODUCT ASPECT	WEIGHT	UNIT	COMPLEXITY
Lines of html code (# characters div. by 40)	0.10	Count(Char)	{1,2}
Lines of dynamic html code (# characters div. by 40)	1.20	Count(Char)	{1,2}
Lines of script code	1.80	Count(Char)	{1,2}
Uncompiled lines of 3rd gen. language code	3.20	Count(Char)	{1,2}
Uncompiled lines of 2nd gen. language code	3.50	Count(Char)	{1,2}
Graphics sources	0.30	Kilobytes	{1,2}
Animated graphics sources	0.40	Kilobytes	{1,2}
Realtime audio	0.40	Kilobytes	{1,2}
(Streaming) Video	0.45	Kilobytes	{1,2}
Database structure: # tables + # fields + #rel.	3.00	Count()	{1,2}

The weight of each multimedia aspect not only depends on the amount of effort that is required to produce it, but also on the unit that is chosen for that aspect. For example, graphics sources may well take up several thousands of kilobytes, and if not properly weighted, their influence would unrightfully overwhelm the effort for the lines of code (if any). Similarly, lines of html code are weighted according to the same principle, because html is not really a programming language. The unit column denotes what the result is of what is counted.

The complexity column is something special. The fact that, for example, the graphics factor is not large does not mean that it did not take a large amount of effort to create these graphics. Therefore, we can multiply the value by a complexity factor which ranges from 1 (not complex) to 2 (very complex), with digits in-between possible (for example, 1.65). Please note that we can also include web-based multimedia products since html and other scripting languages may also be included. A disadvantage is that the SIUN becomes time-dependent: if a new popular scripting language emerges, the definition of the SIUN will have to be adjusted.

Experiences with SIUNS

We have used an initial set of weight factors to compare two multimedia products produced by Atos Origin for international customers. Both of these were the results of similar-scoped projects with a courseware product as a result. Weights that seemed to have a disproportional large effect on the total size were then adjusted. The same was done for a third and a fourth project. For a fifth

project, we found that the weight factors were by then well tuned. All projects were in the range of $30,000 to $90.000. This one resulted in 2835 Siuns, which felt right when compared to less complex projects that resulted in products of 1146 and 1520 Siuns, respectively. A larger number of projects is required to get a statistic reliable range for each weight, though.

One notable subjectiveness, one that cannot easily be eliminated, is the complexity factor. One employee that would regard the graphics as highly complex, could be classified as rather simple by another.
Obviously, a larger set of projects would be needed to produce more statistically valid weight factors.
Please note that the Siun is not a measure for the size or complexity of the project; it particularly serves as variable about the size of multimedia products, especially for software process improvement tools.

Conclusions
While the software engineering industry is currently equipped with proven methods to predict project success through variables such as milestones, project cost, required effort, number of defects, and product size, the multimedia industry is now able to supply the most difficult of these variables, namely size, as well. The fact that the multimedia size unit, called Siun, is an unknown to software engineering, is not a problem, because these prediction methods need an arbitrary but consistent approach to counting. Therefore, the research on multimedia software process improvement now has a useful instrument for already available methods for predicting project success.

The paper may be found in full in [VAA2000].

List of abbreviations used

CAI	Computer Assisted Instruction
CAL	Computer Assisted Learning
CBL	Computer Based Learning
CBT	Computer Based Training
CHI	Computer-Human Interaction
CMI	Computer Managed Instruction
CMM	Capability Maturity Model
DSDM	Dynamic Systems Development Method
EA	Expected Agreement (in inter-reliability)
FDL	Flexible Distance Learning
GUI	Graphical User Interface
ICOM	Improving Control Over Multidisciplinary projects
IFO	Income From Operations
IE	Information Engineering
IS	Information System
KM	Knowledge Management
MMI	Man-Machine Interaction
OA	Observed Agreement (in inter-reliability)
PI	Programmed Instruction
SDM	Software Development Method
SE	Software Engineering
SPI	Software Process Improvement
UI	User Interface
eBS	e-Business Solutions

Resumé of Jan-Willem van Aalst

Jan-Willem van Aalst (1970) graduated from Delft University of Technology in 1995 on the subject of User Interface Design. His Master's thesis was an investigation into design rationale for teaching user interface design; the research was conducted in Canada at the University of Guelph. His master's thesis won the Dutch 1996 NGI-ASI award for best Informatics master's thesis. Since 1995 he has been working for Atos Origin. He started as a multimedia specialist at Atos Origin in Zoetermeer, and switched a year later to the Interactive Media Competence Center, a special department involved in creating courseware and multimedia applications. This is where the idea for a Ph.D. thesis on process improvement in courseware development emerged.

At Atos Origin, Jan-Willem has mostly been involved in web application development augmented by graphics development and interaction design tasks. He has been involved in advising large customers on intranet solutions. Jan-Willem is the technical producer of the knowledge management instrument called Performer, and of the multimedia project experience database called ICOM. He has also taught courses on user interface design and multimedia design at PAO Informatica. Recently (2001) he has been involved in information analysis for e-commerce projects.

Jan-Willem is more a generalist than a specialist, often performing a "bridge-like" function between design, realization, and user interface. Jan-Willem has a no-nonsense attitude on the one hand, but has a creative (artistic) side on the other hand. Interests include producing film music (see listen.to/thelonging/) and recent printed world atlases (see worldatlas.brinkster.net).

You can contact Jan-Willem at janwillemvanaalst@hotmail.com.

Index

D

E

Q

R

S

T

U

V

W

Y

Z