TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Angewandte Softwaretechnik

Dissertation

A Framework for Externalizing Information in Agile Meetings

Jennifer Schiller

TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Angewandte Softwaretechnik

A Framework for Externalizing Information in Agile Meetings

Jennifer Schiller

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 02.11.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 20.03.2011 angenommen.

Danksagung

Ich möchte mich herzlich bei Prof. Bernd Brügge für die Betreuung dieser Dissertation bedanken. Ohne seine Unterstützung und unsere wertvollen Gespräche wäre die Fertigstellung dieser Dissertation nicht möglich gewesen.

Des Weiteren gilt mein Dank Sabine Canditt, Norbert Weber und Ludger Meyer, und all meinen Kollegen bei der Siemens AG, die meine Promotion unterstützt und mir so einen Einblick in die aktuellen Herausforderungen der Industrie ermöglicht haben. Vielen Dank für die wertvollen Diskussionen. Außerdem bedanke ich mich ganz herzlich bei den Kollegen von Siemens Enterprise Communications und Alexander Hauptmann und seinem Team von der Carnegie Mellon University für die erfolgreiche Zusammenarbeit.

Ein großes Dankeschön an Thilo Kraft fürs Korrekturlesen und all meinen Kollegen am Lehrstuhl für Angewandte Softwaretechnik, im Besonderen an Monika Markl und Helma Schneider für die organisatorische Unterstützung.

Erklärung

Ich versichere, dass ich diese Arbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, 2. November 2010 Jennifer Schiller

Zusammenfassung

Software-Entwicklungsprojekte sind geprägt durch Kommunikation. Entscheidungen werden in Besprechungen gefällt, Anforderungen in Workshops mit dem Kunden definiert, neue Aufgaben in Besprechungen vergeben, und Probleme und Lösungsansätze diskutiert. Kommunikation bestimmt den weiteren Verlauf des Projektes. Um die besprochenen Informationen festzuhalten, wird im Anschluss an die Besprechung ein Protokoll verfasst. Die Protokollierung ist jedoch häufig eine unbeliebte Aufgabe, da das Schreiben eines Protokolls zeitaufwändig ist und es schwierig ist, gleichzeitig der Diskussion zu folgen und mitzuschreiben. Dadurch sind die Protokolle vielfach unvollständig oder fehlerhaft. Werden jedoch nicht alle Informationen richtig dokumentiert, kommt es zu Informations- und Wissensverlusten. Verstärkt wird dieses Problem durch die vermehrten informellen Besprechungen, die Bestandteil agiler Methoden sind und immer häufiger eingesetzt werden, da ein explizites Wissensmanagement nicht Bestandteil dieser Methoden ist.

Um die genannten Probleme detailliert zu adressieren, und somit außerdem eine Kombination von empirischen und traditionellen Prozessmethoden, im Besonderen in großen Unternehmen, zu ermöglichen, stellt diese Dissertation eine neue Methodik vor, um Wissen aus Besprechungen zu externalisieren und automatisch Protokolle zu erstellen. Dazu werden die Besprechungen aufgenommen und nach vordefinierten Besprechungselementen durchsucht. Dies geschieht auf Basis einer definierten Grammatik, die Besprechungsphrasen, wie z.B. die Definition eines Problems, oder eine getroffene Entscheidung, abbildet. Aus den gefundenen Phrasen wird das Protokoll erstellt und an alle Besprechungsteilnehmer versandt. Die extrahierten Informationen werden darüber hinaus an die eingesetzten Projektmanagement- und Aufgabenverwaltungswerkzeuge weitergeleitet. So wird beispielsweise eine zugewiesene Aufgabe direkt an das individuelle Aufgabentool gesendet und dort eine neue Aufgabe erstellt mit allen Informationen (Beschreibung, Fälligkeit, etc.) aus der Besprechung. Dieses Vorgehen des automatischen Protokollerstellens wird im STACHUS Framework abgebildet.

Das STACHUS Framework wurde als Prototyp realisiert, um Machbarkeit und Einsatzmöglichkeiten anhand eines experimentellen Fragebogens und mehrerer Fallstudien empirisch zu evaluieren. Die Ergebnisse dieser Evaluierung zeigen, dass die Externalisierung von Informationen, insbesondere aus informellen Besprechungen, schneller ist als manuell verfasste Protokolle, eine bessere Protokollqualität aufweisen, sowie ohne Mehraufwand und unter Einhaltung der agilen Prinzipien im Projektalltag einsetzbar ist.

Abstract

Software development projects are characterized by communication. Decisions are made in meetings, requirements are defined in workshops with the customer, tasks are assigned in meetings, and problems and solutions are discussed. Communication determines the progress of the project. To record the discussed information, a protocol is created after the meeting. However, writing a protocol is often an unpopular task, as it is time consuming and difficult to listen and write at the same time during the discussions. As a result protocols are often incomplete or incorrect. However, if not all information is correctly documented, meeting knowledge will be lost. The problem is worsened by an increase of informal meetings, especially in agile methods, which are more and more applied in software development.

To address these problems, and to allow a combination of empirical and traditional process control methods in large organizations, this dissertation proposes a new methodology to externalize knowledge from meetings and automatically generate a protocol. Meetings are recorded and the audio stream is searched for predefined meeting elements, based on a defined grammar for meeting key words, e.g. for defining a problem, or making a decision. Based on the extracted meeting phrases the protocol is generated and sent to all meeting participants and stakeholders. Additionally, the externalized information can be transferred to project- and task management tools. For example, an assigned task is forwarded to the individual task management tool after the meeting, where a new task is created, containing description, due date, etc. determined in the meeting. The automated protocol generation process is realized in the STACHUS framework.

A prototype of the STACHUS framework has been implemented to empirically validate the automated protocol generation and externalization of information from meetings, based on an experimental survey and three case studies. The results of the evaluation show that an automation of the protocol generation process improves the protocol quality, is faster than manually written protocols, reduces the protocol generation effort, and is applicable in compliance with the agile principles and practices.

Table of Contents

List of Figures

List of Tables

1 Introduction

Motivation and Overview

Today, the way of software engineering is changing – more and more companies are moving their defined software development process to an empirical process control model. The empirical model of process control exercises risk management and control through visibility, and frequent inspection and adaptation for situations where the inputs are varying, and the process is too complex to produce repeatable and predictable outputs. Empirical process control is the essence of all agile methodologies (Van Schooenderwoert, 2005).

Agile methods have continuously gained in importance since their first publication (Takeuchi & Nonaka, 1986), as these methods proved a valuable contribution to successful software development projects (Ambler, 2009) by embracing change, short iterations with complete software packages, test-driven development, and encouraging high quality software. More and more software engineering departments implement the agile practices and principles in their software development lifecycle, as the latest Agile Survey[1], conducted by VersionOne (2010), shows. Larger organizations are also contemplating to change their development processes to a lightweight, agile method.

With the introduction of agile methods, the number of informal meetings is rising, as face-to-face communication is emphasized (Beck & al., 2001). Agile practices and principles are based on tacit knowledge and communication. Tacit knowledge is implicit knowledge and information discussed in a meeting, which is kept only in the developer's memory. The results of informal meetings are not externalized in a protocol and project and process documentation is not necessarily written, if it is not part of the project deliverables (Poppendieck & Poppendieck, 2003).

Agile projects are often criticized for their missing focus beyond project borders (Longstreet, 2008), (Müller & al., 2005), and the missing documentation, which is necessary for a successful introduction of the agile methods in defined process control methods of large organizations. On the other hand, agile teams often deplore the additional work of writing documentation that is

[1] The Agile Survey has been conducted for 5 years now

not necessary when the team works closely together and meets every day. Well-written documentation supports organizational memory effectively, but is a poor way to communicate during a project (Ambler, 2006).

Protocols from 'traditional' meetings also have problems, for instance the meeting minutes are late, incomplete, incorrect, or unread (Lewis, 1997) (Waibel, et al., 2001). Nobody is willing to assume the task of the minutes taker, as writing a protocol is time-consuming and 'unpopular', so the protocols are published with delay or even too late to prepare for the next meeting. As it is difficult to listen and write at the same time during the meeting, the protocols are incorrect or incomplete, especially if the protocol is written after the meeting from memory. However, knowledge can be lost, if the information is only tacit or not correctly documented. Due to the volatility of communication, it is important to capture project information, discussed in meetings, in an externalized knowledge source, e.g. in documents, or store the information directly in project- and task management tools.

The agile methods, for instance Scrum (Schwaber & Beedle, Agile Software Development with Scrum, 2002) and XP (Beck & Andres, 2004), are originally designed for a software development team consisting of five to seven developers, working at the same location. Best practices have been proposed (Pichler, 2008) to use agile practices and principles in larger teams. Nevertheless, large organizations are not only confronted with large teams. The software development methods affect also departments such as Maintenance, Sales, or Quality Assurance (Schiller & Canditt, 2008). Thus, a holistic introduction of agile methods in a traditional process- or plan driven company fast meets its limits, resulting in challenges, as the empirical process model, representing an "agile lightness" with responsibility to change and flexibility, has to be merged with established processes (the defined process models), which makes organizational change difficult.

This dissertation claims that agile methods can be extended and enhanced with a knowledge externalization process in consistence with the agile principles, in particular without increasing the effort for agile teams. A second claim is that it allows the injection of empirical into defined process models. We will show that it is possible to instantiate the empirical process models in a large, traditional enterprise, and additionally facilitate the externalization of tacit knowledge to support organizational memory, the aggregated knowledge of a company.

Our hypothesis is that knowledge can be externalized in protocols, which are automatically generated, so teams with different process models can work together. An approach for externalizing information in an agile software development environment is developed, applicable in small and large projects, without increasing the workload of the agile team.

Continuous and fast change in an agile project leads the team to keep knowledge informal and tacit. Externalization of this discussed information in documents would paralyze that team, as the documentation is obsolete when it is published (Rüping, 2003) (Ambler, 2006). However, missing externalization of discussed information in informal meetings is a problem, as knowledge is lost, especially when introducing agile methods in large organizations.

We investigate a method that supports the externalization of knowledge in empirical software development processes. The framework STACHUS (Smart tool for ad-hoc communication, speech-recognition & document generation in agile software development projects) is designed to support the automatic generation of protocols and integration of the results in the project management workflow.

The focus lies on the detailed description of the framework for protocol generation and meeting workflow support, including the interfaces to project and task management tools, to transfer information discussed in a meeting, e.g. the assignment of a new action item, to the individual task management tool. Thus, effort for manually entering and updating the task within the tool is reduced.

Our hypothesis is that the effort of protocol generation is reduced with an automated protocol engine, the protocol generation process is accelerated, and the quality of the protocols is improved. The STACHUS framework is used to validate these hypotheses.

To demonstrate feasibility the framework is realized as a prototype. In three case studies, two in an academic environment and one industry case study, the hypotheses are validated. The case studies analyzed the effectiveness, effort reduction, and protocol quality improvement of the automated protocol generation process during a software development project. Moreover, the industry case study focused on the evaluation of the applicability of the framework in the project workaday life and the acceptance and practicability of the meeting grammar.

1.1 Overview

The dissertation is organized as follows. In the first part we lay the foundations for the development of the STACHUS framework by describing the impact of agile methods on communication in large organizations. Chapter 2, *Agile in large organizations*, gives an overview of the agile practices and principles and the application of agile methods in small and large organizations. An exemplary process of injecting agile methods in large enterprises is developed. Based on the experiences gathered during the generation and introduction of this process, resultant problems of an integration of agile methods and practices in large organizations with different processes are described.

In Chapter 3, *Communication and Meetings*, we illustrate the significance and characteristics of communication in development projects. The externalization of information with the STACHUS framework is based on a meeting taxonomy and a meeting grammar. The meeting taxonomy classifies the meetings and the meeting grammar defines the meeting elements of the meeting taxonomy, used in the protocols.

Chapter 4 describes the *protocol generation technologies* regarding summarization of spoken conversations and protocol automatism. Speech recognition technologies and techniques for the summarization of texts and information externalization, respectively, are introduced and analyzed for the applicability in the STACHUS framework to extract information from a meeting and automatically generate a protocol.

Chapter 5 introduces the model for automated protocol generation. A specific meeting compiler and engine are described, as well as a grammar for protocol generation. The protocol engine generates the meeting minutes out of the recorded meeting audio stream, based on the grammar. Both components are generated by the meeting compiler.

The applicability of the STACHUS framework is validated in chapter 6 in several student projects. The transition to software development independent communication is also described in this chapter. During the empirical evaluation of our hypotheses the effort reduction and accelerated publication of automated protocols by STACHUS are analyzed. Additionally, the usability and training effort, i.e. the 'easiness' of the grammar for the meeting participants is evaluated, as well as the completeness and correctness of the externalization of meeting information.

Chapter 7 summarizes the dissertation and provides an outlook of future work in particular in the area of speech recognition and automatic report generation, to improve the protocol quality and enhance the STACHUS framework.

The dissertation contains four appendices. A project example of the STACHUS grammar is summarized in *Appendix I*. *Appendix II* covers aspects of the realization of the STACHUS prototype. A short description of the empirical research methods used throughout the dissertation is given in *Appendix III*. The questionnaire for the evaluations used in the case studies is presented in *Appendix IV*.

2 Agile methods in large organizations

Combination of empirical and defined process models

This chapter gives a brief overview of the fundamental ideas of agile methods. Section 2.1 provides a historical overview. Afterwards the application of agile method and principles in large and traditional, plan-driven organizations is analyzed (section 2.2). The integration of empirical processes in a defined process world is challenging. The challenges are discussed in section 2.3, which leads to the research questions, addressed in this dissertation.

2.1 Agility

In the late 1990's software development methodologies began to get increasing public attention, that emphasized an empirical process model (Takeuchi & Nonaka, 1986), close collaboration between the programmer team and business experts, face-to-face communication, frequent delivery of new deployable business value, and tight, self-organizing teams.

The word *agile* was selected to describe the intention of these methodologies as a lightweight method, as the waterfall process was seen as heavyweight and bureaucratic (agilecollab, 2008). The accepted definition of agility was summarized in the Agile Manifesto (Beck, et al., 2001) by software process methodologists in 2001, where agility is defined as the ability to both create and respond to change. Core to agile software development is the use of light human- and communication-oriented rules. Agility is value-driven in the focus on delivering the most important features first, i.e. the highest value items are developed first on what will be used the most by the customer. The development is done with small, dedicated, co-located, self-organizing teams who work in close collaboration with a business customer (Slinger & Broderick, 2008).

The agile methods are characterized as a subset of iterative methods. Iterative development is an approach to build software where the overall lifecycle is composed of several time-boxed iterations. Each iteration is a self-contained mini-project composed of activities such as requirements analysis, design, programming, and test. The goal for the end of an iteration is a release, a stable, integrated, tested and executable system with new features, not a proof of concept (Larman, 2004). This enables flexibility in the further process of the project, because decisions can be made based on working software. The system grows incrementally with new features, iteration by iteration. This concept of growing an always executable system via iterations is at the core of all the agile methods.

In iterative development the requirements, plans, estimates, and solution evolve or are refined over the course of the iteration, rather than fully defined and "frozen" in a major up-front specification effort before the development iterations begin.

Adaptive development implies that the requirements, plans and solutions adapt in response to feedback from prior work and feedback from the users. The intent is the same as iterative development, but the name suggests more strongly a feedback mechanism.

Some methods or methodologies emphasize the term "evolutionary" instead of "iterative" or "adaptive". The ideas and intents are similar, although strictly speaking, evolutionary development does not require the use of timeboxed iterations, where the schedule is divided into a number of separate, but fixed time periods. Moreover, in adaptive development, it is not the case that the requirements are forever unbound or always changing at a high rate. Rather, most requirements discovery and refinement usually occurs during early iterations (Highsmith J. A., 1999).

Agile development methods apply time-boxed iterative and incremental development, adaptive planning, and encourage the rapid and flexible response to change.

Scrum (Schwaber & Beedle, 2002), eXtreme Programming (Beck & Andres, 2004), Feature Driven Development (Nebulon Pty. Ltd., 2002) and Lean Software Development (Poppendieck & Poppendieck, 2003) are the most established agile methods. Each agile methodology has its particular focus. Scrum is a project management framework that forces on dedicated, small, cross-functional, self-managed teams to build increments of product. A set of Backlog features is realized in 30 days, so called Sprints cycles. The core practice in Scrum is the use of daily 15-minutes team meetings for coordination and integration. There are three roles in Scrum: a Product Owner who is responsible for the success of the product, a delivery team who is charged with creating a potentially shippable product increment every 30 days, and a Scrum Master who facilitates communication between the Product Owner and the team and removes obstacles encountered during development.

eXtreme Programming (XP) is a software development methodology, i.e. mainly focused on software development, described with a set of 12 engineering practices that embody and encourage the values of communication, feedback, simplicity, courage, and respect. The practices continuous integration, test-driven development, and pair programming for instance, help the development team stay responsive to the customer's needs and have always working software. XP teams are small (no more than ten) and co-located. Their practitioners embrace the fact that product requirements change, and change for good reason, while working in iterations alongside the customer to capture and implement the requirements for immediate feedback.

Feature-Driven Development (FDD) combines the advantages of agile methodologies with model-driven techniques. FDD focuses on the domain model, the creation of which is the foundational step in the FDD process. The five activities to be followed in FDD are: develop a domain model, build a feature list, plan by feature, design by feature, and build by feature. Sets of features are worked through the completion in two-week iterations. The features to be built are small aspects of client-valued functionality.

Lean Software Development was adapted from Lean Manufacturing, the Toyota Production System, and Bob Charette's Lean Development (Charette, 2002). Lean Software Development focuses on seven principles: eliminating waste, amplifying learning, deciding as late as possible, delivering as fast as possible, empowering the team, building integrity in, and seeing the whole. A set of tools is provided to help teams adhere to the principles and achieve their goals. Lean is a management approach for streamlining the process of providing value to the customer, complementing the existing practices of software development teams.

The next section gives an overview of agile methods applied in large and traditional environments and the related challenges, when the agile methods are adapted to large teams and integrated in defined processes models. Another paragraph introduces an existing agile process model at a large organization that combines defined and empirical process control methods.

2.2 Agile methods in large and traditional organizations

Although Beck (2004) and Schwaber (2002) recommend teams of seven to ten, collocated people, including Product Owner or other customer proxy, developers, and testers, when applying an agile method, larger and distributed teams are also trying to follow the agile practices and principles (VersionOne, 2009). Several best-practices have been proposed to apply agile methods also in large teams, summarized by Eckstein (2004) and Pichler (2008).

Eckstein recommends for instance a jointly *review meeting* to offer all team members a common understanding of the project progress and cross-project *retrospectives* to facilitate project-wide learning, as well as the interaction of the project members across team boundaries. Eckstein distinguishes between several types of project-wide sprint retrospectives, e.g. meta-retrospective or open-space retrospective. Open-space retrospectives are a mode, where team members propose topics, which will be executed and discussed afterwards in parallel groups. The results, changes, and next steps are finally summarized. In meta-retrospective, first all teams have a retrospective and afterwards representatives of each team meet in a meta-retrospective.

Meta-Scrums propose a similar paradigm that enable cross-project self-organization and help to identify project-wide impediments systematically (Sutherland, 2005). The *scrum of scrums meeting* is a recommended technique in scaling Scrum to large project teams, where each team meets daily and then sends a representative to meet with the other representatives.

Pichler recommends a consistent estimation basis, i.e. a common understanding across the different teams for example of story points, an estimation basis of the complexity of a work package. Moreover, guidelines and norms, e.g. coding guidelines, will help the teams to build a consistent and high quality software product.

Also in large projects, the teams should be collocated and not split across several locations.

An adequate communication *infrastructure* helps distributed teams to enable cooperation, including a central code basis, as well as central project storage (e.g. Wikis), configuration management, integration- and build management (managed by an integration team), and tools for collaboration. Also a large team has to be able to deliver frequently a running system for customer feedback, for instance via individual and collective integration.

However, agile methods are currently not only attempted to be adapted to large teams, but also to be integrated in defined processes models with the aim to address the problems of a traditional, waterfall software development approach.

The Standish Group regularly publishes the CHAOS Report, containing statistics about success and failure of software development projects. In 2009 for example, 68% of the analyzed projects were late, over budget, or delivered with less than the required features and functions, or even cancelled prior to completion (Standish Group, 2009). In another study of 6700 projects, four out of the five key factors contributing to project failure are associated with and aggravated by the waterfall model, including inability to deal with changing requirements and problems with late integration, resulting in a risky and expensive way to build software systems (Larman, 2004, p. 75). Due to misconceptions between the customer and the developers, or an unclear vision of the customer regarding the finished product, requirements change. The cost of change grows through the software's development life cycle, as illustrated in Figure 2-1.

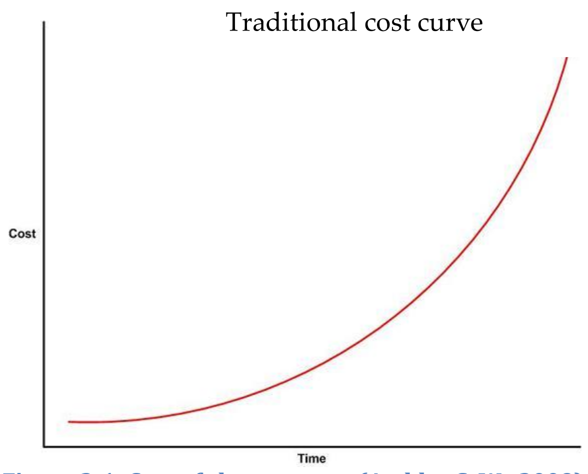


Figure 2-1: Cost of change curve (Ambler S. W., 2009)

Change in requirements and technology solutions, as well as environmental changes, cannot be stopped early in a project, but must be handled throughout the project life cycle (Highsmith &

Cockburn, 2001). Because these changes cannot be eliminated, driving down the cost of responding to them is the only viable strategy. Rather than eliminating rework, the agile strategy is to reduce cost of rework. This is done not only by the development in iterations, where no big up-front specification is written, but also by prioritizing the customers' demands and developing the most important features first, where the feature is analyzed, developed and tested in one iteration.

For this reason, large organizations are investigating agile methods to deal with change and reduce the cost of change throughout a project (VersionOne, 2009). The motivations for the increased adoption of agile methods are in addition an expected increase in productivity and time to market, to reduce cost, and improve the software quality (Schwaber C. , 2005). As a result agile methods are introduced in traditional environments. The question arises, how much change is necessary when transitioning to and using agile methods. Development groups are trying single practices, for instance only pair programming, or only test-first development, and attempt to determine how to get any of these practices to work with internationally distributed teams. The agile philosophies and practices disperse over a wider range of people, projects, and organizations, including many CMM and ISO 9000 organizations (Canditt & Russwurm, 2008), (Glazer et al., 2008).

The adoption of agile development practices means change (organizational change, cultural change, changing tools and processes), which is easier for small projects in small organizations than for large organizations (Schiller & Canditt, 2008). Change is made difficult, due to the defined process world of large businesses and the defined interfaces between departments, as the other departments have to change, too. Moving an enterprise to agile methods means to substantially revolutionize and evolve the methods, the organization, best practices, and even the company culture. Changing paradigms, e.g. from big up front planning to continuous planning and responding to change, or from the management methodology command and control with top-down decision making to collaborative and self-organizing teams, provide both the power and the concerns for agile, because addressing change on such a wholesale basis in an enterprise is challenging (Leffingwell, 2007), as it takes time, is possibly complex and costly, and people have to be convinced to change.

Chung and Drummond (2009) argue that agile methods can be scaled to enterprise level. The scalability of agile methods and introduction in a process-driven environment of a large international organization was evaluated during a case study by Schiller and Canditt. During the case study an agile process, called 'agilePEP' (agile product evolution process), was integrated in the existing, traditional defined process world of a large company, the Siemens AG. The results of this case study are confirmed by evaluation in other large organizations, for example at Allianz AG (Hastreiter, Roberts, & Mathis, 2009) .

The Siemens Reference Process House (RPH) (Schmelzer & Sesselmann, 2007) generically defines all (traditional) software development processes within the Siemens group. The idea is to harmonize the evolved agile processes of small teams within Siemens and integrate them in the RPH to create a process that is standardized and mandatory for the rising number of agile teams of one business branch and suitable also for large agile teams. agilePEP is an empirical process, based on the Scrum methodology that is integrated in the RPH at Siemens Building Technologies (MRT PLM Group Europe, 2007). This process is applied in several agile teams at Siemens (in the USA, Switzerland and Germany), signifying that there are still challenges when combining agile methods with traditional processes, as presented in the following.

While integrating an agile process in an existing, defined process world, observations and interviews at Siemens discovered that agile team members, who are convinced of the benefit of agile methods, believe that the agilePEP process is no more pure agile, it is too 'traditional' due to the required documentations, e.g. fixed specifications and project reports. Whereas traditionalists, i.e. the team members who want to continue the waterfall-approach for software development, and now have to follow the agile principles, are missing many quality assurance documents, detailed specifications and upfront planning. That is, one of the biggest issues, when introducing agile methods to a traditional organization, is the organizational *culture change*, as defined by Schein (2004, p. 444).

Beside the problem of organizational culture change, several other challenges accompany the transition to an empirical process, for example customer availability, change of the physical working environment, or milestone definitions, which will briefly be presented in the following.

Organizations grow a strong *corporate culture* over time, some of which may not be beneficial to agile. Leffingwell (2007) claims that if the accomplished work is measured by hours worked rather than by productivity delivered for example, agile teams will not necessarily feel that they are properly motivated, measured, or rewarded. As agile is tactically intensive, and because of the daily and weekly focus and accountability, the agile manifesto requires that agile processes promote sustainable development (Beck, et al., 2001). That is, if more than a 40-hour workweek in the long run is demanded, agile development will not be the proper choice.

Compensation systems may be poorly designed as well. Systems that reward individual over team performance may prevent the team from the pairing and cooperation necessary to achieve an iteration. Everyone is accountable for his or her own work, but everyone is accountable to the team as well, and reward systems recognize the differences. Moreover, strict command-and-control cultures inhibit agile practices. If management dictates all processes and technologies, the teams will not be able to evolve to the self-organizing and constantly changing teams that characterize the agile methods, nor will they be able to select the optimum technical path to a solution if they are directed by others to a different approach.

The success of the agile methods is dependent on close *customer collaboration*, where the customer is available for questions and regular reviews and the business analyst who works with the project team is willing and able to participate in the fine-grained, detailed review that stories and customer-written acceptance tests require. However, often the customer may be remote or may not have the skills or time available to participate in such a manner; or there is no single customer, as the application has tens of thousands of users. Then a present product manager has to play the proxy role, acting on behalf of the customer, answering questions, and making decisions in cooperation with the customer.

Agile's practice of working on a few stories at a time is a focusing mechanism for the team. But in larger systems, it has to be clarified what the right stories are and if the summation of all stories actually meets the customer's end-to-end use-case needs. That is, the *lack of requirements analysis and documented specifications* has to be solved. A clear visibility into stories others are building has to be assured and if and how they affect the team. When developing solution set (large sets of products that must be deployed together and support end-to-end use cases for the user or customers), it has to be ensured that the stories actually work together to achieve the final objective.

XP and Scrum recommend *cross-functional teams of seven people*, plus or minus two (Schwaber & Beedle, 2002). To an enterprise with 1000 practitioners it has to be clarified how to fit this new model into the existing organizational hierarchy.

Much of the productivity of agile methods comes from the pairing contexts, daily stand-ups, visual signaling of stories and status, and constant informal communication that characterizes these methods. Developers, product owners, and testers are together, not separated by time zones and language barriers. At scale, *collocation* is impractical even for large teams in the same building and other mechanisms must be devised. It is likely that many team members are in different countries and different time zones and perhaps even speak different languages. At scale, all development is distributed development, and the methods and organization must adapt to this challenges.

In agile projects team members work in an *open environment*, and often their managers and supervisors are joining the team at those tables. Additionally, the informality of stories posted on walls, and teams of people that are far more talking than coding are suggestive of inefficiency, unsupervised, chaotic, and even unprofessional project work. The traditional team members first have to familiarize with these changes in the development practices.

As organizations grow, they tend to put in infrastructure to control and measure projects and programs. These organizations often become the drivers behind *formalized policies and procedures* for development, to standardize and harmonize their processes and thus reduce the related risks and training costs. In addition, outside audits by regulatory agencies and customers such as drug companies may look for typical 'controls' on the development process, such as "requirements document complete and signed off" or "test plans complete and approved by QA". These are artifacts that the agile teams don't need und are unlikely to be developed, unless mandated. Agile methods are a concept of a 'barely sufficient' methodology that attempts to answer the question of how much structure is enough, meaning that the demands regarding the processes, documentation, etc. are as less as possible. Many organizations operate on the unspoken assumption that if a little process is good, then lots of process will be better (Highsmith

J. , 2002, p. 5). Nevertheless, the key is having good people – good domain experts, good developers, good chief programmers, because no process makes up for a lack of talent and skill (Highsmith J. , 2002, p. 6).

Individual *control mechanisms* are added by each team from their experiences, gathered in past projects. It is quite common to see stage gates such as "design complete" and "design review sign of" in the published product and system development *process guidelines*. These formalized, published, and adopted guidelines are not so easy to change. And many of these policies and procedures must be amended, changed, or eliminated to facilitate agile. The documented policies cannot simply be ignored, and they can create real impediments for agile adoption.

If a customer comes to a team to deliver functionality X in period Y with resources Z, then by definition it does not meet agile's *fixed-time, variable scope* idea, and the teams will be discouraged right out of the starting gate. This impediment will have to be addressed directly. Teams would like to be able to predict exactly when they can deliver what functionality, but they know they can't.

It is also likely that the teams responsible for developing the product and their extended teammates in marketing or operations distribution, had problems in the past, e.g. delayed, incomplete or incorrect deliveries. For many outside the development teams, it appears that the development organization has simply "failed to deliver again", and mistrust is the result. For those in development, it appears that their outside stakeholders don't understand that it is research and development, not just development. Neither team is right or wrong, but with agile methods, they will be forced to work together in close proximity. As the agile manifesto demands: "Business people and developers must work together daily throughout the project" (Beck, et al., 2001), they must relearn to *trust* each other's skills and contributions.

Enterprises are often organized along *functional* (product management, architecture, development, etc.) rather than product or *business application lines*. In agile methods, teams quickly reorganize to assure they have a full complement of the resources necessary to define, build, test, and deliver a component or feature. This requires dedicated (not heavily multiplexed) resources for the project, or else the team will fail to meet its commitment to the iteration. Reorganization typically requires redefinition of what makes a team a team in the enterprise.

There are often unclear and different understandings regarding *milestone* definitions in the project lifecycle. Especially the transition from definition phase to implementation phase is a problem, as traditionalists are used to get at that point the list of to be realized features and their specification, while this "list" exists in agile not until the end of the project, due to changing requirements or changed prioritizations.

The organization has grown with its successes. For most enterprises, success often involves acquisition of teams or product lines or existing IT organizations along the way. These teams are rarely *collocated*, and the larger the enterprise, the less likely they are to be together in one place. Moreover, raw size alone prevents pure collocation because there is no way physically to co-locate even 100 people in one workspace, so the problem of distributed teams is endemic to agile methods at scale.

As agile methods were originally developed and codified in small team environments, there was freedom to explore and innovate, and most resources the teams needed and most problems that arose could be managed at the level of a single team. It has to be recognized that many successful applications of these methods at larger scale occurred in a context where there were many small and autonomous projects inside the enterprise. So specific projects aligned well with the principles of agility and most of the rest of the enterprise was not dependant on components, subsystems, features, or anything else these teams were delivering. These applications could have been smaller, stand-alone products, or internal applications, or perhaps new web front ends for legacy applications; but in any case, they were relatively isolated endeavors that did not require coordination of large numbers of people or other teams and other departments. The basic small-team constructs of agile methods were inherent attributes of the project, and the organization did not interfere with success. However, when building enterprise class systems, systems of systems, and applications, including components and enterprise systems provided by others, the apparent limits of the methods themselves must be analyzed, as well as the challenges regarding the organization. Barriers to agile adoption at the enterprise arise from two sources: the apparent limitations of the methods themselves and the impediments that exist within the enterprise. Both must be addressed to achieve enterprise agility (Leffingwell, 2007, p. 87). The limits of the agile methods include small team size and collocation, close customer involvement, and missing project documentation. Cohn and Ford (2003) see limits given by the organization in change resistance, the adherence of standards (e.g. FDA (U.S. Food and Drug Administration, 2009), or CMMI (Carnegie Mellon University, 1991)), and established processes.

Especially a lack of documentation, including specification documents, meeting protocols, review protocols, agendas, etc., is critical. In a traditional company it is common to write these documents and reports, which a traditional quality assurance department demands, while most agile teams see no value in it. There are no meeting minutes for informal meetings, as they are often short, no minutes taker is assigned, or they take place for instance in the hallway where there is no possibility to take notes. Agile projects are focused only on the current project and emphasize tacit knowledge and communication, project and process documentation is not necessarily written down to avoid waste (Poppendieck & Poppendieck, 2003) if they are not part of the project deliverables like manuals, or backlogs. The team lives the process daily, meets, discusses about the process, and adapts it. Decisions, information exchanges, specifications and clarifications, etc., which emerge out of a meeting, are only interesting for the life of the project and haven't to be written down, as long as they are present for the team members.

This leads to the danger of losing important information, e.g. regarding the process, if the tacit information is not externalized and saved. Furthermore, knowledge reuse and companywide learning (CMMI Product Team, 2006), especially across projects, are not covered by agile methods. These challenges have to be addressed, when the agile methods are integrated in a large environment with defined processes.

2.3 Research question

This dissertation addresses the problem of missing externalization of tacit project knowledge, especially in an agile software development environment. Today, as the number of informal meetings is rising due to the increased application of agile methods, the process of protocol generation becomes even more critical.

Although there are best practices for meetings of traditional project teams of writing a protocol after the meeting (see chapter 3), they do not deal well with many problems. The creation of meeting minutes is time consuming and it is difficult for the minutes taker to listen and write at the same time. So the protocols are often published too late, incomplete or incorrect.

Our suggestion is to automate the generation of protocols to record the project knowledge, so the software development teams don't have to create the documentation manually.

Our hypothesis is that with an automated protocol generation process, the flexible and communicative character of agile can be preserved. Moreover, the effort for the creation of meeting documentation is minimized and the quality of the documents will improve, as well as the protocol generation is accelerated, which is helpful also for traditional meeting environments. When applying the automated protocol generation tool, the easiness in communication is ensured, that is, the meeting progress is not interfered, and no additional effort for the tool handling, as well as an easy usability of the tool is guaranteed. So the organizational memory – the accumulated data, information, and knowledge summarized in documents, or knowledge in individuals' memories and in procedures and products – is supported by an externalizing knowledge management, which additionally improves the effectiveness of meetings, as action items, etc. are not forgotten by the next meeting.

We will show that it is possible to combine the advantages of documentation as externalized knowledge source with the lightness of agile by the implementation of a protocol generator engine. The protocol generator will be part of a dynamic extensible meeting management and protocol generation framework. The correctness of our hypotheses will be evaluated in two case studies in the academic field and within an industry survey.

In the next chapter we will show the relevance of communication for the software development process in large agile projects. The forms of communication, especially the difference between document-based communication and face-to-face conversation in meetings is presented. Moreover, we analyze how information can be externalized from meetings.

3 Communication & Meetings

The relevance of communication in today’s projects

This chapter elaborates the relevance of communication for software development projects, especially regarding knowledge management (section 3.1). Section 3.2 analyzes communication in meetings in detail, presents a meeting taxonomy, and discusses meeting minutes as one form of externalizing and conserving the tacit knowledge of meetings.

3.1 Communication

Communication is the reciprocal exchange of thoughts and information via speech, gesture, mimic, writings, or pictures of at least two participants. It is a technique for expressing ideas effectively (Oxford University Press, 2010). Communication is a process by which information is transmitted between individuals through a common system of symbols, in an attempt to create shared understanding, spread information, and generate knowledge.

The process of information transmission can be described with a simple model, developed by Claude Shannon and Warren Weaver, where information is sent from a sender to a receiver (Shannon, 1948). This model views communication as a means of sending and receiving information. The strengths of the model are simplicity, generality, and quantifiability. The model is structured based on an information source sending a message to a destination. The information source produces a message, which is encoded into signals, i.e. the sound of the spoken word. The signals are transmitted via the channel. The message is then 'decoded' (reconstructed) from the signal and arrives at the destination.

Wilbur Schramm refined and expanded Shannon’s model to a two-way circular communication (allowing feedback) between the sender and receiver (Schramm, 1961). This model of communication is illustrated in Figure 3-1.

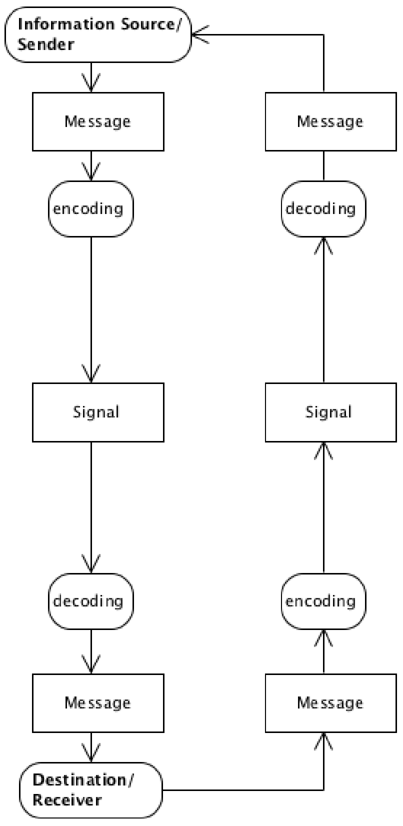


Figure 3-1: Communication model (UML Activity Diagram)

The communication model visualizes the key abstractions of communication: *sender* (by whom), *receiver* (to whom), and the *message*. The sender (source) conveys an idea or some information to the receiver. The message is encoded, send, and the transmitted information is then decoded. The receiver is the destination or target of the message.

Messages are characterized by a *purpose* or pragmatic aspect, the *content* (what is communicated), its *form*, and the *method* (through which medium). The sender transmits a message with the intention of knowledge transfer. Objectives of communication are to discuss open questions, to make a decision, or to check the status of previous action items, as well as to distribute information, gather ideas, or to develop a solution (Brügge & Dutoit, 2004). Communication accomplishes cooperation and innovation, establishes confidence between the team members and the communication flow reduces internal and external response times.

The actual content is the information that is conveyed. This may include facts and figures, a project status update, a summary of recent meetings, or a key project decision.

People communicate with each other in a variety of ways that depend on the message they want to send and the context in which it is to be sent. There are several forms of communication, which can be categorized according to the following criteria. First, there is *written* and *verbal communication*. According to the number of participants, communication can be divided into *one-to-one communication*, i.e. interpersonal communication between two individuals, *one-to-many communication*, i.e. presentational communication, where one person communicates to a group, and *many-to-many* communications. Communication can be *personal* or *technique supported*. Personal communications are for instance face-to-face meetings, whereas in technique supported communication the transmission of information is based on the application of technique, for example via e-mail.

Synchronous and *asynchronous communication* can be differentiated. Synchronous communication comprehends for example hallway conversations, structured interviews, meetings (face-to-face, telephone, video), or synchronous groupware where the participants communicate at the same time, whereas electronic mail, newsgroups, and the World Wide Web are asynchronous communication (Brügge & Dutoit, 2004), where the response is delayed.

Each of these forms of communication can involve *one-* or *two-way communication*, i.e. either there is only one sender and one receiver, or the receiver becomes also a sender to give feedback.

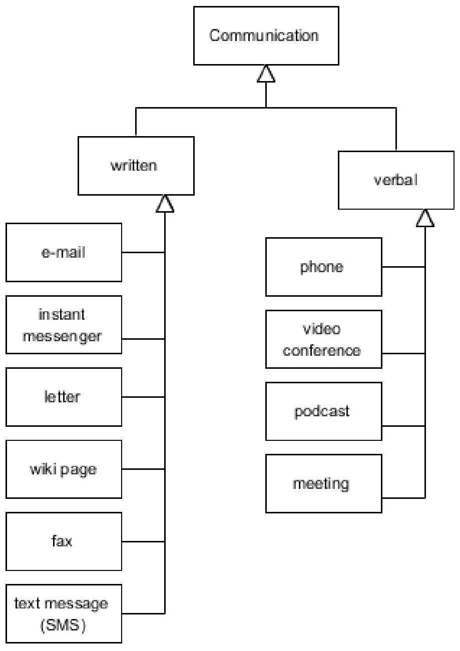


Figure 3-2: Ways of communication: written and verbal

There are multiple *ways of communication* (communication channels) – via phone, email, voicemail, instant messenger, letter, fax, meeting, video conference, audio conference, desktop sharing, text message (SMS), podcasts, video casts, logs, wikis, and more. The methods are distinguished, whether the method promotes a *push approach* that is, information is pushed out to the receiver, as it is via email or presentation, or a *pull approach* in which the person who needs information asks for or collects it when it's needed, as it is the case with blogs or notice boards, for example. Some methods support different forms of communication better than others. The effectiveness of a way of communication defines how well the communication channel is suited to support the information transmission. For instance, e-mail is an effective method for one-to-one and one-to-many communication, but inefficient for many-to-many discussions or conversations.

The richness of a communication channel is defined along a continuum, where highly rich channels as those handling multiple inherent cues simultaneously, such as using feedback, nonverbal gestures and mimic, and several senses simultaneously. A face-to-face meeting, which employs feedback as well as audio and visual senses, is considered extremely rich. On the contrary, a newsletter or fax is lean, however, involving only the visual sense and slow or no feedback.

The empirical process control method is a *communication based process*. That is, communication is in the foreground and controls and influences the software development process. In conversations, the progress of the project is determined. That is, for example in dialogues with the customer, requirements are gathered. The requirements are planned and estimated in planning meetings, and the next features to be developed are prioritized. In daily meetings, the current status is shared, or during pair programming sessions problems are discussed. There are review meetings with the customer to present the current status. In retrospective meetings the software development process is reviewed, to detect problems and gather ideas and solutions for improvement.

For each of these meetings, an agenda can be created. During the meeting the next steps are defined, which flow into the agenda of the next meeting.

3.1.1 Communication and Knowledge

Knowledge is defined as the expertise and skills acquired by a person through experience or education, i.e. via communication (Oxford Corpus, 2005). It is the theoretical or practical understanding of a subject with the ability to use it for a specific purpose if appropriate.

Data, *information*, and *knowledge* have to be distinguished. Examples of data are raw numbers and facts; information is processed data, and knowledge is applied and verified information. Knowledge is information possessed in the mind of individuals: it is personalized information, related to facts, procedures, concepts, interpretations, ideas, observations, and judgments. That is, information is converted to knowledge once it is processed in the mind of individuals and knowledge becomes information once it is articulated and presented in the form of text, graphics, words, or other symbolic form.

In an organizational context, knowledge is the sum of what is known and resides in the intelligence and the competence of people and can provide a competitive advantage/ resource (Nonaka & Takeuchi, 1995). Knowledge is created by individuals. Knowledge can be viewed as existing in the individual or the collective (Nonaka, 1994). Individual knowledge is created by and exists in the individual whereas social knowledge is created by and inherent in the collective actions of a group, summarized in a company knowledge repository. Thus, organizational knowledge creation is understood as a process that "organizationally" amplifies the knowledge created by individuals and crystallizes it as a part of the knowledge network of the organization.

Nonaka distinguishes two dimensions of knowledge in organizations: tacit (or implicit) knowledge and explicit knowledge (Figure 3-3). The tacit dimension of knowledge is rooted in the action of an individual, the experience, and involvement in a specific context. Tacit knowledge represents internalized knowledge that resides only with the individual e.g. subjective insights, intuitions, and hunches, as well as ideas, values, or emotions. The explicit dimension of knowledge represents knowledge that is articulated, codified, and communicated and shared in symbolic form and/ or natural language. The most common forms of explicit knowledge are manuals, documents and procedures, as well as audio-visual material.

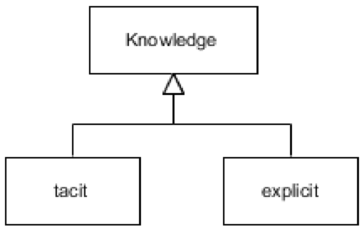


Figure 3-3: Two dimensions of knowledge

Explicit knowledge can be processed by a computer, transmitted electronically, or stored in databases. But the subjective and intuitive nature of tacit knowledge makes it difficult to process or transmit the acquired knowledge in any systematical or logical manner. For tacit knowledge to be communicated and shared within the organization, it has to be converted into explicit knowledge.

A further categorization of knowledge is the distinction between the creation of "new knowledge" (i.e., innovation) vs. the transfer or exploitation of "established knowledge" within a group, organization, or community.

Knowledge management refers to identifying and leveraging the collective knowledge in an organization to help the organization compete. Moreover, it is intended to increase innovativeness and responsiveness, improve performance, share lessons learned, and to assure integration and continuous improvement of the organization (Thompson & Walsham, 2004).

Knowledge management involves the processes of creating, storing/retrieving and transferring knowledge.

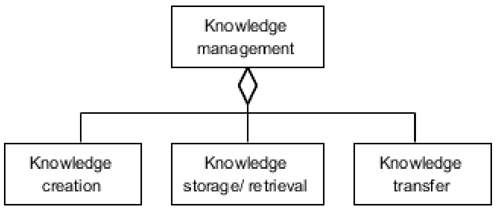


Figure 3-4: Knowledge management and its processes

Organizational *knowledge creation* involves developing new content or replacing existing content within the organization's tacit and explicit knowledge. Through social and collaborative processes as well as in individual's cognitive processes (e.g. reflection), knowledge is created and shared in organizational settings. Four modes of knowledge creation have been identified by Nonaka and Takeuchi: socialization, externalization, combination, and internalization (known as the SECI process) (Figure 3-5). Each mode is based on conversation to create new knowledge. In the socialization mode (tacit to tacit), new tacit knowledge is created from tacit knowledge through conversations, social interactions, and shared experience among organizational members. Externalization (tacit to explicit) refers to converting tacit knowledge to new explicit knowledge (e.g. articulation of best practices or lessons learned in a meeting and writing them down). The combination mode (explicit to explicit) refers to the creation of new explicit knowledge by merging, categorizing, reclassifying, and synthesizing existing explicit knowledge (e.g. literature survey reports). In the internalization mode (explicit to tacit), new tacit knowledge is created from explicit knowledge (e.g. by learning and understanding results from reading or through discussions). The four knowledge creation modes are not pure, but highly interdependent and intertwined. That is, each mode relies on, contributes to, and benefits from other modes.

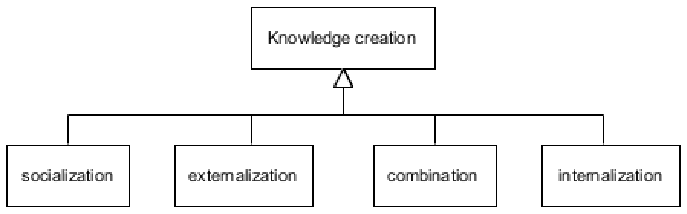


Figure 3-5: SECI: socialization, externalization, combination, and internalization

The *storage* and *retrieval* of organizational knowledge, also referred to as organizational memory, constitutes an important aspect of effective organizational knowledge management. Organizational knowledge includes knowledge residing in various component forms, including written documentation, structured information stored in electronic databases, codified human knowledge stored in expert systems, documented organizational procedures and processes and tacit knowledge acquired by individuals and networks of individuals (Tan, Teo, Tan, & Wei, 1998). The organizational memory has to be stored, however in such a way that an effective and efficient retrieval of information is allowed.

The *transfer* of knowledge occurs at various levels: transfer of knowledge between individuals, from individuals to explicit sources, from individuals to groups, within group, across groups, and from the group to the organization (Alavi & Leidner, 2001). An important process of knowledge management in organizational settings is the transfer of knowledge to locations where it is needed and can be used. Communication processes (dialogues, discussions, and experience sharing) and information flows drive knowledge transfer in organization.

Successful knowledge management effort needs to convert internalized tacit knowledge into explicit knowledge in order to share it – that is from individual's personal to organizational knowledge. Communication can be seen as processes of information and knowledge transmission. Through verbal and written communication knowledge and experiences are conferred. Emergent knowledge is generated through communication. For instance, during a meeting, the SECI process pushes knowledge creation, as the tacit knowledge of the individuals is externalized and then combined to new knowledge. Furthermore, externalized knowledge, for example a book, can be the source for new knowledge generated by internalization.

The dynamic model, called knowledge conversion process, considers a spiraling knowledge process interaction between explicit knowledge and tacit knowledge(Nonaka & Takeuchi, 1995). This model views organizational knowledge creation and transfer as involving a continual interplay between the tacit and explicit dimensions of knowledge and a growing spiral flow as knowledge moves through individual, group, and organizational levels. In this model, knowledge follows a cycle in which implicit knowledge is 'extracted' to become explicit knowledge, is, where applicable, stored, and the explicit knowledge is 're-internalized' into implicit knowledge.

Communication and learning are important activities in software development processes. Project relevant knowledge regarding requirements, agreements, the system models, the software architecture, the design, and tools and used technologies have to spread (via communication processes) around the team. If all team members are informed, i.e. the group memory was transferred to the individual memories, each individual team member can make a decision against the background of the current information state and regarding the made agreements amongst each other and with the customer in consistence with the current project situation.

3.1.2 Communication: conversation vs. documents

While agile projects emphasize people, personal communication, and tacit knowledge, traditional methods emphasize business process engineering, explicit knowledge, and document based communication. In this regard, the agile manifesto promotes: "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."(Beck & al., 2001) An agile project requires open communication and collaboration in order to succeed. Davenport and Prusak (1998) even claim that companies need to shift attention from documents to discussions, as people can transfer ideas faster by talking face to face than by writing and reading documents.

Software development projects proceed more successfully, if more importance is attached to the developers and their communication among each other, as well as to working software, instead of comprehensive documentation and a rigid adherence of a process (Beck & al., 2001). The communication focus allows an agile project to react fast and flexible to changes.

There are various modes of communication that people may choose to apply when working together. Figure 3-6 shows a graph that compares the effectiveness of different ways of communication with the richness of the communication channel employed.

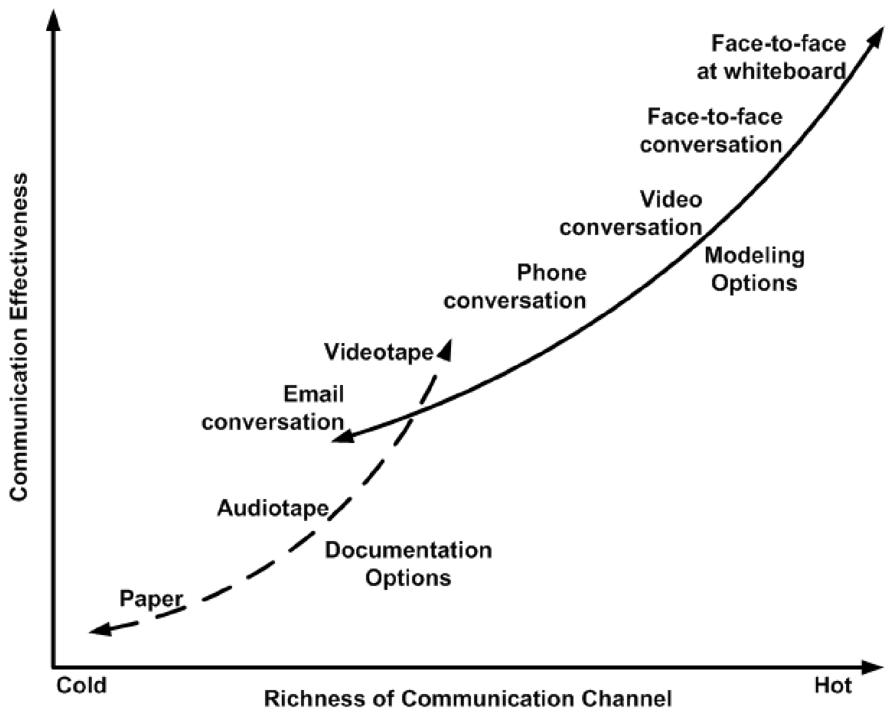


Figure 3-6: Effectiveness of different ways of communication (Ambler, 2009)

Ambler claims that the most effective communication is face-to-face, particularly when enhanced by a shared medium such as a plain whiteboard, flip chart, paper, or index cards. By removing the shared medium or by no longer being face-to-face, a drop in communication effectiveness can be experienced. As the richness of communication channels decrease physical proximity is lost and the conscious and subconscious clues that such a proximity provides. The benefit of multiple modalities is lost, i.e. the ability to communicate through techniques other than words such as gestures and facial expressions. The ability to change vocal inflection and timing is also lost, people not only communicate via the words they say but how they say those words. A speaker may emphasize what he is saying, i.e. changing the way he is communicating, by speeding up, slowing down, pausing, or changing tones. Furthermore, the ability to answer questions in real time, the point that distinguishes verbal and written communication, are important because questions provide insight into how well the information is being understood by the listener (Cockburn, 2002).

Given the high levels of corporate loss of knowledge in commerce and industry, organizations not only have to transfer knowledge, they have to preserve their organizational memory and, in particular, their tacit knowledge. Tacit knowledge of individuals or small teams has to be made available for other and future teams of the organization. This is done by extracting tacit knowledge in an easily accessible format – like documents. Thus, documents constitute a source of knowledge in a rapid changing environment, provide a basis for decision making, and allow traceability.

Understanding is generated by a combination of documentation and conversation. Documentation can provide content (facts), but conversations are better at building context. So documentation as well as conversation is need. However, while documentation contributes to understanding and knowledge transfer (current and future), it also creates barriers to conversation.

Even though most organizational work processes are largely designed around documentation, much remains unrecorded, especially regarding decision-making.

We have seen in this section that the most effective form of communication is conversation, particularly in meetings (personal, via telephone, video conference, etc.), whereas the discussed information and tacit knowledge has to be summarized and stored as explicit knowledge in documents (meeting minutes). In the following, we will go into the details of meetings, their various forms and their meaning for a successful project completion.

3.2 Meetings

Meetings are one of many coordination mechanisms available to organizations, as discussed on page 18. In a meeting, two or more people come together for the purpose of discussing a (usually) predetermined topic, often in a formal setting. Doyle and Straus (1976, p. 3) even claim that "we are a meeting society - a world made up of small groups that come together to share information, plan, solve problems, criticize or praise, make new decision or find out what went wrong with old ones." In addition to coming together physically, communication lines and equipment can also be set up to have a discussion between people at different locations, e.g. a conference call or an electronic 'e-meeting'.

In the following we use a dynamic model for a meeting, which consists of three phases: meeting set- up, run the meeting, and a meeting follow-up.

During the meeting *set-up phase*, the meeting is prepared, that is, purpose, scope and goals of the meeting are defined. An agenda is created upfront to allow the meeting participants to prepare for it. In addition, the attendance list is determined, the meeting is scheduled, a room is booked, and an invitation is send to the participants.

Running the meeting is the second phase. The meeting is started by an opening phase and finished by a closing part. During the meeting several presentations, discussions, brainstorming or voting units are held.

The *follow-up phase* of a meeting involves the process of writing meeting minutes to protocol made decisions, new action items, or discussed problems. To influence the long-range effectiveness of the meeting the phase also includes a review of the meeting notes, paying special attention to the action items assigned to the participant. The meeting protocol should be sent to all participants and stakeholders, who did not participate. It has to be followed up whether the assigned tasks are being carried out, and detected issues and concerns have to be kept in focus. This meeting workflow is illustrated in Figure 3-7.

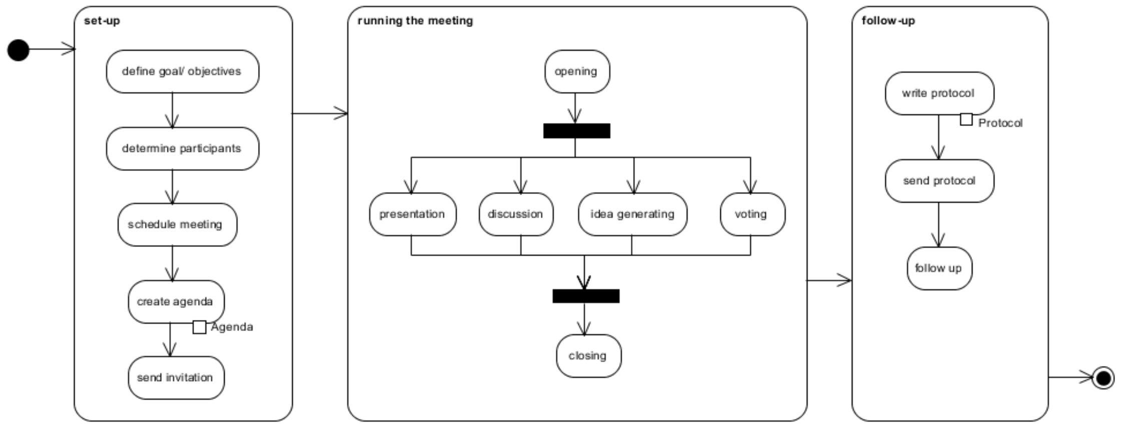


Figure 3-7: Typical meeting workflow

Meetings can be categorized according to their frequency, the formalism, their dimensions, and their types.

The meeting organizer has to determine the repetition and *frequency* of occurrence of the meeting. Options generally include the following: A *one-time meeting* is the most common meeting type and covers *individual* meeting events that are self-contained. The *occasional meeting* is composed of people whose normal work does not bring them into contact and whose work has little or no relationship to that of the others. A *recurring meeting* is a meeting that recurs periodically, such as a staff meeting from 9:00 am to 9:30 am every Monday, or a Daily Scrum Meeting. The meeting organizer wants the participants to be at the meeting on a constant and regular basis. A recurring meeting can be ongoing, such as a weekly team meeting, or have an end date, such as a five week training meeting, held every Friday afternoon. A recurring meeting can be a daily, weekly or monthly meeting. A *series meeting* is like a recurring meeting, but the details differ from meeting to meeting. One example of a series meeting is a monthly 'lunch and learn' event at a company. The place is the same, but the agenda and topics vary.

Meetings can be distinguished according their formalism in *informal meetings* and *formal meetings*. Informal meetings are generally not planned in advance and the participants are not notified through means such as memos or emails. If a team member needs an advice or has a problem, he calls a colleague, expecting that he can help him, or two team members meet on the way to the canteen, talking about the latest news. So the content is mostly limited to one topic. Informal meetings can also take place in neutral surroundings, for instance in a restaurant rather than a formal boardroom. In contrast, formal meetings are preplanned meetings. A predetermined set of topics are discussed along with a set of objectives that one wishes to achieve at the end of the meeting. The members of the meeting are often given a considerable period of notice before the meeting, preferably through formal means. As the title suggest, the atmosphere in such meetings is generally formal. Table 3-1 opposes the criteria of formal to informal meetings.

Criteria	Formal meeting		Informal meeting	
Attributes	- planned		- unplanned	
	- meeting room		- everywhere	
	- fixed date		- any time feasible	
	- Ø 1-2 hours		- short (~10 minutes)	
	- written agenda		- purpose	
	- meeting minutes		- no documentation	
Aim	- jour fix	well pre-pared	- problem solving	quick
	- team building		- new information	
	- brainstorming		- advice needed	
	- problem solving		- information sharing	
	- information sharing		- status update	
Process	- create agenda		- "Do you have time now?" or	
	- find date		- "Good that I see you…"	
	- book room		- discuss problem/ share information	
	- send invitation		- if necessary, each participant takes individual notes	
	- acceptance/ cancelation			
	- conduct meeting			
	- write meeting minutes			
	- send protocol			
Topics	- several topics		- one topic	
	- reprocessing of old topics			
Participants	- e.g. whole team (5-20, even more)		- few (2-4)	

Table 3-1: Formal vs. informal meetings

Agile projects are strongly affected by the communication component. This results in information exchanges that often take place in the office kitchen rather than in formal meeting rooms. Decisions can be made in an informal meeting between the involved persons without waiting for the next official meeting, or even calling an extra formal meeting. The project remains flexible and can quickly react on change. The amount of informal meetings is far greater than the one of formal meetings. It is comparable to an iceberg: formal meetings are only the tip of all meetings. Only a few meetings are formal: the Planning Meeting at the beginning of an iteration, daily 15-minute status meetings, and a review and retrospective meeting at the end of an iteration. However, under the surface, there are a lot more 'meetings' every day, for instance during pair programming where two developers discuss product features and possibly assign new tasks, or some team members meet at the hallway exchanging information and making decisions relevant for the project progress. Especially those unplanned meetings are an important part of the agile communication and a source for knowledge creation and transfer.

In addition to the temporal and formal differentiation, meetings can be labeled according to the importance (from important to unimportant), priority (high to low), or urgency (urgent to trivial). Moreover, a separation according the scope (short to comprehensive) or scheduling (planned or unplanned) can be done.

There are several *meeting types*, corresponding to their participants, the meeting style, aim and purposes, and the applied project management method. Following meeting types can be differentiated (Doyle & Straus, 1976): A classification according the participating people comprises *team meetings*, where colleagues are working on various aspects of a team project and *staff meetings*, typically between a manager and those that report to the manager. In addition, there are *management meetings* among managers, *board meetings*, where the board of directors of an organization or project comes together, and *one-on-one meetings*, a meeting between two individuals. Furthermore, a meeting can be classified as *expert meeting*, a *conference*, or *panel discussion*.

Meetings have different styles, depending on the way how they are conducted. There are *breakfast meetings* or *lunch meetings*, *off-site meetings*, *stand-up meetings*, or *lab meetings*. Moreover, a meeting can take place in a classical *meeting room*, via *telephone*, or *video conference*.

Each meeting has a predefined *purpose*. There are *problem solving meetings*, where a problem/ issue is identified and defined, solutions are searched and evaluated, and if necessary an action plan (including the assignment of tasks) is determined (Brügge & Dutoit, 2004). Moreover, there are *idea generation meetings*, e.g. a brainstorming session in combination with a problem solving meeting to develop creative ideas for solutions. During a *decision making meeting* it is the aim that the participants come to a decision, while the *planning meeting* is targeted to identify the next steps and an action plan, including delegations and task assignments. In addition, there are *feed forward* and *feedback meetings*. Feed forward meetings comprise status reporting and information presentations, as well as a Jour Fix, where a fixed group meets at a fixed date, also without a concrete occasion (to increase the "We-Feeling"). Feedback meetings are reactive and evaluating meetings, including client or project reviews, peer reviews and postmortems/ retrospectives. Furthermore there are combinations of these meetings, like a kick-off meeting – an informative meeting with planning components to define the next steps afterwards. If the meeting is called together spontaneously for a special purpose, it is an *ad-hoc meeting*.

Specific project management methods use special meeting types, for instance Scrum distinguishes between Sprint Planning Meeting, Daily Scrum Meeting, Sprint Review Meeting, and Sprint Retrospective Meeting.

3.2.1 Meeting taxonomy

Each meeting, depending on the purpose of the meeting, consists of several recurrent meeting elements. For instance, a problem solving meeting consists of the following elements: issue (problem) definition, a discussion about alternative positions (possible answers) and arguments (that support or object to a given position or argument), and a decision (the final solution) (Kunz & Rittel, 1970). During a feedback retrospective meeting, there is a discussion and brainstorming part to gather best-practices and improvement potential, followed by a decision phase, where an action plan is approved and where necessary action items are assigned.

Based on the meeting types and their elements we defined a meeting taxonomy. This classification of (recurrent) meeting components forms a logical structure of communication, which shows meeting type specific elements, but also allows the components to be almost arbitrarily combined to new meeting types. Table 3-2 gives an overview of the meeting elements for the meeting types discussed above.

Meeting type	Meeting elements
Problem solving	- Problem definition - Discussion (positions & arguments) - Decision - Definition of actions
Idea generation	- Topic definition - Discussion (idea gathering)
Decision making	- Topic definition - Definition alternatives - Decision
Planning	- Gathering of new requirements - Work package splitting - Task assignment - Re/prioritization - Update plan - Re/estimation of work packages/ tasks
Feed forward: status report	- Status information (task, project, …) - Task delegation

	- Re/estimation of tasks - Controlling (project progress, budget)
Feed forward: information presentation	- Information announcement
Feedback: review	- Customer approval - Change requests, new requirements - Controlling
Feedback: retrospective	- Discussion: gathering process feedback - Decision: best-practices - Brainstorming: improvements - Definition of actions (incl. task assignment)
Meeting type independent	- Greeting - Closing - Scheduling next meeting

Table 3-2: Overview meeting types and corresponding meeting elements

Formally, the table can be interpreted as

$$meeting\ element \subsetneq meeting\ type.$$

That is, for example $problem\ definiton \subsetneq problem\ solving\ meeting$.

The meeting taxonomy is extensible by project method specific meeting types. Figure 3-8 shows the extended taxonomy for the agile project management method Scrum.

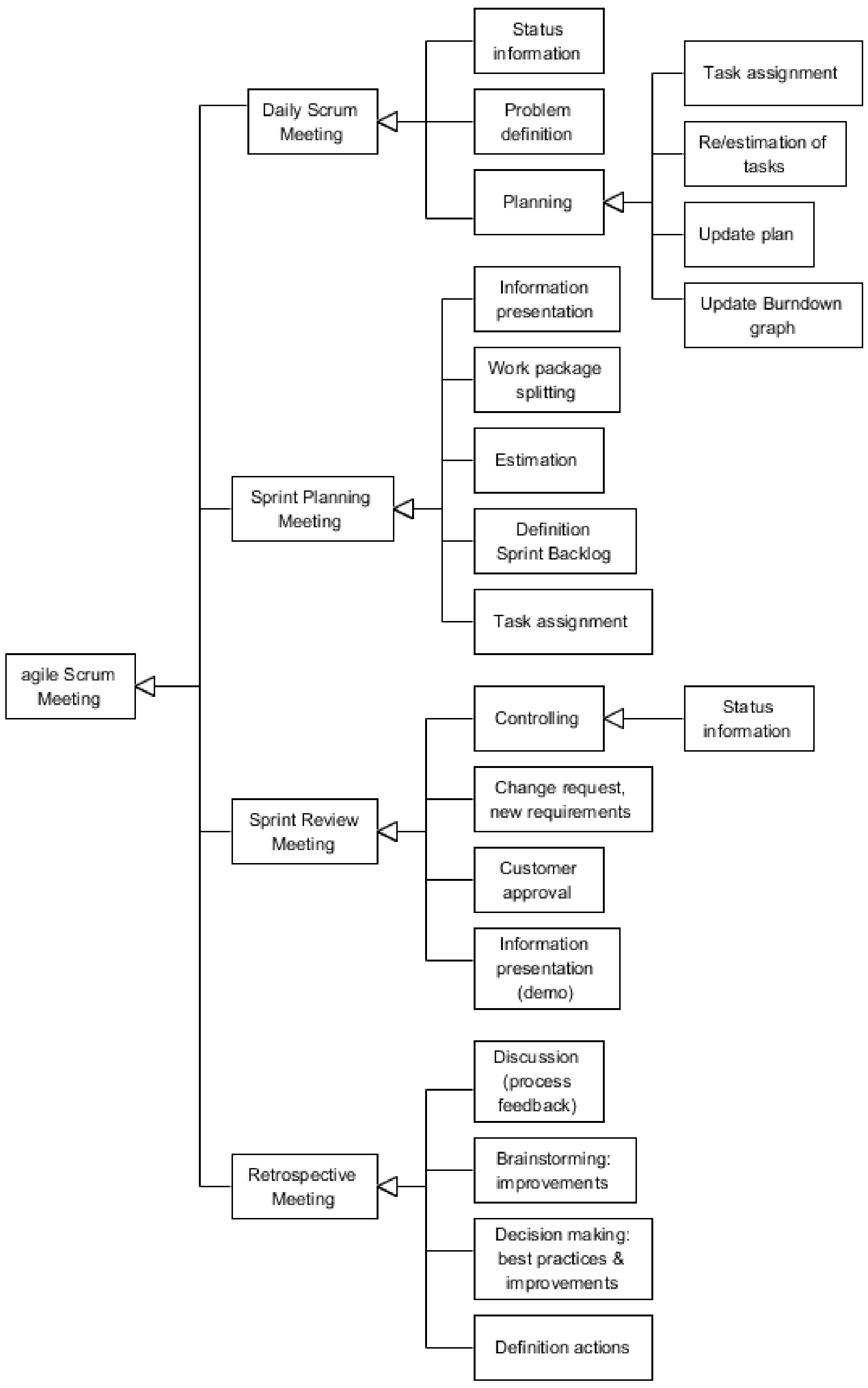


Figure 3-8: Taxonomy for agile Scrum Meetings

3.2.2 Meeting best-practices for formal meetings

Independent of the meeting type, meetings are an important vehicle for personal contact in organizations. They are so common and pervasive in organizations that they are often taken for granted; however, unless properly planned and executed, meetings can be a waste of time and resources (Kayser, 1995). Meetings have often a bad reputation, mainly because people end up sitting through so many meetings that seem pointless.

For this reason, a lot of best practices, advices and optimal meeting procedure have been released, regarding the meeting process, logistics, and roles (Williams, 2008), (Doyle & Straus, 1976). The meeting best practices can be allocated to the three phases: meeting set- up, the meeting itself, and a meeting follow-up, presented in chapter 3.2.

During setting-up a meeting, the needs and goals of the meeting are defined. An *agenda* is created upfront and is sent with the meeting invitation to allow the meeting participants to prepare for it, so that by the time the meeting takes place, they can already be on board as to what the intention of the meeting is. Moreover, the team members need to trust that if a discussion arises in which they should be involved, discussions are not made without them. The meeting agenda includes a list of participants, the purpose and goals for the meeting, a list of discussion topics, and any background material or a list of documents that should be reviewed prior to the meeting. That is, in advance the meeting's objective has to be cleared. The purpose of the meeting, the end products, i.e. the outcomes of the meeting, and standards, the meeting will adhere to, have to be defined.

Additionally, the meeting is labeled as to type and function, whether it is a decision meeting, a brainstorming meeting, a retrospective meeting, or a regular team meeting. Whatever the type of meeting, it should be clearly defined in advance, and a decision made who exactly needs to be there. As it is easy to simply invite everyone to every meeting, the list of 'must attend' *invitees* should be kept as small as possible and only people should be added to the 'optional' list if they truly might be interested in the meeting. Thus the meeting will be more focused and more effective.

When setting-up a meeting, *logistics* have to be considered as well. Depending on the number of participants and the goal of the meeting an appropriate room and material has to be provided. If necessary, internet access, a conference phone, or a white board has to be made available.

All conference *rooms* should be reserved to allow a smooth start without any bad surprises. If the meeting will require the use of laptops, projectors or other technical equipment, additional 15 minutes before and after the designated meeting time should be allowed for to set up and break down any equipment.

In the second phase, a meeting *timekeeper* and *facilitator* have to be assigned. The Timekeeper is responsible for monitoring the clock and notifying the host if the meeting is falling behind schedule, based on the agenda provided. A meeting facilitator has to be assigned to separate the powers, which means that the power of decision making (the boss) has to be separated from the power of the process (the facilitator) (Edelman & Crain, 1994). Neutral and non-evaluating, the meeting facilitator is responsible for making sure the participants are using the most effective methods for accomplishing their task in the shortest time. So the manager, as decision maker, can fully participate in the meeting.

Keeping focused and the meeting as *short* as possible is one of the important best-practices for meetings so the meetings don't become a waste of time for the participants. There are very few real situations where a meeting should take longer than one hour, or even three or four. Usually, long meetings like these are combinations of smaller meetings, which have to be divided.

A *parking lot,* i.e. a flip-chart or whiteboard to write down important parts of the communication, like ideas or decisions, or postponed topics will help keeping focused. Whenever anyone starts getting into a very detailed discussion that would be better dealt with outside the meeting, it can be 'parked' by noting it on the parking lot page. However, it has to be kept in mind that these issues are dealt with, otherwise it will just be seen as trying to silence discussion. A *meeting recorder* will be responsible to take these notes and support the short-term memory of the meeting group (Doyle & Straus, 1976).

The meeting follow-up phase involves the process of writing meeting minutes to protocol the conversation. Summarizing the important parts of the meeting (e.g. decisions or action items) in a *protocol* will help the meeting participants to continue a successful meeting with a successful post-processing and the absent team members to keep informed, what happened in the meeting. Formal protocols have to be approved by the meeting participants or one representative, e.g. a manager, before they are published. The meeting protocol is sent to all stakeholders. Assigned tasks and detected issues and concerns have to be kept in focus after the meeting. In addition, the parking lot items should be reviewed and necessary follow-up meetings be scheduled to address outstanding issues.

3.2.3 Best practices for informal meeting

Based on the established best-practices for formal meetings, the following best-practices for informal meetings are derived.

Informal meetings are unplanned ad-hoc meeting with no preplanning phase, i.e. no set-up phase. The agenda is based on a question, an urgent issue, or a piece of information that one team member wants to discuss with the other meeting participants. Ad-hoc meetings allow no detailed preparation for the meeting. However, also in informal meetings a whiteboard helps the team members to visualize and explain ideas, and record discussed information and decisions.

Informal meetings have to be focused and as short as possible, i.e. only one or a few topics should be addressed in order to reduce the necessary amount of meeting time to a minimum. So the meeting participants can return as fast as possible to their planned work. If there are more questions or topics to be discussed, a formal meeting has to be planned.

After the meeting – also in informal meetings – meeting minutes have to be written, summarizing the important parts of the meeting, for instance decisions or assigned action items. Therefore notes have to be taken during the meeting by one participant.

During the follow-up phase of the meeting, a protocol has to be written based on the taken notes. The protocol is sent to the meeting participants to summarize assigned tasks and detected issue that have to be solved. Moreover, all stakeholders have to be informed, especially regarding decisions, problems and next steps.

3.2.4 Meeting Minutes

Meeting minutes (also know as protocols) are documented records of a meeting, containing externalized information.

Protocols summarize the important parts of the meeting (for instance decisions and action items), hence acting as a steering and controlling tool for the meeting facilitator during the meeting. After the meeting, a protocol is aid to memory for the meeting participants and information source for outsiders or absent team members. Meeting minutes are procedure documentation for realization activities, enabling a successful post-processing, as well as evidence for possible later conception discrepancies (Laufer, 2009).

Each protocol comprises results and/ or actions, as well as the related dates and responsible persons. Meeting minutes can be differentiated into three types: verbatim protocol, narrative and resolution/ decision protocol.

The *verbatim protocol* is a word-for-word recording of everything said in the meeting. Verbatim minutes are appropriate complex, both for the minutes taker and the reader. Thus, they play a minor role in operative business and are written only in two situations: the proceedings of legislative bodies (Congress, Houses of Parliaments, etc.) and protocols at the court are recorded verbatim.

Minutes of Narration are not verbatim, but still the whole meeting progress is described. They include some of the discussions that took place and important details and give a fuller picture of the debate, including an account of discussions leading up to the decision and the views expressed by various members of the meeting. Hence, they are the archived record of discussions that exist after the meeting has finished. It is important in many cases to have well-written narrative minutes because in several year's time people may not just want to know what decisions were made at the meeting but how professional they were made, i.e. whether all the points were discussed, whether proper experts were brought in, and so on.

In operational meetings, normally it is sufficient to apply Occam's razor and record the final results or made decisions. *Resolution/ decision minutes* are limited to the recording of the actual words of all resolutions that were passed, and not the discussions leading up to those decisions, which should have been set out in the narrative minutes.

Furthermore, there is a special form of minutes, the *memory minutes*, written after the meeting from memory.

Protocols consist of three parts. The header comprises the basic information about the conversation. Normally, these information are place, date, and time, as well as the participants (number, un/excused) and the meeting facilitator. Moreover, motivation and agenda items are in the first part of a protocol.

The minutes in the narrower sense are depending on the protocol type. Its structure arises from the agenda and reproduces requests, issues, proposals, and criticisms, plus decisions, action items, and brief rationales.

In the final part – mandatory, if the protocol shall have a legal relevance – are the signatures of the minutes keeper and the meeting facilitator or another responsible person.

On a protocol, in terms of a provable recording, high demands are made. The requirements include completeness and correctness with regards to the content, i.e. all discussed information has to be recorded, accurately and no errors have to be in the protocol. The relevance of the recorded results and activities has to be ensured, as well as authenticity of the authorship, and the validity of the protocol. Only if a protocol guarantees these requirements it can give reliable information. Moreover, the demands have to be satisfied and facilitated by the right time of generation (present vs. protocol from memory), the method for minutes taking (minutes taker or technical equipment), during writing (objectivity) and storage of the data (archiving, stable medium with access control).

Meeting minutes are considered as a fundamental *source of information* for building knowledge bases and repositories (Corrall, 1999), but they are not always available, especially for informal meetings. While tacit knowledge is communicated in meetings, in order to be useful to anyone beyond the person who owns it, the knowledge additionally has to be converted into explicit knowledge. That externalization of knowledge, which is then reusable for knowledge management processes, is done by writing a protocol that satisfies the requirements mentioned above.

The primarily target group of meeting minutes is the project team. Team members and project stakeholders, e.g. the customer, need to be informed. The protocol is a knowledge source and a rational document, i.e. the meeting minutes help the team members to reconstruct decisions in further phases of the project. Moreover, another target group are other projects within the company. Managers and team members may be interested in problems and risks, to consider the issues in their project plan, or profit from established best-practices.

Writing meeting minutes is part of the best-practices. Nevertheless, there are often no protocols, as despite the high benefit, they are considered as dispensable, or there is nobody willing to assume the task of the minutes taker. Sometimes, protocols are written after the meeting from memory. Minutes written in this way are often incomplete and incorrect and possess misconceptions. Subsequent disagreements and rework activities are not unusually.

Writing high quality meeting minutes is a critical success factor for software development projects and the company advancement. To support the principles of an agile proceeding, in addition the effort for writing and publishing a protocol has to be minimized, so the developers are not kept from software development (the aim) by 'unnecessary' work. A new approach is required that reduces the documentation effort with simultaneously increasing the protocol quality.

We propose a protocol automation approach for knowledge externalization. This approach is based on audio-/video-recording the meetings, to record and store the meeting conversation and the discussed information. As only recording the meetings is not sufficient, because the essential information would be lost in the amount of audio- or video material, the important information (decisions, problems, etc.) have to be extracted and summarized in meeting minutes. Therefore a semantic structuring of the conversional recording is required. This will be based on the above proposed meeting taxonomy and a procedure, where each item of the agenda or made result should be adopted directly after it is discussed and be formulate for the protocol, if all agree (Laufer, 2009). So the meeting efficiency is increased, because finished agenda items are closed, the next (open) steps become clearer, and later misconceptions are avoided. In fact, the protocol can become a useful steering tool.

4 Protocol generation technologies

Technologies for automated summaries and protocol generation

The problem of generating an automated concise and meaningful protocol has been addressed from different perspectives, in varying domains and using various paradigms. This chapter intends to investigate approaches for automated text summarization (section 4.1), speech recognition (section 4.2) and summarization of spoken language (section 4.3), as fundamental technologies used in protocol generation. Applications, also focusing on protocol generation, are introduced in section 4.4.

4.1 Automatic text summarization

Automatic summarization of texts is mainly investigated in the field of natural language processing. The predominant approach of natural language processing is based on a statistical procedure, where the sum of significance of all words in one sentence is calculated, the sentences with the highest sums are extracted and thus the text is summarized.

Radev, et al. (2002) define a summary as "a text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually significantly less than that". This definition captures three important aspects that characterize research on automatic summarization:

- Summaries may be produced from a single document or multiple documents,
- Summaries should preserve important information,
- Summaries should be short.

Following, summarization terminology is introduced, based on the definitions of Das and Martins (2007) and Radev et al. (2002): *extraction* is the procedure of identifying important sections of the text and producing them verbatim; *abstraction* aims to produce important material in a new way; *fusion* combines extracted parts coherently; and *compression* aims to throw out unimportant sections of the text. Earliest instances of research on summarizing scientific documents or newswire data used features like *word* and *phrase frequency* (Luhn, 1958), *position* in the text (Baxendale, 1958) and *key phrases* (Edmundson, 1969).

Usually, the flow of information in a given document is not uniform. Thus, the major challenge in summarization lies in distinguishing the more informative parts of a document from the less ones. In this section, we describe eminent extractive techniques in the research field on summarization, as well as approaches involving machine learning techniques. Finally, the field of short summaries is briefly described.

Luhn (1958) was the first, who proposed that the *frequency* of a particular word in an article provides a useful measure of its significance. He put forward several key ideas that have assumed importance in later work on summarization. As a first step, words were stemmed to their root forms, and stop words (e. g. "a", "be", "the", "so") were deleted. Then a list of content words sorted by decreasing frequency was compiled, the index providing a significance measure of the word. On a sentence level, a *significance factor* was derived that reflects the number of occurrences of significant words within a sentence, and the linear distance between them due to the intervention of non-significant words. All sentences are ranked in order of their significance factor, and the top ranking sentences are finally selected to form the auto-abstract.

Baxendale (1958) provided early insight on a particular feature helpful in finding salient parts of documents: the *sentence position*. Towards this goal, 200 paragraphs were examined to find that in 85% of the paragraphs the topic sentence came as the first one and in 7% of the time it was the last sentence. Thus, a naive but fairly accurate way to select a topic sentence is to choose one of these two.

Edmundson (1969) described a system that produces document extracts. He developed a typical structure for an extractive summarization experiment. The two features of word frequency and positional importance were incorporated from Luhn and Baxendale. Two other features were used: the presence of *cue words* (presence of words like *significant*, or *hardly*), and the *skeleton* of the document (whether the sentence is a title or heading). Weights were attached to each of these features manually to score each sentence.

With the advance of machine learning techniques in the field of natural language processing, statistical techniques where published to produce document extracts. Kupiec et al. (1995) describe a method derived from Edmundson (1969) that is able to learn from data. The classification function categorizes each sentence as worthy of extraction or not adequate, using a *naïve-Bayes classifier*. The features, which are compliant to the features proposed by Edmundson, additionally included the *sentence length* and the *presence of uppercase words*. Each sentence is given a score, and only the n top sentences are extracted.

Aone et al. (1999) also incorporate a naïve-Bayes classifier, but with richer features. Their system DimSum makes use of features like term frequency (*tf*) and inverse document frequency (*idf*) to derive *signature words*, those words that indicate key concepts in a document. The idf is computed from a large corpus (a database of speech audio files and text transcriptions that represents the natural language) of the same domain as the concerned documents. A shallow discourse analysis is employed to maintain cohesion, e.g. reference to same entities in the text. The references are resolved at a very shallow level by linking name aliases within a document like

"U.S." to "United States", or "IBM" for "International Business Machines". Synonyms and morphological variants are also merged while considering lexical terms, the former being identified by using Wordnet (Miller, 1995).

The importance of a single feature, the *sentence position*, was studied by Lin and Hovy (1997) based on the ideas of Baxendale (1958). Weighing a sentence by its position in text, which is termed as the "position method", arises from the idea that texts generally follow a predictable discourse structure, and that the sentences of greater topic centrality tend to occur in certain specifiable locations (e.g. title or abstracts). However, the discourse structure significantly varies over domains, so Lin and Hovy investigated techniques of tailoring the position method towards optimality over a genre and how it can be evaluated for effectiveness. A newswire corpus was used, which consists of text about computer and related hardware, accompanied by a set of key topic words and a small abstract of six sentences. For each document in the corpus, the authors measured the yield of each sentence position against the topic keywords. They then ranked the sentence positions by their average yield to produce the *Optimal Position Policy* (OPP) for topic positions for the genre.

Lin (1999) breaks away from the assumption that features are independent of each other and tries to model the problem of sentence extraction using *decision trees*, instead of a naïve-Bayes classifier. He examines a lot of features and their effect on sentence extraction. The data used in this work is a publicly available collection of texts, classified into various topics. The dataset contains essential text fragments (phrases, clauses, and sentences) that are each evaluated by a human judge. The SUMMARIST system, developed at the University of Southern California, is foundation for the experiments of Lin. The system extracts sentences from the documents and matches them against human extracts.

Some novel features are the *query signature* (normalized score given to sentences depending on number of query words that they contain), *IR signature* (the *m* most salient words in the corpus, similar to the signature words of Aone et al. (1999)), *numerical data* (Boolean value 1 given to sentences that contains a number in them), *proper name* (Boolean value 1 given to sentences that contains a proper name in them), *pronoun or adjective* (Boolean value 1 given to sentences that contains a pronoun or adjective in them), *weekday or month* (similar as previous feature) and *quotation* (similar as previous feature). Some features like the *query signature* are question-oriented because of the setting of the evaluation, unlike a generalized summarization framework. Various baselines are experimented with, for instance using only the positional feature, or using a simple combination of all features by adding their values. When evaluated by matching machine extracted and human extracted sentences, the decision tree classifier is clearly the winner for the whole dataset.

In contrast with these feature-based and non-sequential approaches, Conroy and O'leary (2001) model the problem of extracting a sentence from a document using a *hidden Markov model* (HMM). The basic motivation for using a sequential model is to account for *local dependencies* between sentences. Only three features were used: *position* of the sentence in the document (built into the state structure of the HMM), *number of terms* in the sentence, and *likeliness* of the sentence terms given the document terms.

Svore et al. (2007) proposed an algorithm based on neural nets and the use of third party datasets to tackle the problem of extractive summarization. A model is trained from the labels and the features for each sentence of an article that could infer the proper ranking of sentences in a test document. The ranking is accomplished using RankNet (Burges, et al., 2005), a pair-based neural network algorithm designed to rank a set of inputs that uses the gradient descent method for training. For the training set, ROUGE-1 (Lin, 2004) is used to score the similarity of a human written highlight and a sentence in the document. These similarity scores are used as soft labels during training, contrasting with other approaches where sentences are "hard-labeled", as selected or not selected.

Some of the used features based on position or *n*-grams frequencies have been observed in previous work. The novelty of the new frameworks lay in the use of features that derived information from query logs from search engines, e.g. Bing (Microsoft, 2010), Google (Google, 2010) and Wikipedia (Wikimedia Foundation, 2001) entries. If a document sentence contains keywords used in the news search engine, or entities found in Wikipedia articles, then there is a greater chance of having that sentence in the highlight.

Barzilay and Elhadad (1997) used an amount of linguistic analysis for performing the task of summarization. L*exical chains,* sequences of related words in a text, are applied spanning short (adjacent words or sentences) or long distances (entire text). The method progressed with the following steps: segmentation of the text, identification of lexical chains, and using strong lexical chains to identify the sentences worthy of extraction. *Cohesion* in text means sticking together different parts of the text. Lexical cohesion is a notable example where semantically related words are used.

The phenomenon of cohesion occurs not only at the word level, but at word sequences too, resulting in lexical chains, used as a source representation for summarization. Semantically related words and word sequences are identified in the document, and several chains are extracted, that form a representation of the document. To find out lexical chains, Wordnet (Miller, 1995) is used, applying three generic steps: a set of candidate words is selected, for each candidate word, an appropriate chain relying on a relatedness criterion among members of the chains is found, and then the word is inserted in the chain. The relatedness is measured in terms of Wordnet distance. The chains are scored by their length and homogeneity. Then heuristics are applied to select the significant sentences. Strong lexical chains are used to create the summaries.

Ono et al. (1994) put forward a computational model of discourse for Japanese expository writings, where they elaborate a practical procedure for extracting the discourse *rhetorical structure*, a binary tree representing relations between chunks of sentences. This structure is extracted using a series of natural language processing steps: sentence analysis, rhetorical relation extraction, segmentation, candidate generation and preference judgment. Evaluation is based on the relative importance of rhetorical relations. In a following step, the nodes of the rhetorical structure tree are pruned to reduce the sentence, keeping its important parts. Same is done for paragraphs to finally produce the summary.

A unique approach towards summarization, that, unlike most other previous work, does not assume that the sentences in a document form a flat sequence, is described by Marcu (1998a). Discourse based heuristics are used with the traditional features that have been used in the summarization literature. Rhetorical Structure Theory (RST) holds between two non-overlapping pieces of text spans: the *nucleus* and the *satellite*. The distinction between nuclei and satellites comes from the empirical observation that the nucleus expresses what is more essential to the writer's purpose than the satellite; and that the nucleus of a rhetorical relation is comprehensible independent of the satellite, but not vice versa. Marcu (1998b) describes a rhetorical parser producing a discourse tree. Once such a discourse structure is created, a partial ordering of important units can be developed from the tree. Each equivalence class in the partial ordering is derived from the new sentences at a particular level of the discourse tree.

If it is specified that the summary should contain the top $k\%$ of the text, the first $k\%$ of the units in the partial ordering can be selected to produce the summary. Marcu (1998b) states that thus a summarization system based just on this method is possible, where the discourse based heuristics are merged with traditional heuristics. The metrics are *clustering based metrics* (each node in the discourse tree is assigned a cluster score; for leaves the score is 0, for the internal nodes it is given by the similarity of the immediate children; discourse tree A is chosen to be better than B if its clustering score is higher), *marker based metrics* (a discourse structure A is chosen to be better than a discourse structure B if A uses more rhetorical relations than B), *rhetorical clustering based techniques* (measure the similarity between salient units of two text spans), *shape based metrics* (prefer a discourse tree A over B if A is more skewed towards the right than B), *title based metrics*, *position based metrics*, *connectedness based metrics* (cosine similarity of an unit to all other text units, a discourse structure A is chosen to be better than B if its connectedness measure is more than B). A weighted linear combination of all these scores gives the score of a discourse structure. To find the best combination of heuristics, the weights that maximizes the F-score on the training dataset are computed, which is constituted by newswire articles. To do this, a GSAT-like algorithm (Selman, Levesque, & Mitchelle, 1992) is used that performs a greedy search in a seven dimensional space of the metrics.

Rather than aiming to build full summarization systems, *short summaries* are aimed to extract only captions, which approach the meeting protocols, where not the discussions should be summarized, but the conclusions in one sentence.

Witbrock & Mittal (1999) claim that extractive summarization is not very powerful in that the extracts are not concise enough when very short summaries are required. Thus they created a system that generates headline style summaries. The system learns statistical models of the relationship between source text units and headline units. It attempts to model both the order and the likelihood of the appearance of tokens in the target documents. Both models, one for content selection and the other for surface realization, are used to co-constrain each other during the search in the summary generation task. For content selection, the model learns a translation model between a document and its summary (Brown et al., 1993). This model in the simplest case can be thought as a mapping between a word in the document and the likelihood of some word appearing in the summary. To simplify the model, the probability of a word appearing in a summary is assumed as independent of its structure. This mapping boils down to the fact that the probability of a particular summary candidate is the product of the probabilities of the summary content and that content being expressed using a particular structure. The surface realization model uses was a bigram model. Viterbi beam search is used to efficiently find a near-optimal summary. The Markov assumption is violated by using backtracking at every state to strongly discourage paths that repeat terms, since bigrams that start repeating often seem to pathologically overwhelm the search otherwise.

A statistical approach to *sentence compression* is introduced by Knight and Marcu (2000). The idea is that understanding the task of compressing a sentence may be a fruitful first step to later tackle the problems of single and multi-document summarization.

Sentence compression is defined as follows: given a sequence of words $W = w_1\, w_2 \ldots w_n$ that constitute a sentence, find a subsequence $w_{i1}\, w_{i2} \ldots w_{ik}$, with $1 \leq i_1 < i_2 < \ldots i_k \leq n$, that is a compressed version of W. Two different approaches are considered: one that is inspired by the *noisy-channel model* and another one based on *decision trees*. Knight's and Marcu's model has the advantage of decoupling the goals of producing a short text that looks grammatical and of preserving important information (which is done through the channel model). In (Knight & Marcu, 2000), the source and channel models are simple models inspired by *probabilistic context-free grammars* (PCFGs).

Daumé III and Marcu (2002) extend this approach to *document* compression by using *rhetorical structure theory* as described by Marcu (1998a), where the entire document is represented as a tree, hence allowing not only to compress relevant sentences, but also to drop irrelevant ones. In this framework, kernel methods are employed to decide for each node in the tree whether or not it should be kept (Daumé III & Marcu, 2004).

The attention of research in automatic test summarization has drifted from summarizing scientific articles to news articles, electronic mail messages, advertisements, and blogs. Both abstractive and extractive approaches have been attempted, depending on the application at hand. Usually, abstractive summarization requires heavy machinery for language generation and is difficult to replicate or extend to broader domains. In contrast, simple extractions of sentences have produced satisfactory results.

4.2 Speech recognition

Automatic speech recognition converts acoustic signals, captured by a microphone or a telephone, to word phrases. The recognized words can be the final results, as for applications such as commands & control, data entry, and document preparation. They can also serve as the input to further linguistic processing in order to achieve speech understanding.

Speech recognition applications belong to one of three categories: dictation or document creation systems, navigation or transactional systems (e.g. automated voice response systems), and multimedia indexing systems. In dictation systems, the words spoken by a user are transcribed verbatim into text to create documents such as personal letters or business correspondence. In navigation systems, the words spoken by a user are used to follow links on the web or to navigate around an application. In transactional systems, the words spoken by a user conduct a transaction such as stock purchase or banking transactions. In multimedia indexing systems, speech is used to transcribe the audio (possibly extracted from a video) into text, and subsequently, information retrieval techniques are applied to create an index with time offsets into the audio (video). Advances in technology are making significant progress towards the goal of allowing any individual to speak naturally to a computer on any topic and the computer accurately understands what was said. However, we are not there yet. Even state-of-the-art continuous speech recognition systems require the user to speak clearly, enunciate each syllable properly, and have one's thoughts in order before starting.

The first speech recognizer was developed by Davies et al. (1952) and consisted of a device for the recognition of single spoken digits. Fry and Denes (Fry, 1959) (Denes, 1959) built a phoneme recognizer to recognize four vowels and nine consonants. Pattern-recognition ideas were introduced in speech recognition by Velichko and Zagoruyko (1970), Sakoe and Chiba (1978), and Itakura (1975). Tappert et al. (1971) and Jelinek (1975), (1985) investigated in large vocabulary speech recognition at IBM. At AT&T Bell Labs, researchers experimented with speech-recognition systems that are truly speaker independent (Rabiner, Levinson, Rosenberg, & Wilpon, 1979), as well as recognizing a fluently spoken string of words (e.g. digits) based on matching a concatenated pattern of individual words (Myers & Rabiner, 1981), (Sakoe, 1979), (Lee & Rabiner, 1989).

Ferguson (1980) shifted the technology from template-based approaches to statistical modeling methods, using the hidden Markov model approach to problems in speech recognition. The idea of applying neural networks was investigated by Lippmann (1987), and Weibel et al. (Weibel, Hanazawa, Hinton, Shikano, & Lang, 1989).

A major impetus was given to large vocabulary, continuous-speech-recognition systems by the Defense Advanced Research Projects Agency (DARPA) community, which sponsored a large research program aimed at achieving high word accuracy for a 1000-word, continuous-speech-recognition database management task. Major research contributions resulted from efforts at CMU (notably the well-known SPHINX system by Lee et al. (1990)), BBN with the BYBLOS system presented by Chow, et al. (1987), SRI (Weintraub & et al., 1989), MIT (Zue et al., 1989), and research from Lee et al. (1990) at AT&T Bell Labs. In addition, the Hearsay-II system, developed by Erman et al. (1980) was developed during the DARPA-sponsored speech-understanding research program. This system represents a specific solution to the speech-understanding problem and a general framework for coordinating independent processes to achieve cooperative problem-solving behavior to generate and evaluate speech hypotheses. The speech understanding problem is structured as a space, in which the problem solver searches for a solution. The space is the set of (partial and complete) *interpretations* of the input acoustic signal, i.e. the (partial and complete) mappings from the signal to the possible message. The goal of the problem solving system is to find a complete interpretation (i.e. a message and mapping). The key functions of generating, combining, and evaluating hypothetical interpretations are performed by diverse and independent programs called *knowledge sources* (KSs). KSs perform a variety of functions, including extracting acoustic parameters, classifying acoustic segments into phonetic classes, recognizing words, parsing phrases, and generating and evaluating predictions for undetected words or syllables. Because each KS is an independent condition-action module, KSs communicate through a global database called the *blackboard*. The blackboard records the *hypotheses* generated by KSs. Any KS can generate a hypothesis (record it on the blackboard) or modify an existing one. In this framework the blackboard serves in two roles: it represents intermediate states of problem-solving activity, and it communicates messages (hypotheses) from one KS that activate other KSs. The recognition processing finally halts in one of two ways. First, there may be no more partial hypotheses left to consider for prediction and extension; Second, if the system expends its total allowed computing resources (time or space). The highest rated hypothesis/ hypotheses are the result of the recognition process.

While the DARPA program shifted its emphasis to natural language front ends and to retrieval of air travel information, speech-recognition technology was increasingly used within telephone networks to automate as well as enhance operator services.

Pawar et al. (2005) distinguish three approaches to automatic speech recognition by machine: the acoustic-phonetic approach, the pattern recognition approach, and the artificial intelligence approach.

The acoustic-phonetic approach is based on the theory of acoustic phonetics that postulates that there exist finite, distinctive phonetic units in spoken language and that the phonetic units are broadly characterized by a set of properties that are manifest in the speech signal, or its spectrum, over time. Even though the acoustic properties of phonetic units are highly variable, both with speakers and with neighboring phonetic units (the so-called co-articulation of sounds), it is assumed that the rules governing the variability are straight-forwards and can readily be learned and applied in practical situations. Hence the first step in the acoustic-phonetic approach to speech recognition is called a segmentation and labeling phase because it involves segmenting the speech signal into discrete (in time) regions where the acoustic properties of the signal are representative of one (or possibly several) phonetic units (or classes), and then attaching one or more phonetic labels to each segmented region according to the acoustic properties. To actually do speech recognition, a second step is required. This second step attempts to determine a valid word (or string of words) from the sequence of phonetic labels produced in the first step, which is consistent with the constraints of the speech-recognition task (i.e. the words are drawn from a given vocabulary, the word sequence makes syntactic sense and has semantic meaning, etc.).

The pattern-recognition approach to speech recognition is basically one in which the speech patterns are used directly without explicit feature determination (in the acoustic-phonetic sense) and segmentation. As in most pattern-recognition approach, the method has two steps – namely, training of speech patterns, and recognition of patterns via pattern comparison. Speech 'knowledge' is brought into the system via the training procedure. The concept is that if enough versions of a pattern to be recognized (a sound, a word, a phrase, etc.) are included in a training set provided to the algorithm, the training procedure should be able to adequately characterize the acoustic properties of the pattern (with no regard for or knowledge of any other pattern presented to the training procedure). This machine learns which acoustic properties of the speech class are reliable and repeatable across all training tokens of the pattern. The utility of the method is the pattern-comparison stage, which does a direct comparison of the unknown speech (the speech to be recognized), with each possible pattern learned in the training phase and classifies the unknown speech according to the goodness of match of the patterns.

The artificial intelligence approach to speech recognition is a hybrid of the acoustic-phonetic approach and the pattern-recognition approach in that it exploits ideas and concepts of both methods. The artificial intelligence approach attempts to mechanize the recognition procedure according to the way a person applies its intelligence in visualizing, analyzing, and finally making a decision on the measured acoustic features. In particular, among the techniques used within this class of methods are the use of an expert system for segmentation and labeling so that this crucial and most difficult step can be performed with more than just the acoustic information used by pure acoustic-phonetic; learning and adapting over time (i.e. the concept that knowledge is often both static and dynamic and that models must adapt to the dynamic component of the data); the use of neural networks for learning the relationships between phonetic events and all known inputs (including acoustic, lexical, syntactic, semantic, etc.) as well as for discrimination between similar sound classes.

Speech recognition systems are typically based on Hidden Markov Models (HMMs) as presented by Rabiner (1989), which are used to represent speech events (e.g. a word) statistically, and where model parameters are trained on a large corpus of speech data. Given a trained set of HMMs there exists an efficient algorithm for finding the most likely word sequence when presented with unknown speech data. The recognition vocabulary and vocabulary size play a key role in determining the accuracy of a system. A vocabulary defines the set of words or phrases that can be recognized by a speech engine. A small vocabulary system may limit itself to a few hundred words where as a large vocabulary system may consist of tens of thousands of words. Large vocabulary speech recognition systems typically use a sub-word approach where phonetic sub-word models are built instead of an explicit model for each word in the large vocabulary. Such systems also use a statistical language model that defines likely word sequences in a particular domain to provide statistical information on word sequences. The language model assists the speech engine in recognizing speech by biasing the output towards high probability word sequences. Together, vocabularies and language models are used in the selection of the best match for a word by the speech recognition engine. Dictation systems are typically large vocabulary applications, whereas navigation/transactional are typically small vocabulary applications. Multimedia indexing systems could be either large vocabulary or small vocabulary, depending on the specific application.

Dynamic time warping is an algorithm for measuring similarity between two sequences which may vary in time or speed. Kathana (2010) claims, that dynamic time warping is an approach that was historically used for speech recognition but has now largely been displaced by the more successful HMM-based approach. DTW has been applied to video, audio, and graphics – any data which can be turned into a linear representation can be analyzed with DTW. Especially in automatic speech recognition DTW is applied to cope with different speaking speeds. In general, it is a method that allows a computer to find an optimal match between two given sequences (e.g. time series) with certain restrictions, i.e. the sequences are "warped" non-linearly to match each other.

Speech recognition systems output the most probable decoding of the acoustic signal as the recognition output, but keep multiple hypotheses that are considered during the recognition. The multiple hypotheses at each time, often known as N-best word lists, provide grounds for additional information. Recognition systems generally have no means to distinguish between correct and incorrect transcriptions, and a word lattice representation (a directed acyclic graph) is often used to consider all hypothesized word sequences within the context. The nodes represent points in time, and the arcs represent the hypothesized word with an associated confidence level. The path with the highest confidence level is generally output as the final recognized result, often known as the 1-best word list. The N-best word lists are typically used by speech recognition applications to improve the usability and performance of the applications such as dictation systems and multimedia indexing systems.

Speech recognition accuracy is typically represented in terms of word error rate (WER), defined to be the sum of word insertion, substitution and deletion errors divided by the total number of correctly decoded words. The WER can vary dramatically depending on the nature of the audio recordings.

Key-word spotting is a subfield of speech recognition. Key-word spotting is the ability to use the automatic speech recognition engine in order to analyze natural day-to-day conversation between two or more people and indicate whether specific key-words were mentioned. That is, key-word spotting deals with the same problems as continuous speech recognition, namely the continuous spontaneous conversation between two or more speakers that can include an unlimited range of vocabulary. In addition, the speech can be from a noisy environment, various input types and include several speakers.

Various approaches exist for implementing a key-word spotting engine. Natural Speech Communication Ltd. (2005) discusses three underlying processes:

The *LVCSR based Key-Word Spotting* approach uses a two-stage process. In the first stage, the transcription of the speech into words is done using a Large Vocabulary Continuous Speech Recognition (LVCSR) engine, outputting formatted text. In the second stage, a textual search for the key-words within the text is performed. Using this approach, results from LVCSR and the text search are combined to spot the key-words.

The *Phoneme Recognition based Key-Word Spotting* approach transforms in the first stage the speech to a sequence of phonemes. In the second stage, the application searches for phonetically transcribed key-words in the phoneme sequence, obtained from the first stage.

The *Word Recognition based Key-Word Spotting* approach searches for the key-words in a one stage operation. The recognition is phoneme-based and the key-word spotting engine looks for the keyword in the speech stream based on a target sequence of phonemes representing the key-word.

4.3 Summarization of spoken language

While the field of summarizing written texts has been explored for many decades, gaining significantly increased attention in the last ten to fifteen years, summarization of spoken language is a comparatively recent research area. As a result a number of different methods and approaches have been proposed and a variety of systems have been built that can perform different summarization tasks on spoken language input. Although automatic speech recognition technology has matured over time, natural unconstrained scenarios present significant challenges to state-of-the-art systems. For example, spontaneous and realistic interaction, with often accented speech and specialized topics of discussion, as well as overlapping speech, interfering acoustic events, and room reverberation degrade significantly the automatic speech recognition performance. This section places summarization of spoken language in the context of general summarization research, describes its main challenges which are added on top of the already challenging area of written text summarization, and investigates approaches and spoken language summarization systems.

Waibel et al. (2001) claim that meeting recognition is a very challenging Large Vocabulary Continuous Speech Recognition (LVCSR) task. They identified main challenges that have to be addressed in spoken language summarization, in addition to the challenges of written text summarization, to cope with speech disfluencies (breaks, irregularities, e.g. phrases that are restarted or repeated, or 'fillers'), the identification of the units for extraction, and single vs. multiple speakers. Cross-speaker coherence (in case of multi-party conversations) has to be maintained by a spoken language summarization system and it has to cope with speech recognition errors. There are mainly three reasons for the difficulties: first, the conversational style – meetings consist of uninterrupted continuous recordings with multiple speakers talking in a conversational style. Second, the lack of training data – meeting data is highly specialized depending on the topic and participants, therefore large databases cannot be provided on demand. Third, the degraded recording conditions: to minimize interference a clip-on lapel microphone is often chosen instead of a close-talking headset. In order to continue processing of conversations with computer-linguistic methods, the challenges mentioned above have to be recognized and, if necessary, be removed. Not till then, traditional procedures for automatic text summarization can be applied to conversations.

In dialogues cross-reference words, so called anaphora, for example 'he', 'she', and 'it', or demonstrative pronouns, e.g. 'that' and 'these', are applied. EML Research (2005) states that without dissolving the different kinds of pronouns, conversations can't be summarized meaningfully. Speech-comprehending computer systems have difficulties with pronouns, as the systems are lacking in knowledge and relations of the said statement. Objective of research is that computers learn to integrate the pronouns in their context. As well as in automatic summarizing, quantitative or statistical approaches have to be applied, to detect patterns in the speech.

As significant advances in automatic speech recognition were made researchers looked into the question of *spoken language summarization in restricted domains*. The first applications in spoken language summarization were built in the context of speech-to-speech translation systems such as VERBMOBIL, developed by Reithinger et al. (2000) and JANUS, built by Lavie, et al. (1997). Since these systems all operate in very restricted domains, a limited text understanding approach is feasible and allowed, in the case of VERBMOBIL, the generation of abstracts in multiple languages from a single knowledge representation format.

In that context, several spoken dialogue summarization systems are developed, whose goal it is to capture the essence of the task based dialogues at hand. Kameyama (1984), (1996) presented the MIMI System that deals with the travel reservation domain and uses a cascade of finite state pattern recognizers to find the desired information.

The VERBMOBIL, a more knowledge-rich approach is used in the domain of travel planning and negotiation of a trip (Alexandersson & Poller, 1998). In addition to finite state transducers for content extraction and statistical dialogue act recognition, also a dialogue processor and a summary generator are used which have access to a world knowledge database, a domain model, and a semantic database. The abstract representations built by this summarizer allow for summary generation in multiple languages.

In the context of the DARPA Broadcast News workshops and TREC's Spoken Document Retrieval track (SDR), Garofolo et al. (1997), (1999) developed several systems for *summarizing spoken news*. The systems enable the multi-media browsing tools for text, audio, and video data, indexing, and retrieving of audio recordings, sometimes along with summarization of the contents, based on either human transcription or automatic transcription by automatic speech recognition technology.

Speech recognizer transcripts in tandem with the original audio recording system are used by Hirschberg et al. (1999) and Whittaker et al. (1999) to support local navigation for browsing and information extraction from acoustic databases,. While the system interface helps users in the tasks of relevance ranking and fact-finding, it is less helpful in the creating of summaries, due to imperfect speech recognition. More accurate and reliable summaries are obtained by an audio summarization system presented by Valenza et al. (1999), which combines acoustic confidence scores with relevance scores. Hori and Furui (2000) use salience features in combination with a language model to reduce Japanese broadcast news captions by about 30-40% while keeping the meaning of about 72% of all sentences in the test set.

While most approaches to summarizing of acoustic data rely on the word information (provided by a human or automatic speech recognition transcript), Arons (1997) uses *prosody-based detection in spoken* audio focuses on the acoustic signal and makes use of a variety of prosodic features to enable quick skimming/browsing. Passages are extracted which are prosodically emphasized, by training a Hidden Markov Model on transcriptions of spontaneous speech, labeled for different degrees of emphasis by a panel of listeners (Chen & Withgott, 1992). Stifelman (1995) uses pitch based emphasize detection algorithms, developed by Arons (1994), to find emphasized passages in a 13 minute discourse.

Zechner and Waibel (2000), (2001) summarize, for the first time, conversations of two or more parties for *spoken dialogues in unrestricted domains* in the summarization system (DIASUMM). The DIASUMM system addresses the issues of disfluency detection and removal and sentence boundary detection. Cross-speaker information is linked by the system. The components of the DIASUMM system are trained on a large corpus of disfluency annotated conversations (LDC, Linguistic Data Consortium, 1999) and tested on four different genres of spoken dialogues, so the automatic speech recognition word error rate is reduced, compared to previous systems in unrestricted domains.

Zechner (2002) states that as the amount of digitized speech available on-line will rise substantially, and more robust and improved speech recognition is required. There is an increasing interest in summarization of dialogues and audio documents of various kinds. Automatic speech recognition will have to be performed across a wide range of channel conditions, sometimes with substantial noise, cross-talk and other hard parameters. Current state-of-the-art speech recognizers still exhibit fairly high average word error rates of up to 40% for challenging genres such as multi-party meetings in noisy environments with participants using rather conversational speech.

4.4 Current research projects

In the following section an overview of current research projects in the area of automated speech summarization, indexing and retrieval is given. These projects are analyzed regarding externalization of informal meeting information.

4.4.1 Multimedia indexing and retrieval projects

Indexing applications provide the ability to retrieve relevant audio or video segments when presented with a textual query.

A well-known issue in spoken document retrieval is the concept of in-vocabulary terms and out-of-vocabulary terms. Since the speech engine matches the acoustics from the speech input to words in the vocabularies, only words in the vocabulary are capable of being recognized. Words in a vocabulary are recognized based entirely on their pronunciations. In addition, a word not in the vocabulary will often be erroneously recognized as an in-vocabulary word that is phonetically similar to the out-of-vocabulary word.

Chang et al. (1997) and Wactlar et al. (1999) state that multimedia indexing techniques for extraction of semantic information from unstructured video include a wide range of topics from computer vision, pattern recognition, video analysis and summarization, speech recognition, natural language understanding, and information retrieval,.

A more popular approach to video retrieval is to search the audio transcript using the familiar metaphor of free text search, presented by Jones et al. (1996). In this case, speech recognition is applied to the audio track, and a time-aligned transcript is generated. The indexed transcript provides direct access to the semantic information in the video.

The *Informedia* research project on multimedia retrieval at Carnegie Mellon University includes a strong speech retrieval component. Goal of the Informedia initiatives is to achieve machine understanding of video and film media, including all aspects of search, retrieval, visualization and summarization in both contemporaneous and archival content collections. Speech, language, and image understanding technology are combined to automatically transcribe, segment and index the linear video. The same tools are applied to accomplish intelligent search and selective retrieval. Informedia also seeks to improve the dynamic extraction, summarization, visualization, and presentation of distributed video, automatically producing “collages” and “auto-documentaries” that summarize documents from text, images, audio and video into one single abstraction. The process automatically generates various summaries for each story segment: headlines, filmstrip story-boards, and video-skims.

A highly accurate, speaker-independent speech recognizer is applied to automatically transcribe video soundtracks which are then stored in a (time-track corresponding) full-text information retrieval system. This text database in turn allows for rapid retrieval of individual corresponding "video paragraphs" which satisfy an arbitrary subject area query based on the words in the soundtrack. Another innovative concept is the implementation of "video skimming". This enables an accelerated viewing of the key video and audio sequences without the perceptual disturbance of simply speeding up the frame rate and audio. Thus a video abstract is created that conveys the essence of the content in 5 to 20% of the time.

The Informedia system provides full-content search and retrieval of current and past TV and radio news and documentary broadcasts. The system implements a fully automated process to enable daily content capture, information extraction and storage in on-line archives by applying artificial intelligence and advanced systems technology. The prototype database allows for rapid retrieval of individual video paragraphs which satisfy an arbitrary spoken or typed subject area query based on the words in the soundtrack, closed-captioning or text overlaid on the screen. There is also a capability for matching of similar faces and images.

Through the integration of technologies from the fields of natural language understanding, image processing, speech recognition and video compression, Christel et al. (1994) showed that the Informedia project allows the users to explore multimedia data in depth as well as in breadth. The Informedia digital video library project goes far beyond the current paradigm of video-on-demand, where a user can select one video from a limited set and view that video after a delay of a perhaps a few minutes. The computer adds no substantial benefit to this video-on-demand model over a VCR with each video on a tape; the user remains a passive observer of someone else’s produced material. By contrast, the Informedia Project segments hours of video into logical pieces and indexes these pieces according to their raw content (dialog, images, narration). The users can actively explore the information by finding sections of content relevant to their search, rather than by following someone else’s path through the material (as one does when using the current generation of educational CD-ROMs) or by viewing a large chunk of pre-produced material (as with video on demand).

The Informedia project covers three broad categories of technologies to create and search a digital video library built from broadcast video and audio materials (Hauptmann & Smith, 1995): *Text processing* looks at the textual (ASCII) representation of the words that were spoken, and at other text annotations that may be derived from the transcript, from the production notes, or from closed-captioning that is sometimes broadcast with the news stories. Mauldin (1989) claims that text analysis can work on an existing transcript to help segment the text into paragraphs. *Image analysis* looks at the images in the video stream. Image analysis is primarily used for the identification of scene breaks and to select static frame icons that are representative of a scene. Primitive image features based on image statistics, such as color histograms, are used by

Zhang et al. (1995) for indexing, matching and segmenting images. Color histogram analysis and Optical flow analysis are implemented in the News-on-Demand production system. *Speech analysis* provides the basis for analyzing the audio component of the news broadcast. Speech analysis operates only on the audio portion of the video. Using speech recognition a transcript can be obtained, although it may contain errors. To transcribe the content of the video material, the Sphinx speech recognition engine (CMU-Speech, 1995) is used, a large-vocabulary, speaker-independent, continuous speech recognizer created at Carnegie Mellon developed by Hwang et al. (1994).

Acoustic signal analysis is used to identify segment boundaries of paragraph size. Transitions between speakers and topics can be detected which are marked by silence or low energy areas in the acoustic signal. To detect breaks between utterances a Signal to Noise ratio (SNR) computation is applied. This algorithm computes the power of digitized speech samples where each s_i is a pre-emphasized sample of speech gathered over a twenty millisecond frame. A low power level indicates that there is little active speech occurring in the frame. Segmentation breaks between utterances are set at the point of minimum power after smoothing over a one second window.

Hauptmann and Witbrock (1997) distinguish two distinct phases during News-On-Demand processing: library creation and library exploration. Library creation deals with the accumulation of information, transcription, segmentation and indexing. Library exploration concerns the interaction between the system and the user trying to retrieve selections in the database.

To transcribe the content of the video material, spoken words are recognized with the Sphinx-II speech recognizer. The CMU Sphinx-II system uses semi-continuous Hidden Markov Models to model context-dependent phones (triphones), including between word context (Hwang, et al., 1994). The recognizer processes an utterance in three steps: It makes a forward time synchronous pass using full between word models, Viterbi scoring and a trigram language model. This produces a word lattice where words may have only one begin time but several end times. The recognizer then makes a backward pass which uses the end times from the words in the first pass and produces a second lattice which contains multiple begin times for words. An A* algorithm is used to generate the best hypothesis from these two lattices. The language model consists of words (with probabilities), bigrams/trigrams, which are word pairs/triplets with conditional probabilities for the last word given the previous word(s). Rudnicky (1995) constructed the language model from a corpus of news stories from the Wall Street Journal from 1989 to 1994 and the Associated Press news service stories from 1988 to 1990. Only trigrams that are encountered more than once are included in the model, but all bigrams and the most frequent 58800 words in the corpus are included.

Processing the video tape, using the speech recognition system, results in a transcript. This transcript contains errors, which, depending on the quality of the tape and the subject matter, currently range from 20% to 70% word error rate. Regarding automated meeting protocol generation from videos the word error rate would be too high, using the Informedia system, as too many errors would be in the generated protocols.

Although Informedia is intended for information retrieval and does not generate text summarizations (protocols) out of the video, the Informedia project realizes a combination of language understanding and speech recognition and applies an open-source, speaker independent speech recognizer, which could be promising in the application of protocol generation.

The *Medusa* network multimedia system, developed at Cambridge University in collaboration with Olivetti Research Laboratory, is also an information retrieval system, based on word-spotting. Medusa uses a high-speed asynchronous transfer mode network for a video mail application using a 35-word indexing vocabulary for word-spotting, chosen a priori for the specific domain. The retrieval methods developed by Jones et al. (1995) are based on spotting keywords in the audio sound track by integrating speech recognition methods and information retrieval technology to yield a practical audio and video retrieval system. In the applied phone-lattice scanning approach, the speech recognition system generates a generalized sub word, or phone lattice. Spoken words are decomposed into a sequence of phone units and the precomputed lattices are scanned for phone strings corresponding to the query word.

As Medusa is only an information retrieval system, it is not applicable for meeting protocol generation, although the keyword spotting approach can be realized in an automated protocol generator, searching for predefined meeting elements.

4.4.2 Speech mining research projects

The multimedia indexing and retrieval applications are aimed primarily at domains where the data is professionally produced audio or video, stored on tape or disk, and the main problem is to provide users with the ability to quickly find a particular clip, fact, or piece of information. Some systems provide additional functionality, including automatic summarization, clustering, classification, and organization of the recorded information, e.g. the Informedia system. These systems typically operate off line, require two to ten times real time to analyze the recorded data, and generally assume high-quality produced recordings.

Although these systems can be invaluable for managing recordings, such as broadcast news or corporate training videos, they are not directly amenable to spoken discourses i.e. spoken conversation between two or more individuals that takes place anywhere, anytime. Spoken discourse is a rich source of tacit information, and often the only form in which the tacit information is instantiated.

Technologies and applications attempt to capture spoken discourse, convert the discourse to text, apply text analysis to the discourse, and exploit the knowledge discovered in the discourse, for instance in an informal meeting.

At IBM Thomas J. Watson Research Center, Coden and Brown (2001) investigated how to capture spoken discourse and analyze it in real time is done, to both extract knowledge from the discourse and provide additional, related knowledge to the discourse participants. The need for this capability is driven by three separate applications: meeting support, data broadcasting, and call mining. Meeting support in this context includes the ability to understand the current meeting discussion and automatically provide related information to the meeting participants. Data broadcasting is the process of exploiting the unused bandwidth in a television broadcast to send arbitrary data with the television program. In particular, the data should be related to the current television program and provide an enhanced viewing experience for the user, enriching the audio and visual television program with related facts, articles, and references. Call mining is an effort to analyze and index the telephone calls made or taken by customer service representatives at call centers and help desks. As understanding meeting discussions, as well as analyzing speech, is an integral part of the externalization of knowledge from a meeting, the IBM application is considered in more detail. The meeting support and data broadcast applications share a common need for the ability to analyze speech in real time and automatically discover relevant, collateral information.

The *WASABI* system, built at Watson, is a generic framework for analyzing speech. WASABI takes speech audio as input, converts the audio stream into text using a speech recognition system, applies a variety of analyzers to the text stream to identify information elements, automatically generates queries from these information elements, and extracts data from the search results that are relevant to the current discourse.

Input to the system is a raw data stream, i.e. the captured audio of the discourse. The speech recognition system IBM ViaVoice converts the audio into a text stream, which WASABI feeds to one or more analyzers. An analyzer performs a text analysis procedure on its input and produces an output that may be fed to another analyzer or multiplexed back into the original data stream. The task performed by each analyzer depends on the application in which the framework is applied. One of the more important tasks performed by an analyzer is to automatically create a query from the input, use the query to search a relevant knowledge repository, and extract relevant information from the search results that will enhance the input data stream.

Regardless of the task, the analysis must take place in real time, or fast enough to keep up with the incoming data stream. The final enriched data stream is presented to the user in an appropriate user interface, or archived to a data store for indexing and future reference.

Brown et al. (2001) claim that meetings often suffer from the problem that during the meeting the participants do not have convenient access to all of the knowledge resources that might be used to facilitate or enrich the meeting. These resources might be as simple as someone's phone number, or they might be more complex, such as a project database that identifies who is working on what and where expertise on a particular topic can be found. The WASABI framework is being applied to solve this problem in a system called *MeetingMiner*. MeetingMiner is essentially an agent that passively captures and analyzes the meeting discussion and periodically becomes an active participant in the meeting, whenever it finds information that it determines is highly pertinent to the current discussion. The main input to the system is an audio stream generated by one or more microphones that capture the spoken discourse of the meeting. The audio stream is converted to a text transcript by the speech recognition system, and the text transcript is processed by the meeting analyzers.

The current set of meeting analyzers includes a *named entity recognizer*, a *topic tracker*, and a *question identifier*. The named entity recognizer identifies proper names in the text, such as people, places, and organizations. It is based on the algorithms developed in the Textract system by Ravin et al. (1997), and it uses a combination of lexical clues (e.g. capitalization patterns and punctuation) and dictionary lookups to identify named entities. The system uses the identified names of people to search an employee database and retrieve address, phone number, group affiliation, project responsibilities, and expertise information. This information is assembled in a custom address book as the meeting progresses, providing instant access to information about any individual who may be mentioned during the course of the meeting.

The topic tracker uses a combination of automatic text classification and statistical term frequency analysis to identify keywords in the text. The text classification system analyzes a sliding window of sentences and classifies the window content into a predefined taxonomy of topics. The current topic combined with the identified keywords is used to search related knowledge repositories and provide a continuously updated "hit list" of objects that may be relevant to the meeting. For example, during a technical design meeting, the topic tracker might send topic and keyword queries to a database of patents and alert the meeting participants whenever a highly relevant patent is found. This capability can enhance the design session with relevant information and help prevent time being spent on solutions that already exist, or are owned by competitors.

The question identifier uses regular expressions to identify various kinds of questions in the text transcript. In particular, the system identifies who, what, when, where, why, and how questions and feeds them to a question-answering system (Prager et al., 2000). The question-answering system parses the question and returns a concise phrase or small number of sentences that answer the question. This capability works best with questions seeking factual answers (e.g., "When are the budget numbers due?"), as opposed to more open-ended questions (e.g., "How can we improve profitability?"). The effectiveness of the question-answering component is limited by the information available for answering questions. Prager et al. (2000) state that a question-answering system can be created by automatically processing free-text documents, allowing any relevant collection of documents (corporate memos, e-mail messages, reports, etc.) to be included in the question-answering system.

An automated protocol generator could reuse the idea of a named entity tracker, for example to detect the person assigned to a new task, and the topic tracker of WASABI, to identify a protocol entry. However, as the topic tracker works on a statistical term frequency, most protocol entries will probably be missed, because the critical information, e.g. the phrase with the decision, or the assignment of a task, will be spoken only once and not recognized by WASABI. For this reason, the WASABI system does not generate a protocol of the meeting.

At Carnegie Mellon University (CMU), Waibel et al. (1998) developed the *Meeting Browser*, a system for capturing, indexing, searching, and browsing meetings. The work focuses on building speech recognition models suitable for the speaking modes found in meetings and applying post processing steps on the speech recognition transcripts to generate summaries. The use of visual cues captured by video camera is explored to aid in tracking a discussion and to provide enhanced browsing capabilities. Waibel et al. (2007) showed that the Meeting Browser provides functionality for offline reviewing of recorded meetings, automatic analysis, summarization or data reduction, generation of minutes, topic segmentation, and information querying and retrieval of human interactions.

The system consists of four major components:

1.) the speech transcription engine,
2.) the summarizer, a statistical tool that attempts to find salient and novel turns,
3.) the discourse component that attempts to identify the speech acts, and
4.) the non-verbal structure, including speaker types and non-verbal visual cues.

The speech recognition component of the meeting browser is based on the JANUS Switchboard recognizer. The gender independent, vocal tract length normalized, large vocabulary recognizer features dynamic, speaking mode adaptive acoustic and pronunciation models.

In spontaneous conversational human-to-human speech, as observed in meetings, there is a large amount of variability due to accents, speaking styles and speaking rates (also known as the speaking mode). Because recognition systems usually use only a relatively small number of pronunciation variants for the words in their dictionaries, the amount of variability that can be modeled is limited. The meeting browser incorporates a probabilistic model based on context dependent phonetic rewrite rules to derive a list of possible pronunciations for all words or sequences of words. In order to reduce the confusion of this expanded dictionary, each variant of the word is annotated with an observation probability. To this aim the corpus based on all allowable variants using flexible utterance transcription graphs and speaker adapted models is automatically retranscribed. The alignments are used to train a model of how likely which form of variation (i.e. rule) is and how likely a variant is to be observed in a certain context (acoustic, word, speaking mode or dialogue).

The summarizer component produces condensed informative summaries of the meeting. The summaries are created using a statistical approach, whereby salient, relevant and informative passages are flagged and selected from a meeting. As a first metric for selecting informative passages from a human dialog, the Maximal Marginal Relevance (MMR) metric is explored. The MMR iteratively maximizes the similarity between a query and each section of a document while it minimizes the similarity among previously ranked document sections. It thereby identifies the most relevant, yet most diverse, non-redundant sections of a document. Here, a modified version of a MMR is applied to conversational dialogue to find the most relevant, non-redundant turns in a meeting transcript. The top N turns are presented to the user in the original order of the meeting transcript as a summary of the meeting. The summarization algorithm takes as input a textual transcript that is generated manually or from an actual speech recognition run. The algorithm can be divided into the following steps:

1. Eliminate all stop words from consideration
2. Identify the most common stems from the set of remaining words
3. Weight each turn or utterance
4. Include the highest weighted turn in the summary
5. Eliminate the most common stem word and the included turn from consideration
6. If a preset summary size has not been reach, go to step 2.

An important aspect of generating meeting summaries or minutes is the successful and efficient delivery of the result. The meeting browser allows the user to review and browse transcribed and summarized meetings. The browser also includes video capture of the individuals in the meeting for use in meeting rooms or video conferencing. In addition to being a tool for browsing and viewing meeting records, the meeting browser also attempts to provide tools for more rapid and informative access of key events in the meeting. The meeting browser interface displays meeting transcriptions, time-aligned to the corresponding sound and video files. The user can select all or a portion of these files for playback. Text highlighting occurs in sync with the sound and video playback.

Although the Meeting Browser generates a summarization of the meeting, the summaries are based partly on a manually generated transcript, i.e. the protocol generation process is not fully automated. The summaries are generated using only a statistical approach to extract relevant, not-redundant phrases. Available information regarding the meeting and the meeting elements, as presented in our meeting taxonomy, is not included in the summarization process of the Meeting Browser system. The incorporation of the meeting information would improve the quality of the summaries, as the most relevant phrases can be externalized. The generated summarizes are only available within the Meeting Browser, a connection to other project management tools, applied in the project, is not realized.

In the next chapter the protocol generation framework STACHUS is presented that addresses the improvement ideas of incorporating available meeting information, enhancing a pure statistical approach for automated protocol generation, and providing the meeting results to task management tools, allowing an effective meeting follow-up.

5 STACHUS

A framework for meeting report generation

The STACHUS framework for protocol generation supports the meeting workflow in externalizing formal and informal meeting communication. It is a framework for information retrieval and summarization of meeting conversation, incorporating existing speech recognition engines and protocol generation techniques, presented in chapter 4. The framework focuses on knowledge externalization and preparation of information in protocols.

The aim is to actively reuse the project context in terms of old meeting protocols, the system model, e-mails, chats and other project documentation to enhance the speech recognition engine and thus the protocol quality. STACHUS summarizes meetings by a semantic analysis. It is based on the meeting taxonomy presented in chapter 3.2. This allows the framework to create accurate and complete meeting protocols that comprise only the important parts of the meetings, as summarized in chapter 3.2.3.

STACHUS also provides connectivity for existing project management tools. That is, the results of the meeting, for instance assigned action items or issues, are automatically transferred to a task or project management tool, where the discussed information from the meeting are directly stored in the individual tool of the project team. So traceability of decisions is improved and the meeting workflow and procedure is supported, as well as the completion of assigned action items.

The framework consists of a compiler generator and a protocol engine. The protocol engine is dynamically generated by the compiler generator.

Section 5.1 describes the requirements, including a scenario of a meeting and a use case model. The underlying meeting grammar is presented in section 5.2. Section 5.3 presents dynamic and object models of meeting progresses and the protocol generation process automated by STACHUS. The components of the framework are described in section 5.4 and the software architecture is presented in section 5.5.

5.1 Requirements

5.1.1 Meeting workflow – a scenario

This section describes a meeting scenario in a Scrum-based project. A daily status meeting is presented, where protocols are automatically generated by the STACHUS protocol generator and the lightweight character of the Scrum meeting is emphasized.

Meeting participants are the software developers Tom, Joshua, Ben and their Scrum Master Michael. It's 9:30 am, the team members are assembled in a meeting room and they are ready to start their daily Scrum meeting.

Michael starts the audio recording by turning on the installed recorder in the meeting room, while the team member check either their pluggable microphone or one hand microphone for proper operation. Then Michael welcomes the team and asks Tom to start with his report, about "what he did yesterday", "what he will do today", and if there "are any problems or impediments" keeping him from doing his work.

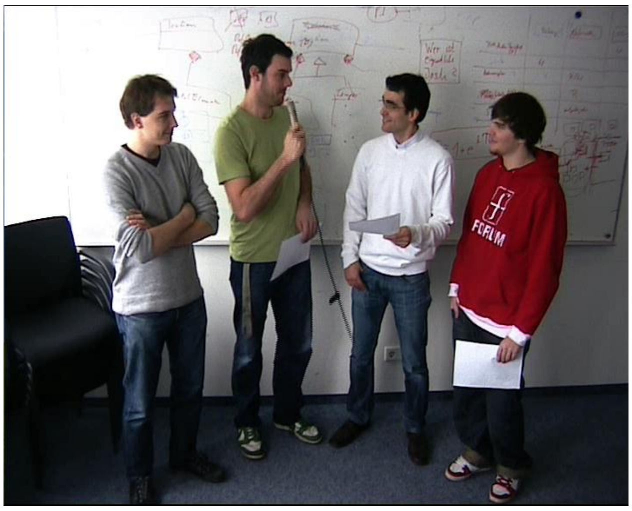
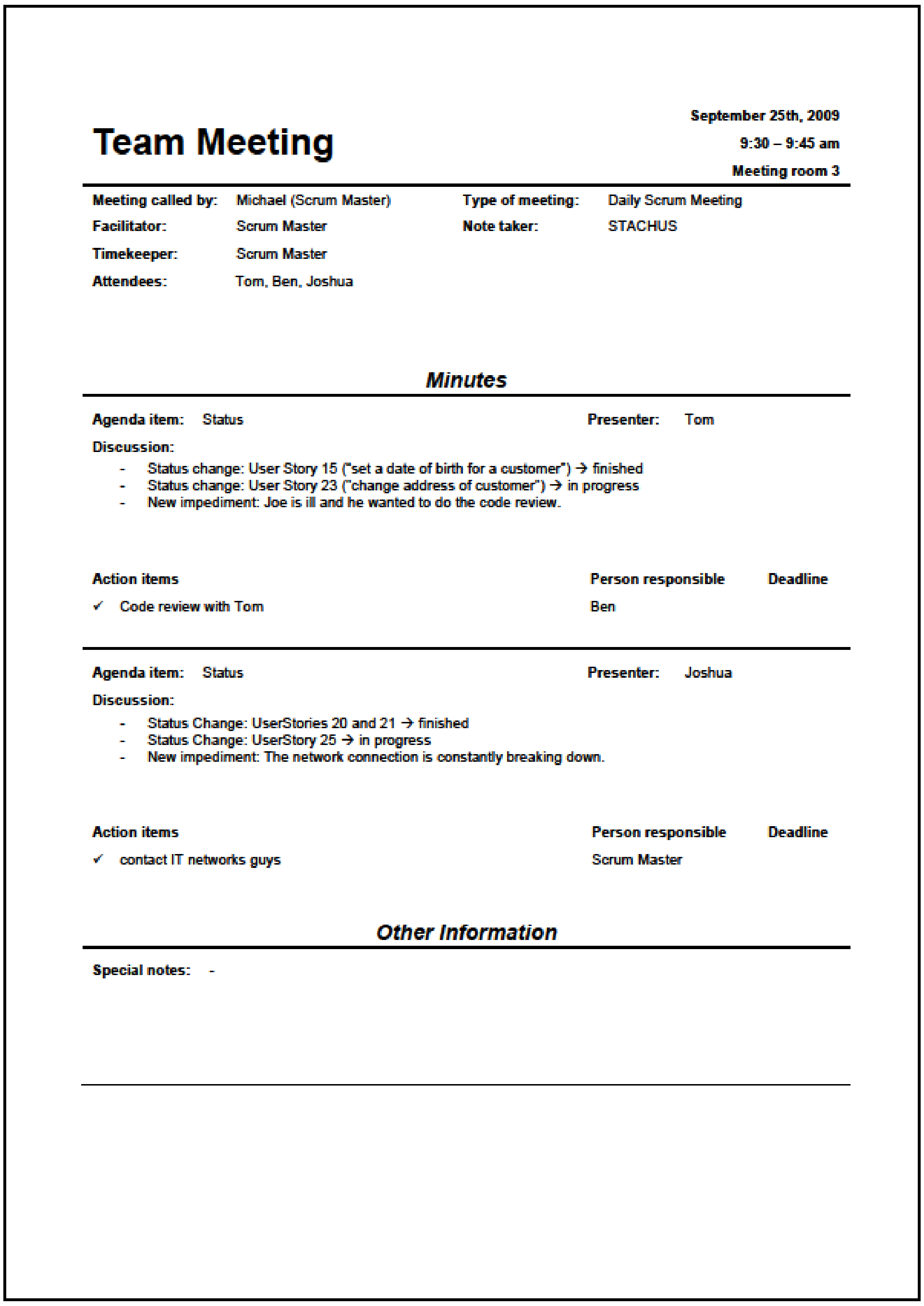


Figure 5-1: Daily Scrum Meeting

Tom informs the team that he has finished User Story 15 ("set a date of birth for a customer") yesterday, and started User Story 23 ("change address of customer"). Tom also mentions that he has an impediment that prevents him to finish the current User Story 23. Joe is ill and he wanted to do the code review. Michael asks, if Ben could help Tom. Ben agrees with this suggestion, and the Michael summarizes the new action item for Ben: "Code review with Tom."

Afterwards, its Joshua's turn to inform the team what he did yesterday and will do today. Joshua tells that he finished the User Stories 20 and 21. This morning he started with User Story 25. And he has a new impediment; the network connection is constantly breaking down.



Team Meeting

September 25th, 2009
9:30 – 9:45 am
Meeting room 3

Meeting called by:	Michael (Scrum Master)	**Type of meeting:**	Daily Scrum Meeting
Facilitator:	Scrum Master	**Note taker:**	STACHUS
Timekeeper:	Scrum Master		
Attendees:	Tom, Ben, Joshua		

Minutes

Agenda item: Status **Presenter:** Tom

Discussion:

- Status change: User Story 15 ("set a date of birth for a customer") → finished
- Status change: User Story 23 ("change address of customer") → in progress
- New impediment: Joe is ill and he wanted to do the code review.

Action items	**Person responsible**	**Deadline**
✓ Code review with Tom	Ben	

Agenda item: Status **Presenter:** Joshua

Discussion:

- Status Change: UserStories 20 and 21 → finished
- Status Change: UserStory 25 → in progress
- New impediment: The network connection is constantly breaking down.

Action items	**Person responsible**	**Deadline**
✓ contact IT networks guys	Scrum Master	

Other Information

Special notes: -

Figure 5-2: Meeting minutes of the daily Scrum meeting

The Scrum Master is responsible to smooth these impediments out; he assigns himself a new action item to contact the IT network guys.

After all team members informed each other about their current status, the Scrum Master closes this daily Scrum meeting and stops the recording.

Subsequently, the team members go back to their workplace and can immediately read the protocol of the just finished meeting (see Figure 5-2). Moreover, all new action items, impediments and status changes of finished or in progress user stories are updated in their team project management and task-list maintenance tool, respectively.

Based on the given scenario, we present a more abstract description of the meeting workflow. The process of data detection during a meeting is shown in Figure 5-3, based on the usage of a key word spotting approach of meeting elements, like 'new issue', 'new action item' or 'decision'.

At the beginning of a meeting, an audio recorder is started. The audio stream is processed by a speech recognition engine that converts the audio stream into text phrases. Afterwards the output is searched through for pre-defined keywords of the meeting grammar, the meeting elements. If a keyword is found, the kind of keyword indicates the further process steps, carried out afterwards, i.e. the keyword is categorized. We take for instance the daily status meeting, as described in the scenario above. There, each team member presents its current work progress, whether a task corresponding to a requirement could be completed, is still in progress, or a new task was opened. Additionally, impediments are reported and new action items are defined and assigned to a team member. Based on the defined meeting taxonomy in chapter 3.2.1, there are special keywords for a daily status meeting to recognize status changes of a task, new action items, and new impediments, which are summarized in the meeting grammar (see section 5.2).

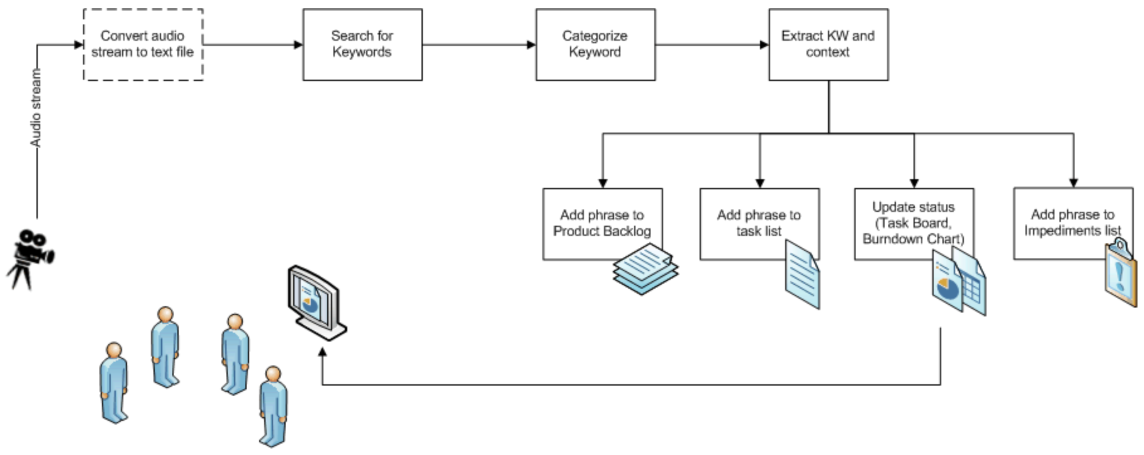

Convert audio stream to text file
Search for Keywords
Categorize Keyword
Extract KW and context
Audio stream
Add phrase to Product Backlog
Add phrase to task list
Update status (Task Board, Burndown Chart)
Add phrase to Impediments list

Figure 5-3: Meeting workflow

If a predefined keyword is found, the keyword and its context are extracted and the information is, depending on the keyword, processed. For instance, a detected new action item is stored in a task list, issues are sent to the impediments list and additionally, decisions, as well as action items and issues, are prepared and documented in the protocol. Moreover, the information is transferred to project and task management tools, and reports and protocols are automatically generated, containing a summary of the meeting, as defined in chapter 3.2.

In the following we present the meeting model used in the framework.

5.1.2 Meeting and protocol generation models

Meetings are attended by meeting participants, who attend a meeting and read a protocol subsequent to the meeting that summarizes the discussed topics.

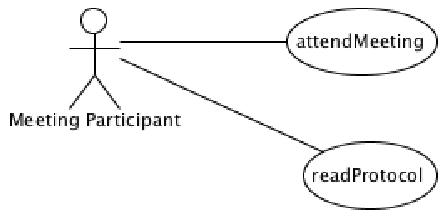


Figure 5-4: Meeting participation (UML Use Case Diagram)

During meetings, problems are discussed, new action items assigned, statuses presented, information shared, decisions are made, and knowledge is created.

Each meeting consists of meeting elements. In the case of a daily Scrum meeting, for instance, a team member reports about *status changes*, whether he started or finished a task. Moreover, the team member informs the other colleagues about *new issues* that prevent him from doing his work properly. *New action items* are assigned to individual team members, if necessary.

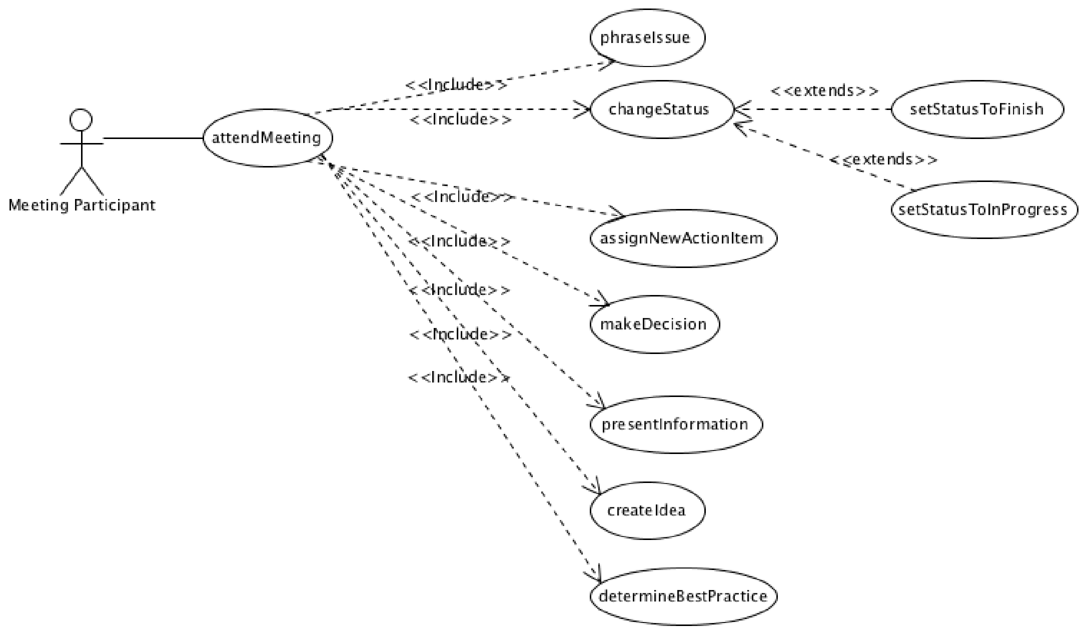


Figure 5-5: Meeting activities (UML Use Case Diagram)

A meeting summary has to be created at the end of the meeting containing all discussed information (e.g. status changes, new issues or action items) to externalize the knowledge of that meeting.

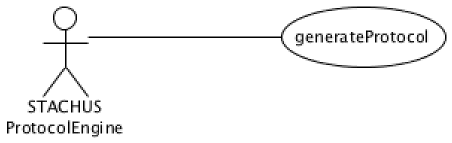


Figure 5-6: Protocol generation (UML Use Case Diagram)

The process of protocol generation incorporates updating the task lists and the Burndown Chart, as well as the creation of other reports or documents (like the Sprint Backlog), as illustrated in Figure 5-7.

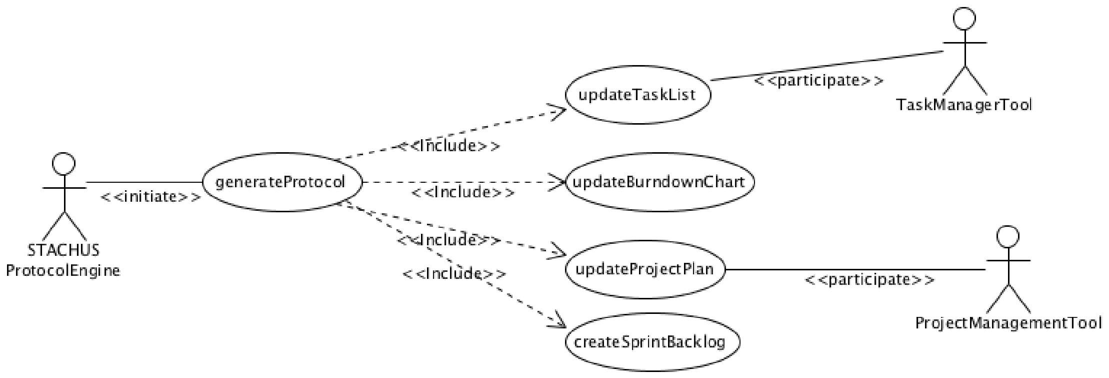


Figure 5-7: Protocol generation and report updating (UML Use Case Diagram)

Protocol generation is based on recordings of the meeting conversation. The STACHUS Audio Recorder captures the meeting communication and allows the STACHUS Protocol Generator to externalize the information of the meeting and summarize them in a protocol afterwards. The audio recording is started and stopped by a meeting participant (see Figure 5-8).

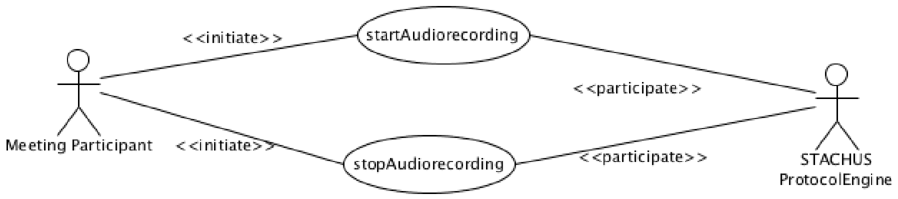


Figure 5-8: Audio recording (UML Use Case Diagram)

5.1.3 Functional requirements

A meeting management framework, supporting the automatic generation of meeting minutes, and the automatic forwarding of information to project tools, like a to-do manager, has to fulfill several functional and nonfunctional requirements to provide a useful tool for the meeting participants in an agile and traditional process environment. The framework has to be modular and flexibly extensible to incorporate additional or updated speech recognition engines and project management tools, and a dynamically self-improving system to constantly improve the protocol quality.

The functional requirements for a meeting management can be classified into three main categories. First the framework must provide necessary recording functionalities and storage of the verbal meeting content. Second, the audio stream has to be prepared to allow searching for tacit knowledge that will be externalized in the protocol. Third, the protocol generation and sending, as well as an additional storage, e.g. in a task-list manager, of the information discussed in the meeting have to be provided by the framework.

FR1: Recording of meetings
The meeting management framework has to provide necessary recording functionalities, including interfaces to an audio recorder to record meeting conversations.

FR2: Information externalization
The framework has to extract the 'important' information from the meeting conversation recordings. Therefore, the framework has to distinguish, which parts of the conversation are important, namely those that will be documented in the protocol later, like new action items, decisions, problems, information, etc. This filtering is done on the basis of a flexible and extensible grammar, which is part of the framework.

FR3: Protocol generation
The meeting management framework has to automatically generate meeting minutes to document the volatile information of the meeting, as simple text files, Excel-documents, and as lists. Moreover, beside a protocol, additional reports have to be created or updated by the protocol generator, for instance a Product Backlog has to be filled, or a Sprint Backlog be updated, the Burndown Charts be refreshed, and project plans have to be updated.

FR4: Tool connection
If the team applies electronic project management tools (e.g. for the management of tasks or project planning) the STACHUS framework has to implement an interface to these tools, to allow information forwarding. That is, statuses and estimates have to be updated, new action items can be created, assigned and their deadlines can be determined. Moreover, finished tasks have to be ticked off in the task list, priorities of Backlog items can be changed, and an impediment list has to be expanded.

5.1.4 Nonfunctional requirements

An effective and efficient application of a framework for automatic report generation in the project lifecycle involves also non-functional requirements, listed in the following.

NFR1: Support for agile lightness
The framework should support an *easy set up* and recording a meeting without additional effort. The framework has to allow the users to easily familiarize with the system and ensure a *simple handling*, for instance only one button to start and stop the recording without any need for further configuration. The system has to be operated with as *little effort* as possible to support the agile idea of eliminating waste.

NFR2: Correctness
One of the problems when writing a protocol manually, as described in chapter 3, is that the protocol is often incomplete or incorrect. The meeting minutes' taker can't always concentrate on listening and writing at the same time – mistakes can easily creep in. Consequentially, an important requirement for an automated meeting management framework is that the protocol generator produces correct and complete outputs, i.e. all mentioned meeting elements are recognized and summarized in the protocol.

NFR3: Timely delivery
Meeting minutes and reports have to be generated immediately after the meeting by the framework and provided to all team members and further interested persons. Aim is to support the meeting follow-up process as soon as possible with the meeting protocol. That is, the automated protocol generation has to be faster than manually written protocols and with less effort.

NFR4: Non-intrusiveness
A smooth meeting workflow has to be supported by the framework. That is, the meeting may not be disturbed by the audio recording or automatic protocol generation. Moreover, the audio recording (starting and stopping of the recorder) has to be easy to handle, i.e. without additional effort and has to be intuitive, i.e. no comprehensive training is necessary to start and stop the protocol generation process. Thus an easy meeting communication has to be facilitated by the framework, where the meeting participants can concentrate on the communication and are not constrained by the automated protocol generation.

NFR5: Availability
The framework has to be designed to allow a meeting at any time and any place. So the flexible character of an agile team, having meetings at any time and everywhere is supported.

NFR6: Extensibility
The framework has to provide the ability to modify and add context. In particular, the extensibility of new key words, supporting additional meeting types, has to be provided. Moreover, the framework has to allow a flexible enhancement to support new technologies, for instance in the area of speech recognition.

5.2 STACHUS Grammar

In this section a rhetorical model is introduced, which is applied in the meeting management and protocol generation framework.

Continuous, spontaneous conversations between two or more speakers include a potentially large vocabulary. In addition, meetings can include several speakers or are conducted in a noisy environment which complicates speech recognition and results in a reduced word recognition rate.

A formal language is 'mapped' to the conversation of a meeting to address these problems. Based on the results of our meeting taxonomy, where each meeting element that is a potential candidate for the protocol is characterized by a dedicated key word – for instance "action item" for the process of defining an action item – a rhetorical model is defined. The rhetorical model includes a Chomsky-2 grammar with predefined terminal symbols. The STACHUS grammar consists of a set of rules that describe how to form strings from the language's alphabet that are valid according to the language's syntax.

The STACHUS grammar G for the rhetorical model consists of

- a finite set N of nonterminal symbols,
  the meeting elements, e.g. <action item>, <problem>, <information>
- a finite set T of terminal symbols, where $\mathrm{N} \cap \mathrm{T} = \emptyset$
  e.g. "new action item", "issue"
- a finite set P of production rules with a left and right-hand side consisting of a sequence of these symbols: $(\mathrm{N} \cup \mathrm{T})^*\mathrm{N}(\mathrm{N} \cup \mathrm{T})^* \rightarrow (\mathrm{N} \cup \mathrm{T})^*$
  e.g. $< problem > \rightarrow "issue"$
- and a start symbol S, a meeting phrase, where $\mathrm{S} \in \mathrm{N}$.

The terminal symbols correspond to words in the vocabulary, spoken during a meeting. The nonterminal symbols correspond to phrases or sentences formed using words from the vocabulary. Terminal symbols may appear in isolation ("task") or as sequences of words separated by whitespace characters ("action item"). During the project lifecycle, new rules and terminals can be added dynamically, extending the grammar. This allows to enhance the speech recognition and thus the protocol generation. This process of continuous improvement is realized in the STACHUS Framework Compiler – a detailed description is given in sections 5.4.2.

The terminal and nonterminal symbols depend on the context of the project. A core grammar and a core vocabulary (Figure 5-9) are used to cover a basic set of meeting phrases.

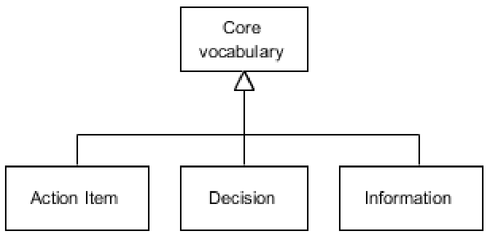


Figure 5-9: Core vocabulary elements

Moreover, extensions are identified, for instance the team members (address book entries), as well as the management method and its vocabulary, e.g. "retrospective" or "user story", and the system model and domain vocabulary, e.g. "Cocoa", or "touch screen". The core vocabulary, project management vocabulary, address book entries, and the system model are part of the STACHUS grammar, as visualized in Figure 5-10.

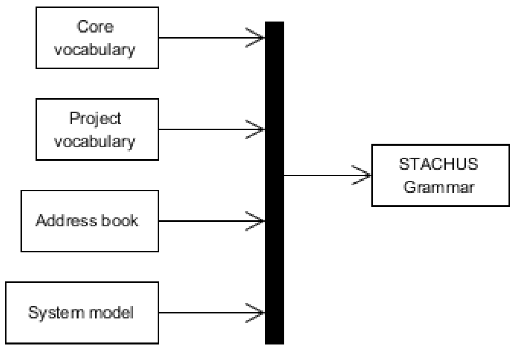


Figure 5-10: Project context flows into the grammar (UML Activity Diagram)

The project vocabulary consists of a set of vocabulary that is independent of the applied project management method, as well as vocabulary that is based on the applied project management method, e.g. Scrum vocabulary. So the nonterminals are grouped in a project independent set (e.g. the <date> for the next meeting) and a project dependent set, e.g. <Backlog item> or <unit test> (Figure 5-11).

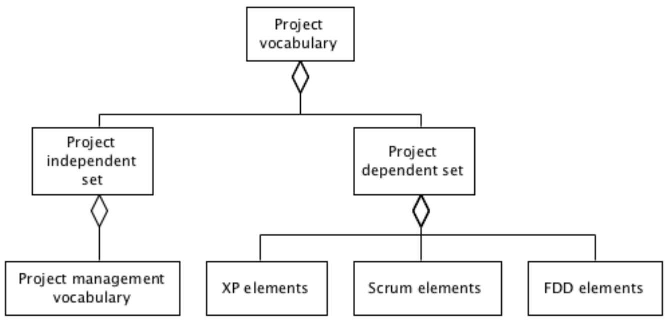


Figure 5-11: Project management vocabulary

The project management vocabulary is based on IBIS (Kunz & Rittel, 1970) and consists of meeting elements that are independent of the applied project management method. Examples are shown in Figure 5-12.

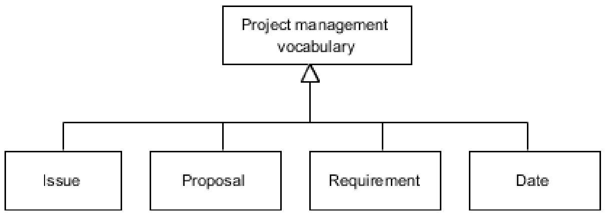


Figure 5-12: Project management vocabulary elements

Each class of meeting elements, as defined above, can be instantiated, representing the synonyms that are spoken in a meeting. Figure 5-13 shows the instances of "Action Item".

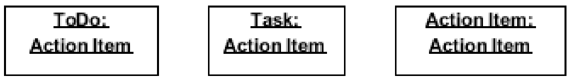


Figure 5-13: Project management vocabulary elements and its synonyms

The project dependent set consists of nonterminals that are dependent of the applied project management method. The nonterminals for Extreme Programming, Scrum, and Feature Driven Development are shown in Figure 5-14.

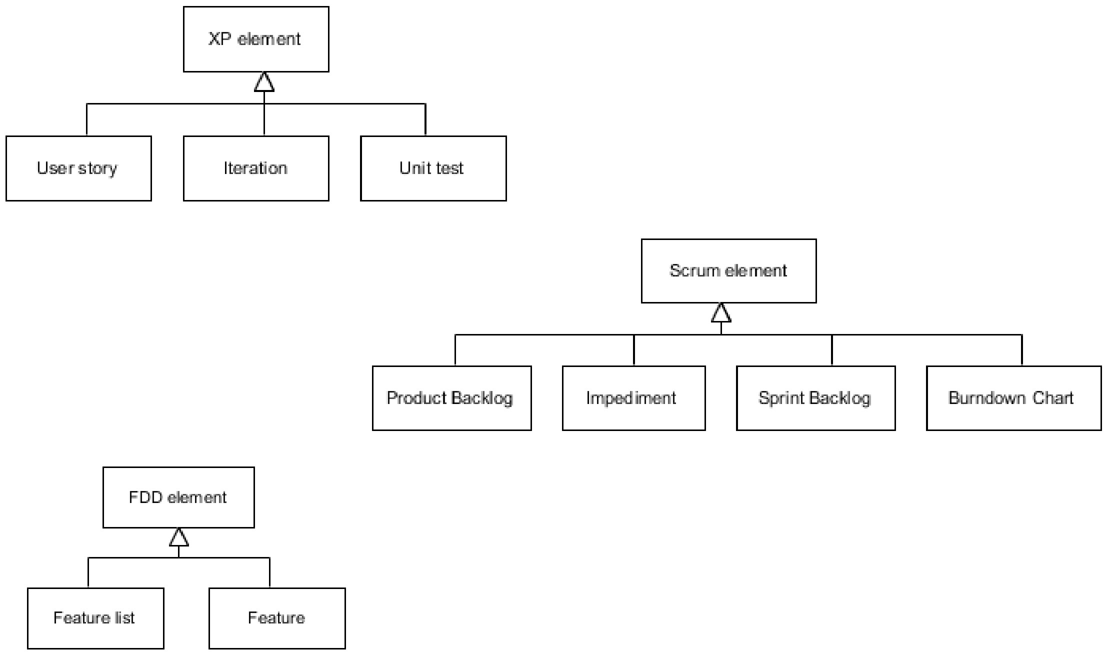


Figure 5-14: Decomposition of project management method dependent vocabulary

An example of a grammar, which was used during meetings of a student project at the Technische Universität München during the summer term 2009, is attached to the Appendix I.

5.3 STACHUS protocol generation process

An automatic protocol generation workflow starts, when a meeting participant starts audio recording. The Protocol Engine records the meeting audio stream and generates a protocol from the output. This process is illustrated in the sequence diagram in Figure 5-15.

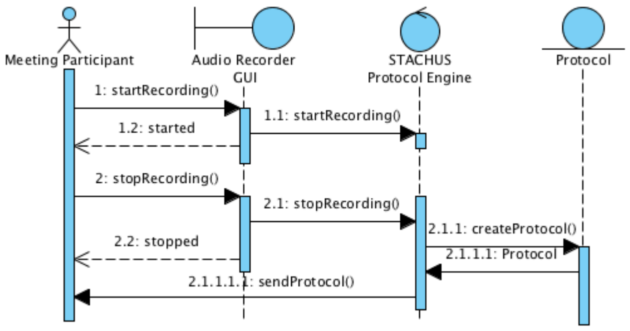


Figure 5-15: Protocol Generation (UML Sequence Diagram)

In the sequence diagram, the actor "Meeting Participant" is visualized, who starts the recording via the audio recorder GUI (the boundary object). After the recording is stopped by the actor, the STACHUS Protocol Engine (the control object) creates the "Protocol" (the entity object). Figure 5-16 gives a detailed representation of the protocol generation process.

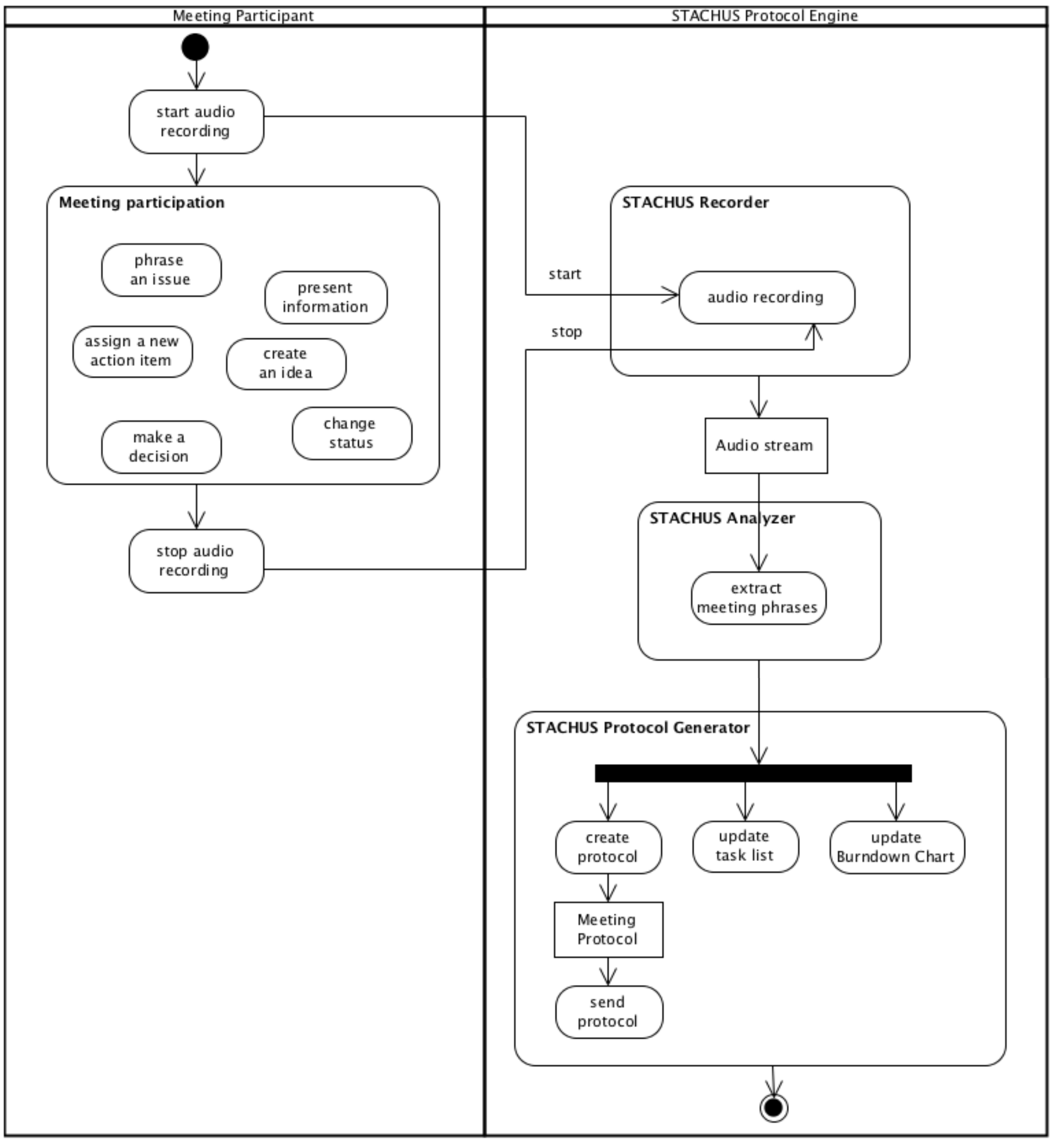


Figure 5-16: Protocol Generation Process (UML Activity Diagram)

The automatic protocol generation process, done by the STACHUS Protocol Engine, gets a current meeting as input and generates a meeting protocol for this meeting. This workflow is summarized in Figure 5-17.

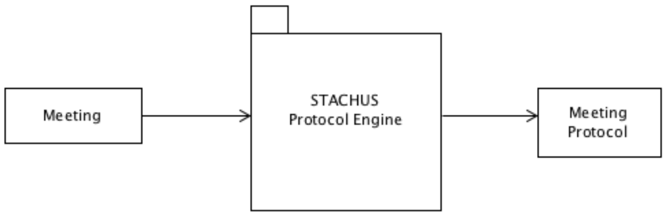


Figure 5-17: Workflow Meeting – Protocol generation

The meeting consists of the audio stream of the meeting conversation, the participating team members and documents that are reviewed, regarded, or consulted during the meeting.

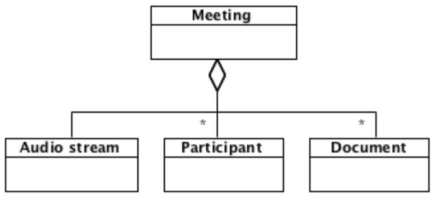


Figure 5-18: The meeting (UML class diagram)

A document can be the agenda for this meeting, old protocols, test and project plans, as well as process documentation, system model objects, or forum and wiki pages.

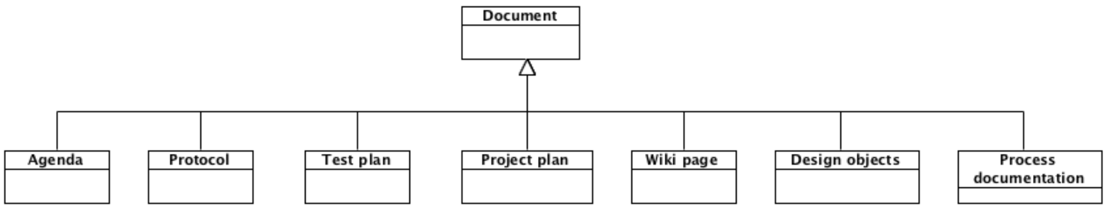


Figure 5-19: Documents (UML class diagram)

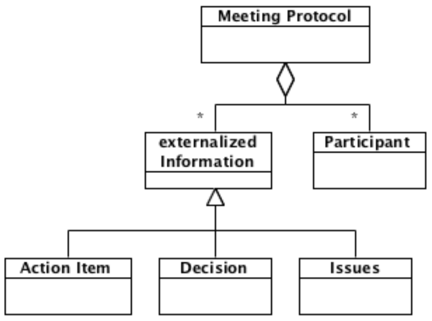


Figure 5-20: Meeting Protocol (UML Class Diagram)

The meeting protocol, generated by the Protocol Engine, consists of externalized information from the meeting and the meeting participants.

The protocol engine can also generate the agenda for the next meeting, based on assigned action items from the current meeting. Moreover, simple list in XML or the inputs for project management and task maintenance tools can be created, as shown in Figure 5-21. Here, the file format (Excel, XML or text) is separated from the meeting artifacts (protocol, agenda, etc.), to address the easy extensibility by further formats (e.g. word-documents or annotated audio files) and further artifacts (for instance Burndown Charts or status reports).

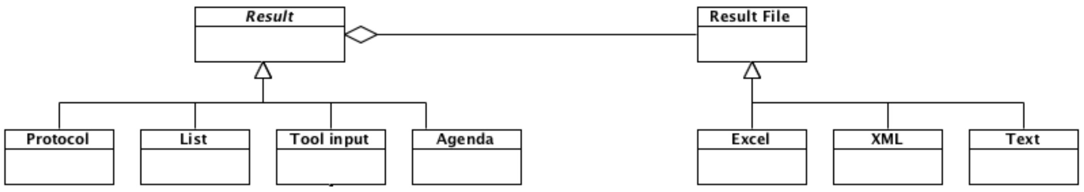


Figure 5-21: Outputs of the protocol engine (UML Class Diagram)

5.4 STACHUS framework: Protocol Engine & Compiler

The STACHUS framework consists of a compiler and a protocol engine, including a recorder, an analyzer and a protocol generator (see Figure 5-22).

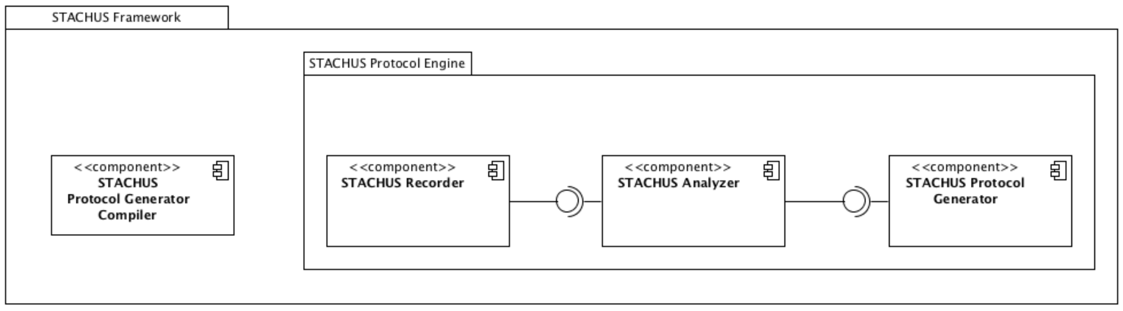

STACHUS Framework
STACHUS Protocol Engine
<<component>> STACHUS Protocol Generator Compiler
<<component>> STACHUS Recorder
<<component>> STACHUS Analyzer
<<component>> STACHUS Protocol Generator

Figure 5-22: STACHUS Framework (UML Component Diagram)

5.4.1 The STACHUS Protocol Engine

The STACHUS Protocol Engine accepts the current meeting and generates a meeting protocol. The STACHUS grammar, presented in chapter 5.2, is integrated in the information externalization process of the Protocol Engine.

The Protocol Engine is composed of three components: the STACHUS Recorder, STACHUS Analyzer, and STACHUS Protocol Generator.

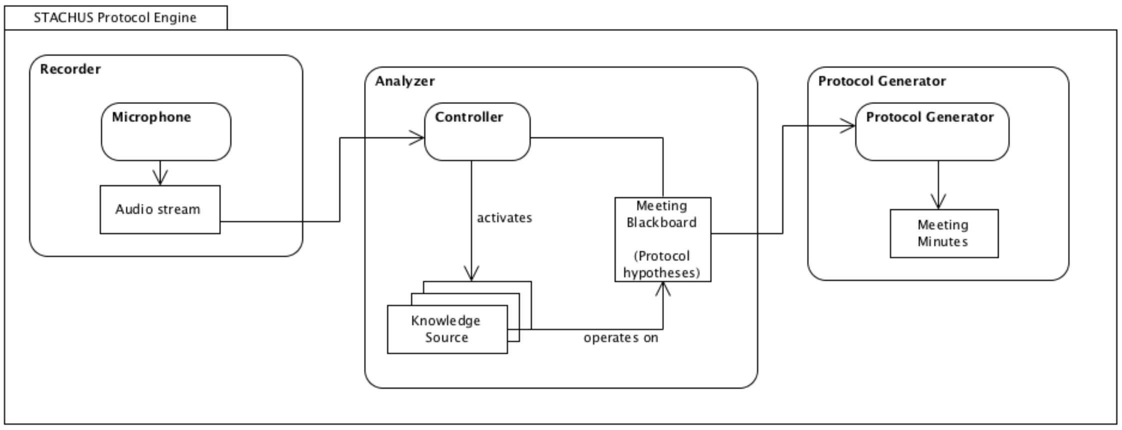


Figure 5-23: The STACHUS Protocol Engine (UML Activity Diagram)

STACHUS Recorder implements the interface to a microphone and is responsible for recording the meeting audio stream; STACHUS Analyzer analyzes the audio stream and searches for meeting information that is to be documented in the protocol; and STACHUS Protocol Generator creates the protocol (and if necessary, other documentation, like the next agenda) from the output of the Analyzer and sends the generated protocol to the meeting participants. As the main part of information externalization and extraction is done by the Analyzer, we will go into the details of the STACHUS Analyzer in the following.

The STACHUS Analyzer is based on the blackboard architectural model, where a common knowledge base, the blackboard, is iteratively updated by a diverse group of specialized knowledge sources (Buschmann et al., 1998). The knowledge sources of the blackboard model are applied in the STACHUS framework to cooperatively solve the speech recognition and information extraction and thus improve the protocol quality. So the problems of the current speech recognition systems, whose word recognition rate is not perfect, are addressed by working together to solve the speech understanding. Thus, the STACHUS Analyzer is built on the ideas of the first major blackboard system, the Hearsay-II system, implemented between 1971 and 1976 (Erman et al., 1980), a speech-understanding system, where various knowledge sources cooperate to solve the speech recognition.

The STACHUS Analyzer consists of three major components: the knowledge sources, a blackboard and a controller. The knowledge sources (KSs) are specialist modules, where each KS provides a specific expertise. In STACHUS the blackboard is a shared repository of partial solutions of to be externalized meeting information, which is readable and writable by all KSs. The control shell of the STACHUS Analyzer controls the flow of problem-solving activities in the system, that is the controller provides a mechanism to organize the use of the KSs in the most effective and coherent fashion.

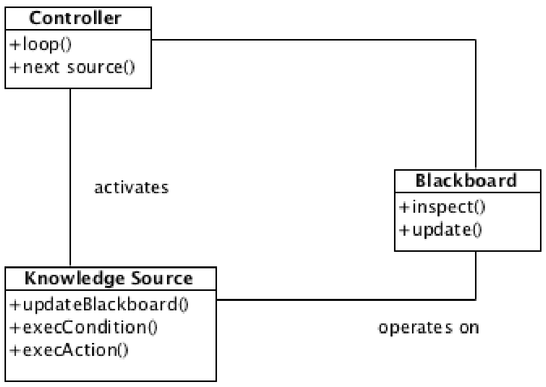


Figure 5-24: Blackboard system (UML Class Diagram)

Each knowledge source adds hypotheses (meeting phrases of to be externalized information) to the blackboard. This process of adding contributions to the blackboard continues until the problem has been solved, i.e. the meeting stream is finished and all meeting information is extracted to be written in the protocol. In this way, the specialists work together to solve the problem.

The ability to support interaction and cooperation among diverse KSs creates flexibility for the framework, as modules can easily be replaced as they become outmoded or obsolete, if better and more effective speech recognition components are developed.

The STACHUS knowledge sources are the speech recognition knowledge sources from external suppliers, as well as the STACHUS Lexer, the STACHUS Parser, and the STACHUS Semantic Analyzer. The knowledge sources for speech recognition transform the meeting audio stream into a textual representation. These can be commercial speech recognition software components (e.g. Naturally Speaking), as well as human transformations (e.g. via Amazons Mechanical Turk (amazon.com, 2005)), that work together in parallel.

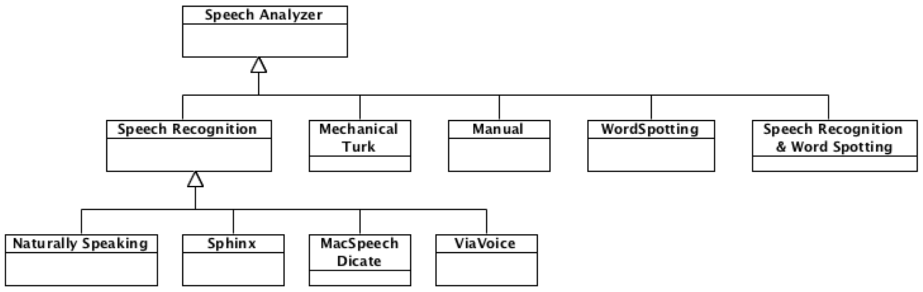


Figure 5-25: Speech recognition components (UML Class Diagram)

The Amazon Mechanical Turk is one of the suites of Amazon Web Services that enables computer programmers (known as *Requesters*) to co-ordinate the use of human intelligence to perform tasks that computers can't solve due to the difficulty of the problem or the complexity of the required algorithms. The Requesters are able to pose tasks known as HITs (*Human Intelligence Tasks*), such as the transformation of a meeting audio into text. *Workers* (also called Providers) can browse among existing tasks and complete them for a monetary payment set by the Requester.

The STACHUS Lexer conducts a lexical analysis of the hypotheses created by the speech recognition components and updates the partial solutions on the blackboard. The Lexer breaks the text into small pieces called *tokens*, where each token is a single atomic unit of the language.

The STACHUS Parser performs a syntax analysis based on the STACHUS grammar that involves parsing the token sequence to identify the syntactic structure of the hypothetical phrases on the blackboard and generates a syntax tree.

The STACHUS Semantic Analyzer searches the syntax tree for predefined key words, the meeting elements, defined by the STACHUS grammar. Only hypothetical phrase containing a meeting element will remain on the blackboard and will get part of the protocol. Thus only 'relevant' information is documented in the protocol.

Figure 5-26 presents the STACHUS Analyzer, the controller, blackboard and the KSs.

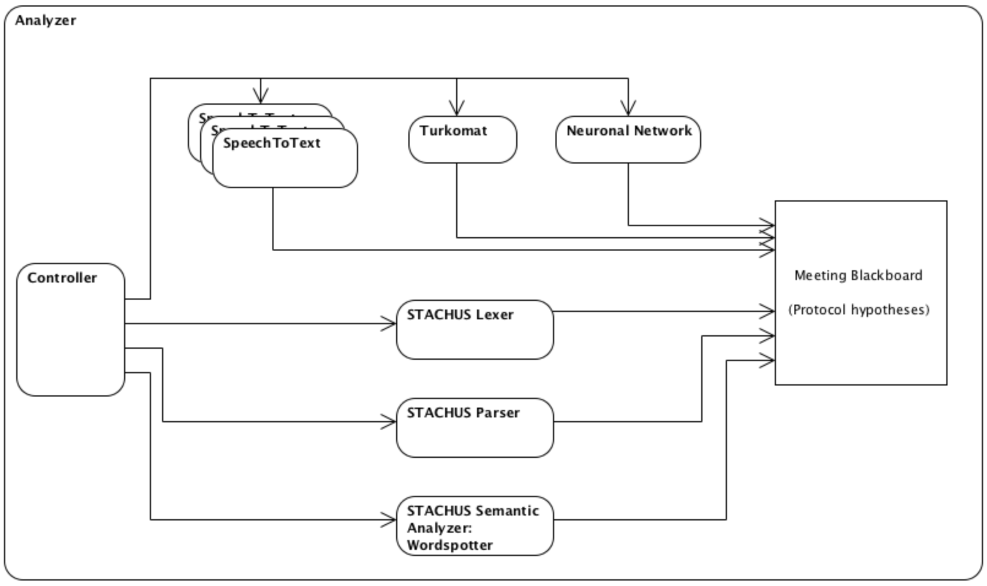


Figure 5-26: Details of the STACHUS Analyzer (STACHUS Protocol Engine) (UML Activity Diagram)

5.4.2 The STACHUS Compiler

The STACHUS Protocol Engine is generated by the STACHUS Protocol Generator Compiler.

The STACHUS Protocol Generator Compiler is a compiler-compiler that generates, based on the project context and the grammar, the STACHUS Lexer, Parser, and Semantic Analyzer. Afterwards the generated modules are integrated with the speech recognition components in the STACHUS Protocol Engine, as shown in Figure 5-27.

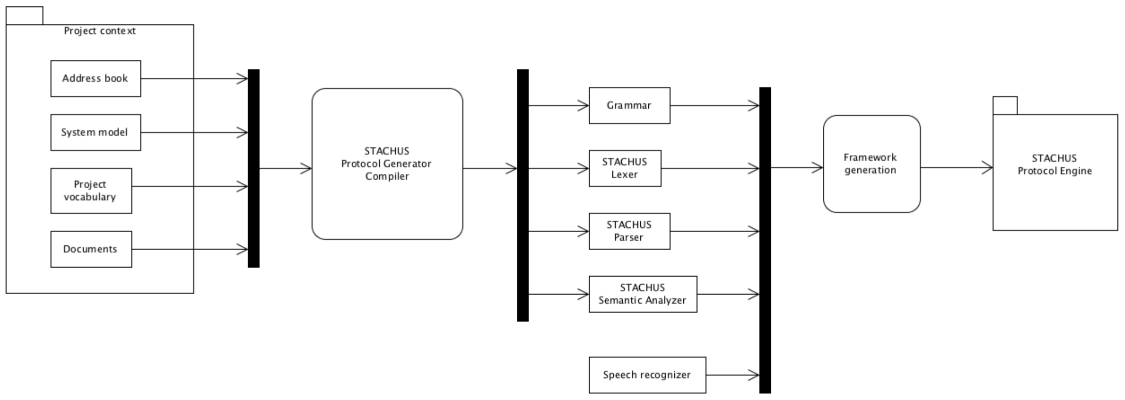


Figure 5-27: STACHUS framework: Protocol Engine generation (UML Activity Diagram)

The modules Grammar, Lexer, Parser, Semantic Analyzer and the speech recognition components are the knowledge sources of the Protocol Engine. The STACHUS Protocol Generator Compiler updates the STACHUS Grammar, Lexer, Parser, and Semantic Analyzer each time, when there are changes in the project context. For example, if a new member joins the project team, the STACHUS Compiler updates the Grammar and rebuilds the STACHUS Protocol Engine, so the name of the new team member can be recognized in the next meeting conversation (Figure 5-28).

The process of integrating the generated modules and generation of the STACHUS Protocol Engine includes updating the STACHSU Protocol Engine Controller, which coordinates the activities of the knowledge sources.

Figure 5-29 shows the procedure, if the system model changes, due to a new understanding based on a recent meeting. During the discussion in the meeting, the system model is updated. Due to the changed system model, the STACHUS Compiler updates Lexer, Parser and the Systematic Analyzer and the protocol engine. That is, the next meeting protocol is generated on basis of the new system model and the included vocabulary.

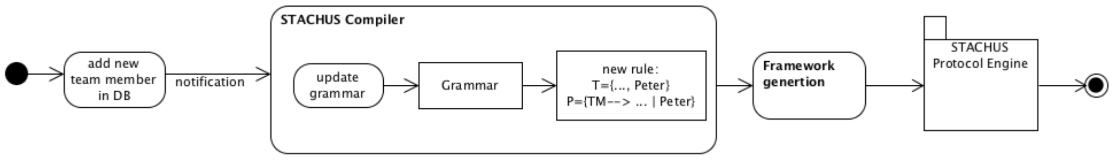


Figure 5-28: Continuous integration of project changes: new team member

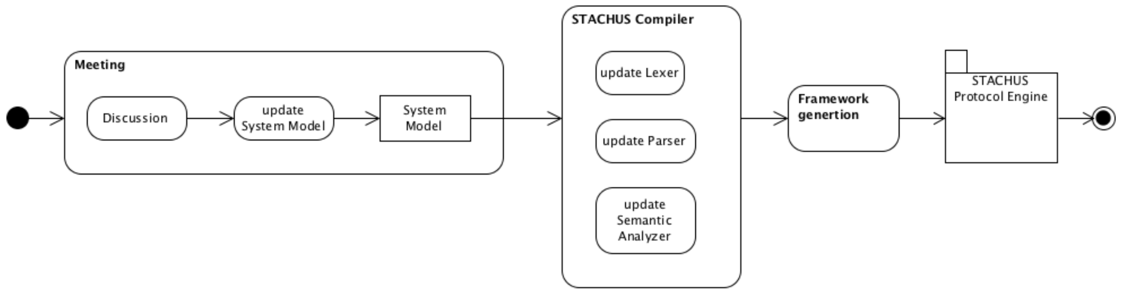


Figure 5-29: Continuous integration: update system model

The protocol itself flows back into the STACHUS Compiler and thus influences the protocol generation of the next meeting, as illustrated in the example in Figure 5-30, where a new action item is defined and assigned during a meeting. This action item is documented in the protocol, which is entered in the STACHUS Compiler that updates the STACHUS Protocol Engine components.

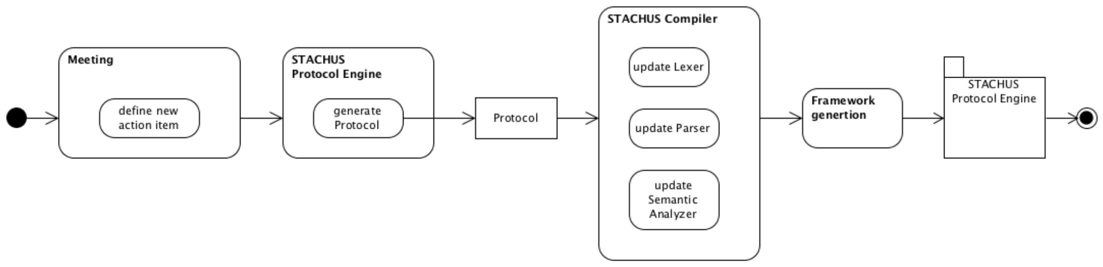


Figure 5-30: Continuous integration: Action item

The protocol engine is regularly updated by the compiler so changes in the project context are easily, fast, and continuously be integrated. That is, the framework is always up-to-date and a self-improving system that continually improves its knowledge sources and thus the quality of the generated protocols, as the compiler-compiler integrates the current project context.

5.5 Software architecture

The following section describes the architecture of the framework.

The overall architecture of our approach is shown in Figure 5-31. The interface or presentation layer contains the STACHUS Audio Recorder and a microphone. The application layer provides the STACHUS Analyzer and STACHUS Protocol Generator, externalizing information from the meeting stream provided by the audio recorder and generating the meeting protocol. The project repository, i.e. the project database containing project documents and the team address book, are covered by the database layer.

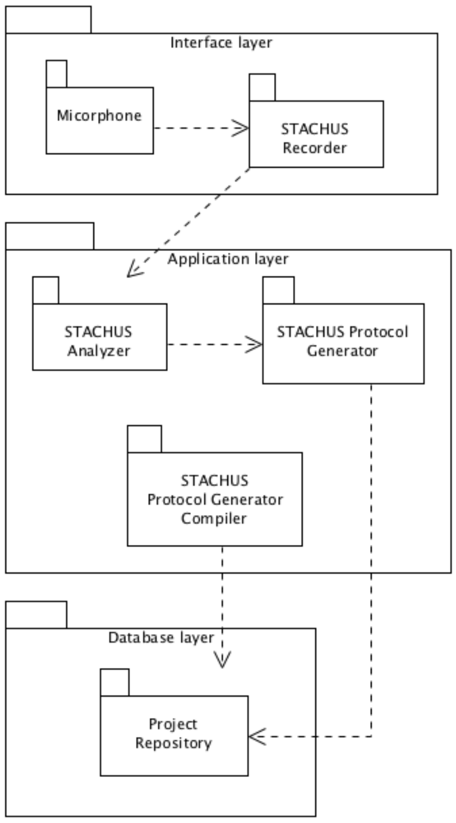


Figure 5-31: Three-tier architecture (UML Package Diagram)

In the following three deployment diagrams of the STACHUS framework and its components are shown.

Figure 5-32 shows a configuration where all subsystems are located on one node. STACHUS Protocol Generator Compiler, STACHUS Protocol Engine, as well as the microphone that records the meeting conversation and the database, including the project repository are on one device, for instance a notebook. The meeting is recorded with the integrated notebook-microphone. The protocol engine, also running on the notebook, generates, based on the recorded meeting audio file, the meeting minutes and stores them locally. The advantage is that the notebook can be carried to each meeting room and no additional network, server connection, or other devices are necessary.

The deployment diagram shown in Figure 5-33 separates the application server, containing the STACHUS subsystems, from the database and microphone, and the devices that access the STACHUS Protocol Generator Compiler and Protocol Engine.

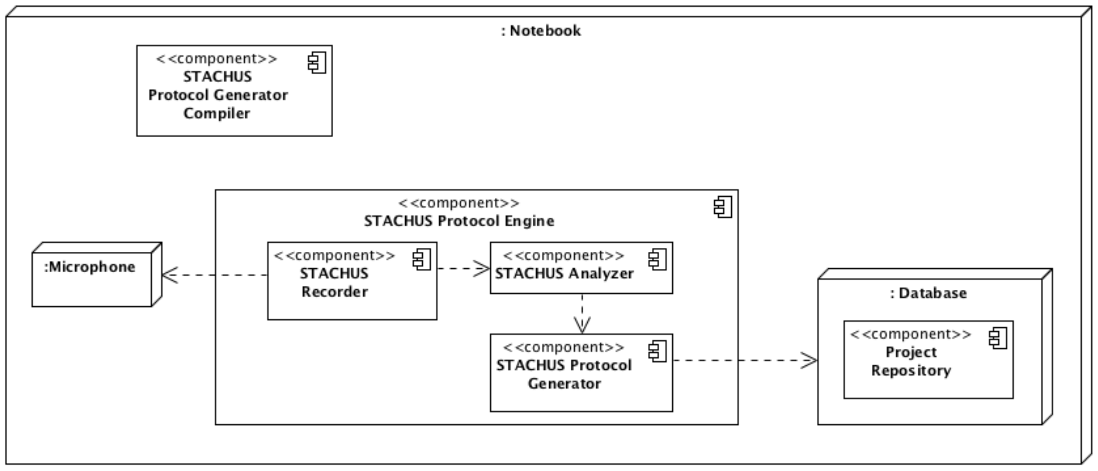

: Notebook
<<component>> STACHUS Protocol Generator Compiler
<<component>> STACHUS Protocol Engine
:Microphone
<<component>> STACHUS Recorder
<<component>> STACHUS Analyzer
<<component>> STACHUS Protocol Generator
: Database
<<component>> Project Repository

Figure 5-32: Architecture of the STACHUS Framework on one node (UML Deployment Diagram)

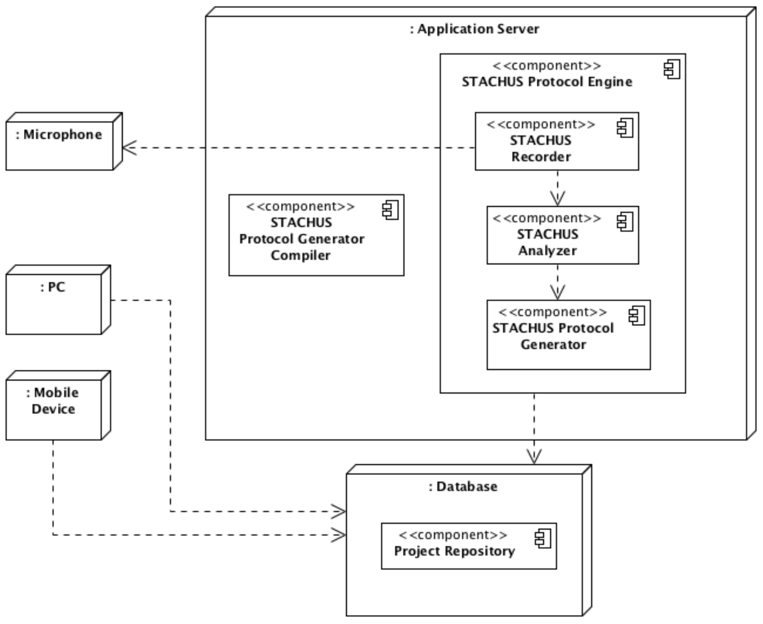

: Application Server
<<component>> STACHUS Protocol Engine
: Microphone
<<component>> STACHUS Recorder
<<component>> STACHUS Protocol Generator Compiler
<<component>> STACHUS Analyzer
: PC
<<component>> STACHUS Protocol Generator
: Mobile Device
: Database
<<component>> Project Repository

Figure 5-33: Architecture with several nodes (UML Deployment Diagram)

Figure 5-34 illustrates the configuration of a thin client and a backend application server. Thus, for instance an ad hoc meeting is supported. The meeting is recorded via the integrated microphone of a smart phone, e.g. an iPhone. Then the audio file is sent to an application server that creates the meeting protocol. The resulting meeting summary is presented at the end of the meeting at the smart phone.

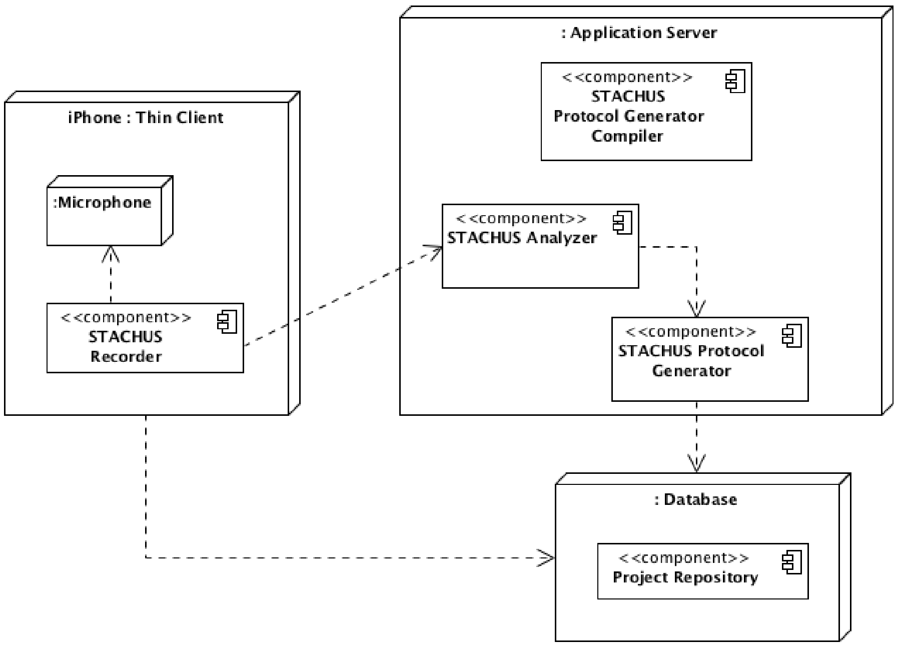


Figure 5-34: Architecture of a thin client solution (UML Deployment Diagram)

6 Empirical evaluation

Evaluation of the hypotheses

The previous chapters discussed the problem of missing externalization of information in meetings, to support an introduction of agile methods in defined process environments. An approach was presented to solve this problem by an automatic protocol generation engine – the STACHUS framework. This chapter presents the empirical validation of the STACHUS framework for automatic protocol generation and externalization of information from meetings. The choice of empirical methods, design of the empirical studies and the derivation of conclusions are covered in this chapter.

In the following we refer to the definitions of Rosnow & Rosenthal (2008) and Bortz & Döring (2005). An overview of empirical research and the methods applied during the evaluation of our hypotheses is also given in Appendix A.III.

Section 6.1 motivates the empirical research and null hypothesis significance testing. In Section 6.2 the hypotheses regarding the automated protocol generation framework are described. The empirical evaluation of the STACHUS framework is based on three empirical studies: The first one is an experimental survey with students, using the STACHUS framework in a software development project that adapts the mobile version of the protocol generator. The experimental survey is described in section 6.3.1. To evaluate the application of automatic protocol generation for traditional meetings and with professionals an explorative industry case study was executed. Section 6.3.2 presents this case study and the resulting conclusions. An additional case study as part of a large student project in cooperation with a real customer is discussed in section 6.3.3.

6.1 Empirical research

Empirical research is a scientific methodology that is applied to gain conclusions about the applicability of the STACHUS framework and the results of the protocol generation engine, by systematical evaluation of experiences via interviews, observations and measuring. On the basis of the collected data, hypotheses are verified and can be accepted or rejected (falsified). Intention is to allow conclusions and theories about the particular object of research. However, conjectures are scientific only if they claim validity, which goes beyond the subjective opinion and workaday experiences of indivicuals. Empirical research is applied in the context of this thesis to evaluate our hypotheses regarding the practicability, usability, and effort reduction of the STACHUS framwork for meeting minutes generation.

A *Hypothesis* is a proposed explanation for observable phenomena, i.e. it represents a "research idea that serves as a premise or supposition" (Rosnow & Rosenthal, 2008, p.417). A hypothesis is called a scientific hypothesis, if it can be tested, is plausible, and concise. Moreover, it has to go beyond individual cases (generalizability) and be falsifiable based on observational data. "Scientific hypotheses state a more or less precise relationship between two or more variables that hold for a defined population of comparable objects." (Bortz & Döring, 2005, p.12) A distinction is drawn between a null hypothesis (the hypothesis to be nullified that states that there is no relation between two variables) and alternative hypothesis (the experimental hypothesis).

Null hypothesis significance testing (NHST) uses statistics and probabilities to evaluate null hypotheses and determine the significance level α and the probability (p value). NHST evaluates if for example a difference between two means might be due to chance. In this context a *Type I error* implies that the decision maker mistakenly rejected the null hypothesis (H_0) when it is, in fact true and should not have been rejected.

$$P(accept\ H_1|H_0\ true) \leq \alpha \text{ i.e. } P(Type\ I\ error) \leq \alpha$$

A significance level of 5%, the .05 alpha, is seen by many scientists as a good "fail-safe" standard to decide about rejecting or accepting H_0, because it is convenient (most statistical tables show 5% values) and stringent enough to protect from too often concluding that the null hypothesis is false when it is actually true.

The collected evaluation data is analyzed and validated by *statistical tests*. Beside describing data and measuring relationships, researchers are usually interested in making comparisons using statistical test. In the following, we will analyze the data based on t-tests for small samples and normal distribution. A t-test is a test of statistical significance that examines the difference between two means against the background of the within-group variability (i.e. the variability of the scores within the sample). The larger the difference between the means, and/ or the smaller the within-group variability for any given size of study, the greater will be the value of t. Because large t values are associated with differences between means that are more statistically significant, researchers generally prefer larger t values. That is, larger t values have a lower level of probability (the p value) and, in turn, allow researchers to reject the null hypothesis that there is no difference between means. When the results are not statistical significant, the evidence is termed to be anecdotal evidence.

The unpaired or independent-sample t-test examines the differences between two independent means that is the two groups are independent of one another, i.e. the results in one group are not influenced by the results in the other group. Considering the t-test, the degrees of freedom (symbolized as *df*) are defined as $n_1 + n_2 - 2$ for … and n-1 for …

Paired or dependent samples t-tests typically consist of a sample of matched pairs of similar units, or one group of units that has been tested twice (a "repeated measures" t-test).

The p values are probability values, obtained in the t-tests. P values can be two-tailed or one-tailed. The two-tailed p values are associated with a result supporting a prediction of a nonspecific direction of a research result, e.g. $\mu_0 \neq \mu_1$, whereas one-tailed p values are associated with a result supporting a prediction of a specific direction of a research result, e.g. $\mu_0 > \mu_1$.

The t value is calculated as follows:

Independent one-sample t-test (tests the null hypothesis that the population mean is equal to a specified value μ_0):

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$\bar{x}$ is the sample mean of the data, *n* is the sample size.

$$t = \frac{\bar{x} - \mu_0}{s}\sqrt{n}$$

where *s* is the standard deviation of the sample. The degrees of freedom used in this test is $n - 1$.

$$s^2 = \frac{\sum(x_i - \bar{x})^2}{n - 1}$$

$$s = \sqrt{s^2}$$

Independent two-sample t-test (this test is only used when the two sample sizes (that is, the number n of participants of each group) are equal and it can be assumed that the two distributions have the same variance.

The t statistic to test whether the means are different can be calculated as follows:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{s_{x_1 x_2}}\sqrt{\frac{n}{2}}$$

where

$$s_{x_1 x_2} = \sqrt{\frac{s^2{}_{x_1} + s^2{}_{x_2}}{2}}$$

is the standard deviation for 1 = group one and 2 = group two. The denominator of t is the standard error of the difference between two means. The degrees of freedom used in this test is $n + m - 2$.

Dependent t-test for paired samples (this test is used when the samples are dependent)

$$d_i = x_i - y_i$$

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i$$

$\bar{d}$ is the mean of the sample differences and s_d the standard deviation of the sample. The degrees of freedom is $n - 1$.

$$s_d = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(d_i - \bar{d})^2}$$

$$t = \frac{\bar{x}_d - \mu_0}{s_d}\sqrt{n}$$

Once a t value is determined, a p-value can be found using a table of values from Student's t-distribution. If the calculated p-value is below the threshold chosen for statistical significance, the null hypothesis is rejected in favor of the alternative hypothesis.

The confidence interval is the upper and lower bounds of a statistic, where confidence is defined as $1 - \alpha$

$$\left[\bar{x} - t\left(1 - \frac{\alpha}{2}, n - 1\right)\frac{s}{\sqrt{n}}; \bar{x} + t(1 - \frac{\alpha}{2}, n - 1)\frac{s}{\sqrt{n}}\right]$$

If the sample is not normally distributed, non-parametric tests are used – e.g. a Wilcoxon rank-sum test for independent samples or Wilcoxon signed-rank test for the case of two related samples or repeated measurements on a single sample.

During a survey the data is gathered for the statistical test that rejects or accepts a hypothesis. Internal and external *validity* are essential quality criteria of a survey. Validity is the degree to which what was observed or measured is the same as what was purported to be observed or measured. Internal validity is the degree of validity of statements made about whether X causes Y. External validity is the dependability of causal generalization across persons, settings, treatment, and outcome variations. In particular external validity – the degree of generalizability – has to be assured during research. *Reliability* is a second quality criterion, which is the extent to which observations or measures are consistent or stable.

6.2 Hypotheses and research questions

The focus of our empirical evaluation was to analyze the practicability and effectiveness of the framework STACHUS and its integration in the project and meeting workflow to value the automated protocol generation process and the generated protocols. Based on the functional and non-functional requirements, defined in chapter 5.1, twelve hypotheses for automated protocol generation framework are derived, to evaluate STACHUS and its components by assessing the hypotheses. Each hypothesis is related to functional or non-functional requirements (FR/ NFR), that are validated with this hypothesis. The valuation of the hypotheses is distributed to three case studies (C1 – C3) – shown in Table 6-1 − two in an academic environment (C1 & C2) and one larger, industrial case study (C2).

Hypothesis		**FR/ NFR**	**Case study coverage**
1	The STACHUS Protocol Engine reduces the effort of protocol generation in comparison to manually written protocols	FR3, NFR3	C1, C2
2	STACHUS Protocol Engine (audio recording & protocol generation) ensures easiness in communication, i.e. the communication is not complicated or constrained by audio recording and the process of automated protocol generation	NFR1, NFR4	C1 , C2
3	The STACHUS meeting grammar is easy to learn and apply and does not constrain the meeting progress	NFR1, NFR4	C1, C2
4	STACHUS Audio Recorder and Protocol Generator are easy to use	FR1, NFR1, NFR4	C1, C2
5	The STACHUS Compiler & Protocol Engine support knowledge management by information externalization	FR2, FR3, FR4	C1, C2
6	The STACHUS Protocol Engine reduces the information overload	FR3	C1
7	The STACHUS Protocol Engine supports externalization of volatile communication and thus improves the forgettability of action items, decisions etc., especially during informal meetings, in comparison to the manual protocol generation from memory	FR2, FR3	(C1)
8	The STACHUS Protocol Generator accelerates the protocol generation, i.e. the protocols are published faster than manually written protocols	NFR3	C2
9	The STACHUS Protocol Engine support agile practices	NFR1, NFR5	C2
10	The STACHUS Protocol Generator reduces gaps between different tools in the project & meeting workflow and thus simplifies the meeting process	FR4	C2

11	Protocol generation procedure (STACHUS Protocol Engine and STACHUS Compiler) can be improved by a dynamic extending grammar	NFR6	C2
12	The STACHUS Compiler and STACHUS Protocol Engine improve the correctness of protocols compared to the correctness of manually written meeting minutes	NFR2	C3

Table 6-1: Relationship between the hypotheses of the dissertation, requirements of STACHUS, and coverage by case studies

Effort reduction for protocol generation (H1)
An effective application of the automatic protocol generation engine – especially in an agile environment – requires the reduction of effort for generating a protocol. The manual creation of protocols is time consuming, why most agile projects do without protocols as the effort is larger than the value for them. If it is the aim to increase the number of protocols in agile projects and relieve the protocol generation also in traditional environments, the effort for creating a protocol has to be minimized. Thus, the reduction of effort is important for the effectiveness of the meeting protocol generation framework.

The STACHUS Protocol Engine reduces the effort of protocol generation in comparison to manually written protocols. Effort reduction is measured using the questions Q8 (effort for writing a protocol) and Q9 (effort for post-processing a meeting). Additionally the effort reduction is evaluated using the questions Q17 and Q18, regarding the potential for fewer errors.

Easiness in communication (H2)
The easiness in communication is depending on the easiness of the grammar (H3) and the usability of the framework during meetings (H4), i.e. the STACHUS Audio Recorder and STACHUS Protocol Generator. That is, the easiness in communication is supported, if the grammar is easy to learn, does not need a comprehensive training and it is easy for the meeting participants to stick to the grammar rules. Moreover, the execution of the grammar must not disturb the meeting progress. In addition, the STACHUS Audio Recorder has to be easy to use, that is it should support an intuitive user prompting, so no training is required to activate the meeting recorder. Moreover, setting up the audio recorder and protocol generator has to happen in minimal time and with no additional effort.

Thus, an automated protocol generation approach has to support an easy communication, to be better to use and therefore improve the effectiveness of the meetings. Aim is that the automation of protocol generation does not complicate and constrain the communication during a meeting (especially regarding the application of a grammar).

STACHUS Protocol Engine (STACHUS Audio Recorder & Protocol Generator) ensures easiness in communication, i.e. the communication is not complicated or constrained by audio recording and the process of automated protocol generation.

Furthermore, possible (negative and positive) changes of the communication behavior are analyzed to evaluate the effects of applying a meeting grammar and the fact of recording a meeting. The changes in communication behavior are measured using questions Q23, as well as using the respondents' comments belonging to this question and observations during the whole student project.

Easiness of the meeting grammar (H3)
As mentioned above, easiness of communication is depending on the acceptance of the applied meeting grammar. The automated protocol generation is effective, if the meeting participants can easily learn the meeting grammar (needed for the protocol generation) and don't need comprehensive trainings. Moreover, it has to be easy to stick to the rules of the meeting grammar during the meeting and don't confuse the meeting participants. Thus, the meeting grammar must not constrain the meeting process.

The STACHUS meeting grammar is easy to learn and apply and does not constrain the meeting progress. The easiness of the grammar is measured using the question Q27.

Easiness of the framework (H4)
The easiness of the STACHUS framework for the meeting participants is depending on the usability of the STACHUS meeting recorder and the STACHUS Protocol Generator. The STACHUS Audio Recorder has to support an intuitive user prompting, so no trainings for the activation of the meeting recorder are required and starting a meeting recording at the begin of the meeting can be done by each meeting participant in no time. Additionally, the protocols are automatically and without any effort generated by the STACHUS Protocol Generator and handed out to the meeting participants; and ideally no additional effort is involved for reviewing and correcting incorrectly or incompletely automatically created protocols.

The easiness of the STACHUS framework for the meeting participants increases not only the easiness in communication (as the conversation is not disturbed by the audio recording process) but also the acceptance of the tool is increased and handling errors are reduced. The STACHUS Audio Recorder and Protocol Generator are easy to use, in terms of the definition above. The easiness of STACHUS is measured using Q20, observations during several meetings, and using the open questions Q28 – Q31.

Knowledge management support (H5)
Protocols are a source for knowledge management. Knowledge is created or combined in meetings by socialization (tacit to tacit), during the conversations, and externalization (tacit to explicit), by creating a protocol. This 'people-to-document' approach (codification approach) makes implicit knowledge explicit. Moreover, the knowledge is made independent of the person from whom it is extracted and stored in protocols, reports, or databases. These protocols are read by other project members and thus make a contribution to knowledge spreading, combination of knowledge (explicit to explicit), and internalization (explicit to tacit).

Possibly, the knowledge combination or generation of new knowledge during this process of externalization and internalization of knowledge has an effect on the comprehension of the project and system model. That is, protocol generation conduces to knowledge transfer and thus to knowledge generation. Protocols keep decisions, the understanding and knowledge of the team in written form to reuse and apply the knowledge in the future or in other projects.

Hypothesis H7 states that the STACHUS Compiler & Protocol Engine support knowledge management by information externalization and protocol generation. This support of knowledge management is evaluated by a systematic observation during a judgment study that accompanied the whole student project and it is measured using the question Q21.

Reduction of information overload (H6)
Information overload, popularized by Alvin Toffler (1970), refers to the presence of too much information to find the 'right' information, understand an issue or make a decision.

The automation of the protocol generation process will increase the number of available protocols and reports. The access to so much information may increase the information overload of individual team members, as searching for the 'right' or a specific information becomes difficult in such an amount of information. The sole creation of paper protocols does not support the meeting participant in retrieving requested information of the meeting. So additionally the protocol generator has to reduce the information flood caused by meeting minutes, by preparing the discussed information to allow an easy searching and retrieval of specific information.

The STACHUS Protocol Engine (of the smart phone version) reduces the information overload. The reduction of information overload is measured using the question Q22.

Forgettability (H7)
Informal meetings are characterized by volatile communication. That is, if the discussed information is not externalized and recorded in a protocol, it will be forgotten and get lost. As a result, assigned action items for instance may be overlooked (as no written documentation is available to look up important parts of the conversation after the meeting) and thus are not executed by the next meeting, where they have to be discussed again. An effective application of the automation of protocol generation has to reduce the 'forgettability' of discussed, but volatile information.

The STACHUS Protocol Engine supports externalization of volatile communication and thus improves the 'forgettability' of action items, decisions, etc., especially during informal meetings, in comparison to the manual protocol generation from memory. The volatility of information of meetings is evaluated using the question Q16. Additionally an experiment was executed to measure the level of forgettability by opposing the number of discussed information in a meeting to the number of remembered information after the meeting.

Accelerated protocol generation and publication (H8)
One observation of traditional meeting minutes, as mentioned in chapter 3, is that they are published often too late, i.e. just before the next meeting, so an extensive preparation for the next meeting is impossible and assigned action items are not executed. An improved meeting management needs to shorten the time of the protocol generation process, to allow the meeting participants to prepare for the next meeting and contribute to the project progress.

The STACHUS Protocol Generator accelerates the protocol generation, i.e. the protocols are published faster than manually written protocols. The accelerated protocol generation is measured using the question Q8 (protocol writing effort), Q10 regarding the average protocol publication time and Q11.

Support of agile practices (H9)
The effective integration of the STACHUS Audio Recorder and Protocol Generator in the meeting process, especially for agile teams, requires the support and compliance with the agile practices, summarized in chapter 2. That is, the lightweight and flexible character of agile meetings and projects should be maintained, the protocol generation may not result in additional effort for the meeting participants, and the STACHUS framework has to support informal meetings. In addition, the agile projects shall be enriched by the externalization and storage of knowledge.

STACHUS Protocol Engine support agile practices. The degree of agile support is measured using the question Q21.

Tool support und simplified meeting process (H10)
Updating to-do-lists and maintenance of project management tools can be time consuming. Surveys, e.g.(VersionOne, 2010), yielded that task and project management is mostly done electronically. Thus, the effective integration and application of the STACHUS framework in the project lifecycle can be ensured, if the applied project management tools are connected to STACHUS and information, discussed in a meeting, can be directly transferred to these tools. That is, if a new task was assigned during a meeting, it is recorded in the protocol and directly added to the individual task-management-list. Furthermore, the protocols created by the STACHUS protocol generator have to be adaptable to the individual project format and layout standards.

The STACHUS Protocol Generator reduces gaps between the tools in the project & meeting workflow and thus simplifies the meeting process. The tool integration and workflow support is measured using the question Q19.

Protocol improvement by dynamic grammar (H11)
Today, the speech recognition rate is still not 100% accurate. To guarantee high quality protocols that are automatically generated, the speech recognition has to improve. This can be done

by the integration of project vocabulary (names of the team members, tools, etc.), system models and project domain context (technical terms) in a dynamic grammar to improve the protocols.

The protocol generation procedure (STACHUS Protocol Engine and STACHUS Compiler) can be improved by a dynamic extending grammar, resulting in a better protocol quality. The improvement of the protocol generation is measured based on several experiments, where the speech recognition rate and protocol quality (correctness, completeness) is analyzed before and after adding additional project vocabulary from the context of the project and those from the system model for instance, to the speech engines of STACHUS.

Less error (H12)
As shown in chapter 3, protocols are often published to late, or the protocols are incomplete or incorrect.

The STACHUS Compiler and STACHUS Protocol Engine improve the correctness of protocols compared to the correctness of manually written meeting minutes. The error reduction is measured using question Q12 regarding the amount of incorrect protocols and question Q18 about the error reduction potential.

The following sections describe our case studies and the empirical experiments used to evaluate the hypotheses stated in this section. Each section consists of a brief introduction, where the case study is presented, followed by a description of the empirical methods, applied for evaluation. Afterwards, the results are derived, which are finally discussed and evaluated.

6.3 Empirical evaluation of the hypotheses

6.3.1 Case study I

The first case study was executed in the context of a student project at the Technische Universität München during the summer semester 2009. The objective was to evaluate the applicability of the automated meeting protocol generation framework, ensure its feasibility, and improve the framework during the project if necessary. Moreover the practicability of the meeting grammar and its application during meetings should be tested, to analyze the effects on the meeting conversation.

At the end of the student project, the hypotheses H1 – H7 were analyzed. The other hypotheses are evaluated in the case studies II and III.

The case study was designed as a one-group pre-post case study, i.e. it was designed with an experimental group using the STACHUS framework, compared to the meeting and protocol generation experiences of this group with the previous manual protocol generation.

The case study took place during the summer semester 2009 within the "iPhone Praktikum 09", a practical student's course for programming mobile multimedia applications with the iPhone SDK.

Objective of the student's project was to develop a mobile version of the STACHUS framework that runs on an iPod/iPhone, to support informal and ad-hoc meetings. Prerequisite for the team was to apply the STACHUS meeting management framework for developing the mobile version.

To answer the research questions, the participants of the survey needed to have experiences with meetings and taking meeting minutes, and attended ideally several kinds of meetings in the software development environment. Therefore, the inclusion criteria of the protocol-generation-team were bachelor/ master degree in software engineering, experience in real software development projects, and experience and interest in meeting management. Additionally, importance was attached to grouping a heterogeneous student team with different levels of meeting experience and interests.

The population of the practical course was 31 students, five (one female and four male students) passed the inclusion criteria and became a member of the protocol-generation-team.

The case study consisted of two parts: on the one hand a six month project accompanying observation (n = five students) and on the other hand a final questionnaire. A sample of four male and female students and four further project participant (project partners and customers, and potential users) participated in the concluding questionnaire, which additionally required first experiences with the STACHUS framework to answer the questions. 25 % of the respondents were female, 75% male, at the age between 24 and 55.

In this case study we use observations, interviews and a questionnaire. The questionnaire is attached in Appendix III.

In the following, the results of the questionnaire and the statistical tests are shown, structured according to the hypotheses mentioned in section 6.2.

H1: The STACHUS Protocol Engine reduces the effort of protocol generation in comparison to manually written protocols
This hypothesis is tested at a significance level of 5%, comparing the means of effort for manually writing a protocol and for generating it via STACHUS.

H_0: $\mu_{manual} \leq \mu_{STACHUS}$
H_1: $\mu_{manual} > \mu_{STACHUS}$

The interpretation of the respondents' data shows that the mean effort for manually writing a protocol is between 7 and 113 minutes, depending on the meeting duration. In contrast, the effort for the automated generation of meeting protocols is approximately five minutes, for starting and stopping the meeting recording and starting the application– the generation process is fully automated. The detailed numbers for writing a protocol are shown in Diagram 6-1.

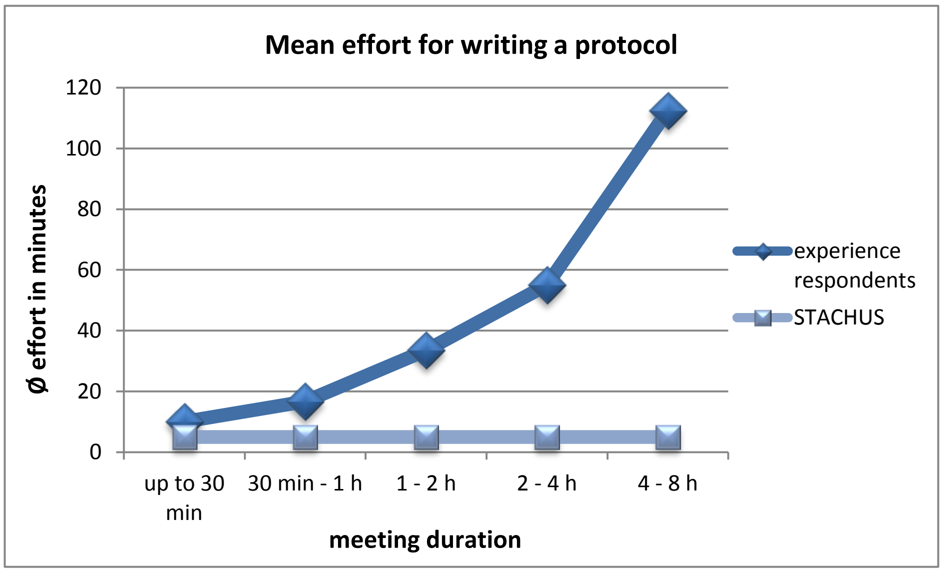


Diagram 6-1: Questionnaire responds: mean effort for writing a protocol

Moreover, the effort for the meeting post processing, e.g. for updating the individual to-do list, which should be add the manual effort, was between 13 and 105 minutes, depending on the meeting duration.

The results of the statistical test are given in Table 6-3, which shows that the mean effort for writing a protocol manually is higher than the mean effort for starting the STACHUS Protocol Engine for an automated protocol generation.

The calculations are exemplarily shown for meeting with duration of 60 - 120 minutes:

$$\bar{x} = \frac{1}{n}\sum x_i = \frac{1}{7}\,(45 + 30 + 60 + 30 + 10 + 30 + 30) = 33{,}6 \approx 34$$

	x_i	x_i-µ	$(x_i$-µ$)^2$
	45 min	11,4	130,6
	-		0,0
	30 min	-3,6	12,8
	60 min	26,4	698,5
	30 min	-3,6	12,8
	10 min	-23,6	555,6
	30 min	-3,6	12,8
	30 min	-3,6	12,8
sum		0,0	1435,7
s²			205,1
s			14,3

Table 6-2: Calculation of s² and s for 60 - 120 min meetings

$$s^2 = \frac{\sum(x_i - \bar{x})^2}{n-1} = 205{,}1$$

$$s = \sqrt{s^2} = \sqrt{205{,}1} = 14{,}3$$

$$t = \sqrt{n}\frac{\bar{x} - \mu_0}{s} = \sqrt{7} * \frac{34 - 5}{14{,}3} = 5{,}4$$

$$\left[\bar{x} - t\left(1 - \frac{\alpha}{2}, n-1\right)\frac{s}{\sqrt{n}}; \bar{x} + t(1 - \frac{\alpha}{2}, n-1)\frac{s}{\sqrt{n}}\right]$$
$$= \left[34 - t\left(1 - \frac{0{,}05}{2}, 6\right)\frac{14{,}3}{\sqrt{7}};\ 34 + t(1 - \frac{0{,}05}{2}, 6)\frac{14{,}3}{\sqrt{7}}\right]$$
$$= \left[34 - 2{,}447 * \frac{14{,}3}{\sqrt{7}};\ 34 + 2{,}447\frac{14{,}3}{\sqrt{7}}\right] = [20{,}8; 47{,}2]$$

$$p\ value_{\ t=5{,}4\ and\ df=6} < 0{,}001\ (one - tailed)$$

Measure	Manual	Manual	Manual	Manual	Manual	STACHUS
Meeting duration	< 30 min	30 - 60 min	1 - 2 h	2 - 4 h	4 - 8 h	meeting independent
$\bar{x}$ (in min)	10	17	34	55	113	5
s	0	9,4	14,3	18,7	13,0	0
s^2	0	88,9	205,1	350	168,8	0
n	7	6	7	6	4	12
t	-	3,1	5,4	6,5	16,6	-
p	-	<0,025	<0,001	<0,001	<0,0005	-
95% confidence interval	-	7,1 to 26,9	20,8 to 47,2	35,4 to 74,6	92,3 to 133,7	-

Table 6-3: Results: effort for protocol generation manual vs. STACHUS generated

The savings of time, when using automated instead of manual protocols, are on average between 5 and 108 minutes per meeting protocol. The results of the independent t-test on these data, depending on the meeting duration, yielded $t \geq 3{,}1$, $4 \leq df \leq 7$, and $p < 0{,}025$ one-tailed.

As the calculated p-value is below the threshold of $p = 0{,}05$ for statistical significance, the null hypothesis can be rejected in favor to the alternative hypothesis.

The acceptance of the alternative hypothesis can be confirmed by the results of the question Q17 and Q18, where 100% of the respondents believe that using the STACHUS framework will save time and 87,5% of the respondents are convinced that the automatically generated protocols have less errors than the manual written protocols. This can be seen as an additional time saving factor, as the effort for reviewing protocols and fixing them is decreased compared to the manual creation.

To verify the hypothetical error reduction rate in automatically generated protocols by the STACHUS framework a small experiment was done comparing manual and automated protocols. This experiment showed that although the error rate of both protocol versions is almost the same (the minutes taker does errors due to distractions and the fast communication, whereas the protocol generator is confronted with a speech recognition with less than 100% word recognition rate), the number of recorded items is slightly higher in the automated protocols. One explanation could be that the minutes taker rejects – from his view – irrelevant items, which is not done in the automated version. However, if an initially irrelevant item becomes relevant in the future it is not part of the manually written minutes in contrast to the automated protocols. Moreover, an improved speech recognition rate in the future will additionally reduce the error rate of the automated protocols.

H2: STACHUS Protocol Engine (audio recording & protocol generation) ensures easiness in communication, i.e. the communication is not complicated or constrained by audio recording and the process of automated protocol generation

Beside the easy application of the framework (H3) and easy grammar (H4), the effects on the communication behavior contribute to the easiness in communication. The possibly changed communication behavior is evaluated in question Q23 and should be minor or predominantly positive to support the alternative hypothesis.

75% of the respondents answered the question, if they believe in a changed communication behavior, when using STACHUS, in the affirmative. However mostly a positive effects is seen, e.g. "one would try to speak more articulately and systematic" or "think before speaking".

These results are confirmed by observations during different meetings. For an observer it is not obviously to see, if the team is currently applying the STACHUS framework for protocol generation and the meeting is recorded or not, as the meeting workflow is not influenced by the automated protocol generation process.

For this reason, the team members were asked during an interview, to evaluate the communication behavior of a traditional meeting and of a STACHUS-meeting. If the hypothesis is right, the evaluation should increase, that is

$$H_0: \mu_{trad} \geq \mu_{STACHUS}$$
$$H_1: \mu_{trad} < \mu_{STACHUS}$$

The results are summarized in Table 6-4 and Table 6-5.

Measure	Traditional meeting mgmt	STACHUS meetings
$\bar{x}$	1,67	2,5
s	0,8	0,5
s^2	0,7	0,3

Table 6-4: Mean and Variability for the communication behavior evaluation

	Traditional meetings	coded	STACHUS meetings	coded	d	$d_i-\bar{d}$	$(d_i-\bar{d})^2$
1	to be improved	2	ok	3	1	0,17	0,03
2	bad	1	to be improved	2	1	0,17	0,03
3	bad	1	ok	3	2	1,17	1,36
4	ok	3	ok	3	0	-0,83	0,69
5	bad	1	to be improved	2	1	0,17	0,03
6	to be improved	2	to be improved	2	0	-0,83	0,69
$\bar{x} \mid \bar{d}$		1,67		2,5	0,83		
sum						0,00	2,83
s_d^2							0,57
s_d							0,75

Table 6-5: Mean and Variability for the communication behavior evaluation

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i = 0{,}83$$

$$s_d = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(d_i - \bar{d})^2} = \sqrt{\frac{2{,}83}{5}} = 0{,}75$$

$$t = \frac{\bar{d} - \mu_0}{s_d}\sqrt{n} = \frac{0{,}83 - 0}{0{,}75}\sqrt{6} = 2{,}7$$

The dependent t-test for paired samples on the data of the interviews yielded $\bar{d} = 0{,}83$, $s_d = 0{,}75$, $t = 2{,}7$, df = 5, and $p < 0{,}025$, so we can reason that STACHUS has positive effects on the communication behavior of meetings.

H3: The STACHUS meeting grammar is easy to learn and apply and does not constrain the meeting progress

Hypothesis H3 is accepted, if the majority of the respondents verify the easiness of the grammar. As the response choices of the corresponding research question Q27 are coded with natural numbers from “very difficult” (1) in ascending order to “very easy” (4), the arithmetic mean should be larger than 2,5 to reject the null hypothesis (again at a significance level of 5 %).

H_0: $\mu \leq 2{,}5$

H_1: $\mu > 2{,}5$

Measure	Responds
$\bar{x}$	2,875
s	0,64
s^2	0,41
n	8

Table 6-6: Mean, Variability and Number of Participants for Q27

75 % of the respondents evaluated the grammar, the learning process, and to follow the grammar during the meeting as "easy" or "very easy". However, the independent t-test yielded t = ed t = 1,66, df = 7, a 95% confidence interval ranging from 2,34 to 3,41, and $p < 0,10$ one-tailed, which results in the acceptance of the null hypotheses and rejection of the alternative hypothesis, due to the 5 % significance level. Though, at a significance level of $\alpha = 0,1$the null hypothesis is rejected.

H4: STACHUS Audio Recorder and Protocol Generator are easy to use

The hypothesis H4 is accepted, if the majority of the respondents confirm an easy usability of the audio recorder and protocol generator, and an intuitive user prompting to easily start the audio recorder and create a protocol. As the response choices of the corresponding research question Q20 are coded using natural numbers, that is, "yes" is coded with 3, "fair" with 2, and "no" is coded with 1, the mean 'usability' should be larger than 2 to accept the alternative hypothesis.

H_0: $\mu \leq 2$

H_1: $\mu > 2$

Measure	Responds
$\bar{x}$	2,71
s	0,70
s^2	0,49
n	8

Table 6-7: Mean, Variability and Number of Participants for Q20

The independent one-sample t-test on these data yielded t = 2,89, df = 7 , $p < 0,025$ one-tailed, and a 95% confidence interval ranging from 2,13 to 3,30.

Due to the results of H2 – H4, the hypotheses are accepted that the easiness of the communication is ensured and the communication is not disturbed, when using the STACHUS protocol generator.

H5: The STACHUS Compiler & Protocol Engine support knowledge management
This hypothesis will be accepted, if the majority of the respondents verify a positive effect on knowledge management by STACHUS, i.e. that the STACHUS Protocol Engine generates knowledge and makes a contribution to the transfer of knowledge, by externalizing implicit knowledge and preparing in protocols.

The response choices of the corresponding research question Q21 are coded with natural numbers (no (1), partly (2), yes (3)), so the average (μ) should be larger than 2 to reject the null hypothesis at significance level $\alpha = 5\%$.

$H_0: \mu \leq 2$
$H_1: \mu > 2$

Measure	Responds
$\bar{x}$	2,87
s	0,4
s^2	0,1
n	8

Table 6-8: Mean, Variability and Number of Participants for Q21

The question Q21 asked, if the participant believes that the automated protocol generation supports the project knowledge management. 87,5% answered this question in the affirmative. The remaining responds (12,5%) see at least a partial support in knowledge management by the STACHUS framework.

The independent t-test yielded $t = 7$, $df = 7$, and $p < 0{,}00025$ one-tailed, i.e. the alternative hypothesis is accepted – the automated protocol engine generates knowledge, by externalizing implicit knowledge and makes a contribution to the transfer of knowledge by preparing the information in protocols.

H6: The STACHUS Protocol Engine reduces the information overload
At the beginning of the student's project, the participants were asked during an interview to estimate the information overload of meetings and the summarizing meeting minutes (very low (1), low (2), fair (3), high (4), very high (5)). The arithmetic mean yielded $\bar{x}_1 = 3{,}8$. The same question was asked after the completion of the mobile version of STACHUS (with $\bar{x}_2 = 2{,}8$).

Moreover, in a final questionnaire, the participants were asked to evaluate the mobile meeting application regarding its potential of reducing the information overload.

The alternative hypothesis, that STACHUS reduces the information overload will be accepted, if on average (μ) the question Q22 was answered with useful (>3) and the estimations regarding the information overload of traditional meeting minutes is statistical significant higher than the information overload of protocols generated by the STACHUS protocol engine.

$H_0: \mu \leq 3 \wedge \bar{x}_1 = \bar{x}_2$
$H_1: \mu > 3 \wedge \bar{x}_1 > \bar{x}_2$

The dependent t-test for paired samples on the data of the interviews at the beginning and end of the project yielded $\bar{d} = 1$, $s_d = 0{,}71$, $t = 3{,}2$, $df = 4$, and $p < 0{,}025$.

Measure	Responds
$\bar{x}$	3,75
s	0,9
s^2	0,8
n	8

Table 6-9: Mean, Variability and Number of Participants for Q22

The independent one sample t-test on the data of question Q22 yielded $t = 2{,}39$, $df = 7$, $p < 0{,}025$ one-tailed and a 95% confidence interval ranging from 3,0 to 4,5.

Based on these results, the null hypotheses can be rejected in favor of the alternative hypotheses: STACHUS reduces the information overload of meetings minutes, due to a more focused summarization.

H7: The STACHUS Protocol Engine supports externalization of volatile communication and thus improves the 'forgettability' of action items, decisions etc., especially during informal meetings, in comparison to the manual protocol generation from memory

The existence of a positive effect of the STACHUS framework on the volatility of discussed meeting information was evaluated during the following experiment. It is analyzed, how much information is lost, if the protocol is not written directly after the meeting (which is especially the case in informal meetings, where there it is harder to take notes during the meeting). Our hypothesis is that the mean number of recorded items is higher when protocols are automatically generated with STACHUS than in manually written protocols, summarized after the meeting from memory.

$H_0: \mu_{manual} \geq \mu_{STACHUS}$
$H_1: \mu_{manual} < \mu_{STACHUS}$

Four volunteers were asked to participate in a two-hour meeting. Afterwards they had to write a protocol of the conversation after one day and after three days. This experiment was repeated three times in different meetings. The completeness of these protocols was analyzed, that is if all mentioned items (action items, decisions, issues, etc.) are listed in the protocol. The results of the manually written protocols after one and after three days were compared to the automated notes generated by STACHUS and all mentioned items. The results are summarized in Table 6-10.

Measure	# items	#items STACHUS	#items after 1 day	#items after 3 days
$\bar{x}$	100% (22)	97,0% (21,3)	61,3% (13,3)	48,5% (10,5)
s	0	3,1	6,8	3,8
s^2	0	9,6	46,4	14,6

Table 6-10: Mean and variability for the "Forgettability-Experiment"

If it is not possible to take some notes right after the meeting, the mean of remembered items is approximately 61% after one day, and 49% after three days, respectively.

The t-test yielded $|t_{\text{man.after 1 d}}| = 10{,}5$ and $|t_{\text{man.after 3 d}}| = 25{,}5$ and $p < 0{,}001$. Thus, the null hypothesis can be rejected at a significance level $\alpha = 5\%$.

This reduced loss of meeting information, when applying the STACHUS Protocol Generator, can be supported by the results of the question Q16. 87,5% of the respondents evaluate the mobile meeting application as useful or above regarding the externalization of discussed information and thus the improvement of forgetfulness (the remaining 12,5% of the respondents answered "don't know").

Risk of validity

The sample size of this survey (n = 8) is too small to generalize our conclusions. So, the validation obtained from this survey provides only anecdotal evidence. A second case study was designed and executed to address this problem of external validity for the hypotheses 1 - 7, which is described in the following section. Moreover, four additional hypotheses are evaluated, which could not be analyzed in this case study due to the required experience of the participants, e.g. regarding the creation of traditional meetings minutes or the application of agile methods in defined processes, which could not be provided by the students.

Observation

The overall findings of the observations support the results of the questionnaire indicating the usefulness, the usability, and effort reduction for the protocol generation. Feasibility was verified by the development of a mobile iPod application. The meetings during the development project were summarized by the STACHUS protocol generator. Moreover the practicability of our meeting grammar, its usability, and the effects on the meeting conversation were tested by the students in several meetings.

Acceptance of alternative hypotheses: The statistical significance from the t-test results rejects the null hypothesis and accepts the alternative hypotheses H1, H2 and H4 to H7. This gives evidence that automatic protocol generation is useful, easy to apply, assistance in the project workaday life, and supports the knowledge management of tacit knowledge, discussed in meetings.

Although hypothesis H3 ('the meeting grammar used in the STACHUS framework is easy to learn and apply') was rejected, observations during the whole project showed that once the meeting participants were used to the grammar, it was easy for them to abide by the grammar rules.

Moreover, a correlation analysis on basis of the data from the questionnaire yielded a strong positive linear correlation between the profession and the meeting grammar evaluation – correlation coefficient $r_{xy} = 0{,}8$. Noticeable is, that traditional software developers evaluated the meeting grammar worst (mean = 2 equates difficult), whereas project managers and agile developers in our case study evaluated the grammar as easy (= 3) on average. The data is summarized in Diagram 6-2. From these results it seems reasonable to assume that traditional software developers are not used to meetings, as they seldom attend meetings, whereas project managers, and the agile developers in our case study, often participate in meetings and are used to the meeting procedures, as meetings are an essential part of agile methods. So the additional factor of familiarization to the grammar is less for them than for the software developers, who first have to familiarize with the meetings itself. However, if that is true, it would mean that project members with agile experience would evaluate the meeting grammar better than those with no agile knowledge. The correlation analysis yielded a correlation coefficient of $r_{xy} = 0{,}79$, verifying the positive linear correlation. That means that the aimed target audience (project members with lots of meetings, as it is typical in agile projects) evaluated the meeting grammar as useful (mean = 3,2, s = 0,4, t = 4, df = 5, and a p-value of $p < 0{,}1$ one-tailed), resulting in the rejection of the null hypothesis in favor of the alternative hypothesis.

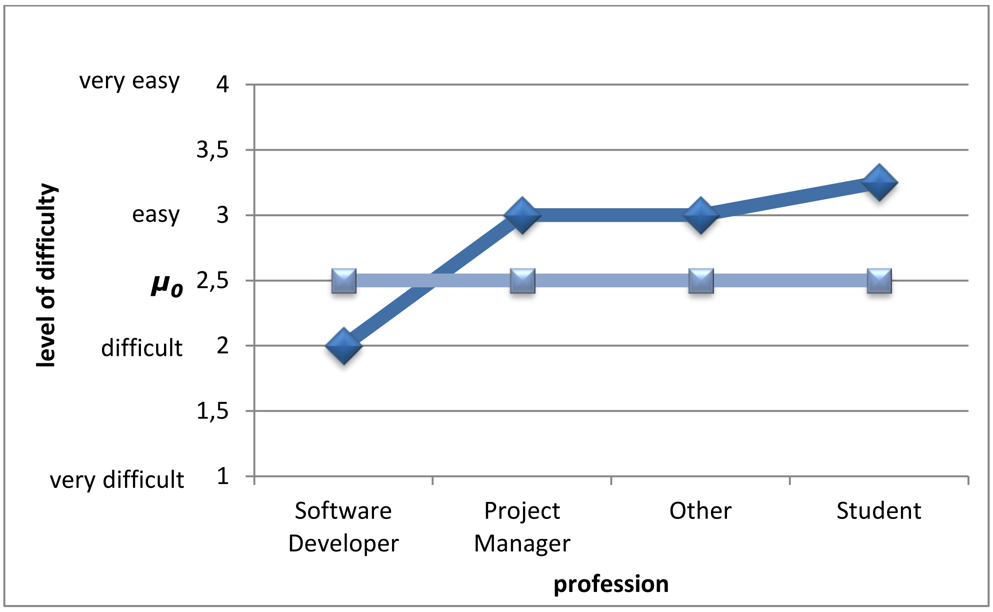


Diagram 6-2: Correlation: grammar evaluation – profession

6.3.2 Case study II

The second case study is an exploratory study of the STACHUS framework for protocol generation in business environments, and evaluated the expedience of the framework in the daily project routine. While case study I was done in an academic environment, case study II is designed for an industrial environment. So the transferability to meetings in common work-a-day life should be analyzed.

In this case study the validation of the hypotheses H1 – H5, and H8 – H11 is analyzed by observations and a self-administered questionnaire. Hypotheses H6 and H7 are evaluated only in case study I, due to the focus of informal vs. formal meeting. Case study II is also designed as a one-group pre-post case study.

The questionnaire used in the study can be found in Appendix III.

The case study took place in winter 2009/2010 and consisted of several meeting observations and an online questionnaire. To analyze the STACHUS framework in an industrial environment with defined processes, the observations had to be done at a large (international) organization. The observations were done at Siemens AG, as one example for a large company. Additionally, the questionnaire was send to large companies, e.g. Fujitsu and Capgemini AG. Software professionals and project managers were the target population for the case study and the questionnaire; ideally they had already experience with agile software development methodologies.

Meeting Observations
The meeting observations were conducted in the context of monthly Jour fixes of the process consulting team CT T DE TC 1 within Siemens Corporate Technology, consisting of ten team members. The STACHUS Protocol Engine was used to automatically generate a protocol at the end of each meeting. The meetings were evaluated regarding the practicability of the grammar, i.e. how easy it was to learn and follow the grammar during meeting conversations and discussions.

Questionnaire
The questionnaire was completed by 42 consultants, software developers and project managers with more than ten years work experience on average. The questionnaire was sent to more than ten companies, amongst other, Fujitsu Enabling Software Technology GmbH, it-economics GmbH, Capgemini AG, and Siemens AG to reach a broad distribution of meeting experiences and protocol practices.

In the following the results of the statistical tests are summarized, structured according to the hypotheses from chapter 6.2.

H1: The STACHUS Protocol Engine reduces the effort of protocol generation in comparison to manually written protocols
To validate this hypothesis the mean effort for manually writing a protocol is compared to the mean effort for automatically generated protocols by the STACHUS Protocol Engine. The hypothesis is tested at a significance level of 5%.

$$H_0: \mu_{manual} \leq \mu_{STACHUS}$$
$$H_1: \mu_{manual} > \mu_{STACHUS}$$

According to the respondents' data, the mean effort for manually writing a protocol was between 10 and 85.5 minutes, depending on the meeting duration. In contrast, the effort for the automated generation of meeting protocols was approximately five minutes for starting the STACHUS Protocol Engine and stopping the meeting recording. The detailed numbers of the questionnaire for writing a protocol are summarized in Table 6-11.

Meeting duration	< 30 min	30 min - 1 h	1 - 2 h	2 - 4 h	4 - 8 h
Mean effort (in min): manual ($\bar{x}$)	10	18,1	30,2	52,1	85,5
Mean effort (in min): Automated ($\bar{x}$)	5	5	5	5	5

Table 6-11: Effort and duration for manually and automated protocol generation

Table 6-12 shows, that the mean effort for writing a protocol manually is higher than the mean effort for starting the STACHUS Protocol Engine to automatically generate a protocol. That is, time savings between 5 and 80,5 minutes per protocol can be achieved, when the protocol is automatically generated.

The results of the independent t-test on these data, depending on the meeting duration, yielded $t \geq 5{,}4,\ 9 \leq df \leq 25,$ and $p < 0{,}000025$ one-tailed. Table 6-12 gives the detailed results of the statistical test.

Meeting duration	< 30 min	30 - 60 min	1 - 2 h	2 - 4 h	4 - 8 h
Mean effort (in min) ($\bar{x}$)	10	18,1	30,2	52,1	85,5
s	0	12,41	23,09	28,14	32,90
s^2	0	153,99	532,96	791,84	1082,25
n	27	26	25	14	10
t	-	5,38	5,46	6,26	7,74
p	-	<0,00001	<0,00001	<0,000025	<0,000025
95% confi-dence interval	-	13,1 to 23,1	20,7 to 39,7	35,9 to 68,3	62,0 to 109,0

Table 6-12: Results: effort for manual protocol generation

As the calculated p-value is below the threshold of p = 0,05 for statistical significance, the null hypothesis can be rejected in favor to the alternative hypothesis, which validates the results of the first case study also for business environments.

H2: STACHUS Protocol Engine (audio recording & protocol generation) ensures easiness in communication, i.e. the communication is not complicated or constrained by audio recordings and the process of automated protocol generation

The case study also analyzed, beside the application of the framework (H3) and handling and easy of learning of the grammar (H4), the effects on the communication behavior. The communication behavior is evaluated in question Q23.

80% of the respondents answered the question, if they believe in a changed communication behavior, when using STACHUS, in the affirmative. However mostly a positive effects is seen, e.g. “more meeting discipline” or “less meaningless discussions”.

These results are confirmed by observations during several meetings during the case study.

For this reason, the team members were asked, to evaluate the communication behavior of a traditional meeting and of a STACHUS-meeting. If the hypothesis is right, the evaluation should increase, that is

H_0: $\mu_{trad} \geq \mu_{STACHUS}$
H_1: $\mu_{trad} < \mu_{STACHUS}$

The results are summarized in Table 6-13.

Measure	Traditional meeting mgmt	STACHUS meetings
$\bar{x}$	1,72	2,47
s	0,78	0,51
s^2	0,61	0,26

Table 6-13: Mean and Variability for the communication behavior evaluation

The dependent t-test for paired samples on the data of the interviews yielded $\bar{d} = 0{,}75$, $s_d = 0{,}65$, $t = 6{,}94$, $df = 35$, and $p < 0{,}0001$, so we can reason that STACHUS has positive effects on the communication behavior of meetings.

H3: The STACHUS meeting grammar is easy to learn and apply and does not constrain the meeting progress

This hypothesis will be accepted, if the majority of the respondents verify the easiness of the grammar, i.e. the respondents evaluate the grammar as easy and fast to learn, as the rules are simple to keep in mind and adopt during a meeting. As the response choices of the corresponding research question Q27 are coded with natural numbers from "very difficult" (1) in ascending order to "very easy" (4), the arithmetic mean should be larger than 2,5 to reject the null hypothesis (again at a significance level of 5 %).

H_0: $\mu \leq 2{,}5$
H_1: $\mu > 2{,}5$

Measure	Responds
$\bar{x}$	3,03
s	1,0
s^2	1,0
n	36

Table 6-14: Mean, Variability and Number of Participants for Q27

The independent t-test on this data yielded $t = 3{,}17$, $df = 35$, $p < 0{,}0025$ one-tailed, and a 95% confidence interval ranging from 2,69 to 3,37, which results in the acceptance of the alternative hypothesis and rejection of the null hypothesis at a 5 % significance level.

H4: STACHUS Audio Recorder and Protocol Generator are easy to use

The hypothesis H4 is accepted, if the majority of the respondents confirm an easy usability of the audio recorder and protocol generator, and an intuitive user prompting to easily start the audio recorder and create a protocol. As the response choices of the corresponding research question Q20 are coded using natural numbers, that is, "yes" is coded with 3, "fair" with 2, and "no" is coded with 1, the mean 'usability' should be larger than 2 to accept the alternative hypothesis.

H_0: $\mu \leq 2$

H_1: $\mu > 2$

Measure	Responds
$\bar{x}$	2,44
s	0,73
s^2	0,53
n	9

Table 6-15: Mean, Variability and Number of Participants for Q20

The independent one-sample t-test on these data yielded $t = 1{,}84$, $df = 8$, $p \approx 0{,}05$ one-tailed, and a 95% confidence interval ranging from 1,89 to 3,00.

The hypothesis is accepted that the audio recorder and protocol generator are easy to handle, i.e. no comprehensive trainings are required to use the tool, and the audio recorder can easily and without additional effort be started.

H5: The STACHUS Compiler & Protocol Engine support knowledge management by externalizing tacit information

The participants of several meetings were asked during an interview if they see a positive effect of the STACHUS protocol generator on knowledge management, by externalizing meeting information and generating a protocol.

This hypothesis will be accepted, if the majority of the respondents verify a positive effect on knowledge management by STACHUS, i.e. that the STACHUS Protocol Engine generates knowledge and makes a contribution to the transfer of knowledge, by externalizing implicit knowledge and preparing in protocols. The response choices are coded with natural numbers (no (1), partly (2), yes (3)), so the average (μ) should be larger than 2 to reject the null hypothesis at significance level $\alpha = 5\%$.

$H_0: \mu \leq 2$
$H_1: \mu > 2$

Measure	Responds
$\bar{x}$	2,28
s	0,63
s^2	0,40
n	39

Table 6-16: Mean, Variability and Number of Participants

The independent t-test yielded $t = 2,77$, $df = 38$, and $p < 0,005$ one-tailed, i.e. the alternative hypothesis is accepted – the automated protocol engine generates knowledge, by externalizing implicit knowledge and makes a contribution to the transfer of knowledge by preparing the information in protocols.

H6: The STACHUS Protocol Engine reduces the information overload
The hypothesis H6 is evaluated only in case study I, as information overload of meetings and meeting minutes was claimed only by the participants of case study I, and could not be observed in case study II.

H7: The STACHUS Protocol Engine supports externalization of volatile communication and thus improves the 'forgettability' of action items, decisions etc., especially during informal meetings, in comparison to the manual protocol generation from memory
Hypothesis H7 is analyzed and described within case study I during an extra experiment, which is not repeated in case study II. Additionally, the focus of case study I, with the implementation of a smart-phone prototype, is on informal meetings, whereas case study II is more focused on traditional meetings in an industrial environment.

H8: The STACHUS Protocol Generator accelerates the protocol generation, i.e. the protocols are published faster than manually written protocols
The hypothesis is evaluated only in this case study, as in case study I no manual protocols were written. Hypothesis H8 is accepted, if the mean effort for manually writing a protocol is larger

than the mean duration for automatically generated protocols by the STACHUS Protocol Engine.

H0: $\mu_{manual} \leq \mu_{STACHUS}$

H1: $\mu_{manual} > \mu_{STACHUS}$

Meeting duration	< 30 min	30 min - 1 h	1 - 2 h	2 - 4 h	4 - 8 h
Mean effort (in min): manual ($\bar{x}$)	10	18,1	30,2	52,1	85,5
Mean duration (in min): automated ($\bar{x}$)	8	13	22	38	59
s	0	12,41	23,09	28,14	32,90
s^2	0	153,99	532,96	791,84	1082,2
n	27	26	25	14	10
t	-	2,10	1,78	1,87	2,55
p	-	<0,025	<0,05	<0,05	<0,025
95% confidence interval	-	13,1 to 23,1	20,7 to 39,7	35,9 to 68,3	62,0 to 109,0

Table 6-17: Mean, Variability and Number of Participants for Q8

The t-test on this data yielded $1{,}78 \leq t \leq 2{,}55$, $9 \leq df \leq 25$, and $p < 0{,}05$ one-tailed, which results in the acceptance of the alternative hypothesis – the protocols generated by STACHUS are faster generated than manually written ones – and rejection of the null hypothesis at a 5 % significance level.

In addition, the respondents stated in their answers to questions Q10 and Q11 that it takes on average 2,28 days until the minutes taker finds time to manually write the protocol and send it, which in 17% of the cases was too late to allow an adequate preparation for the next meeting. The automated creation and publication of a protocol, generated by the STACHUS Protocol Engine, takes maximal 1 hour.

H9: The STACHUS Protocol Engine is compliant with agile practices (lightweight, flexibility, avoiding waste)

The hypothesis H9 claims, that the STACHUS Protocol Engine internalizes and supports the lightweight and flexible, adaptable character of agile methods and practices, and follows the principle of avoiding waste. To evaluate the STACHUS framework regarding a compliance with the agile practices, only evaluations of experts (respondents with practical experience in agile methods) were regarded, so the hypothesis was not analyzed in case study I.

The response choices of the corresponding question Q21 are coded with natural numbers ("no" (1), "yes, partly" (2), "yes" (3)), so the arithmetic mean 'agile support' should be larger than 2 to accept the alternative hypothesis at a significance level $\alpha = 5$. The hypothesis will be accepted, if on average the respondents confirm the involvement and compliance of the agile practices in the STACHUS framework.

H_0: $\mu \leq 2$
H_1: $\mu > 2$

Measure	Responds
$\bar{x}$	2,28
s	0,6
s^2	0,4
n	32

Table 6-18: Mean, Variability and Number of Participants for Q21

90,6% of the respondents believe that the STACHUS Protocol Engine is compliant with the agile methods and supports the agile practices and principles, due to a lightweight approach and reduced effort for writing meeting minutes.

The independent t-test yielded t = 2,5, df = 31, p < 0,01 one-tailed, and a 95% confidence interval ranging from 2,05 to 2,51, i.e. the hypothesis is accepted, that the STACHUS protocol generator is compliant with the lightweight and flexible character of agile methods and supports the agile methods with an knowledge management approach for externalizing tacit knowledge without increasing the effort.

H10: The STACHUS Protocol Generator reduces gaps between tools in the project & meeting workflow and thus simplifies the meeting process

The evaluation of Q14 showed that 82,9% of the respondents use electronic tools for the management and administration of their tasks, for instance Outlook, Excel, Bugzilla, XPlanner, AT-notes, or a task management tools on their mobile phone. The STACHUS Protocol Generator integrates task and project management tools, e.g. XPlanner and Unicase (chapter A.2.1.2), in the meeting workflow and protocol generation process and thus reduces the information gabs that occur if information, discussed in the meeting, has to be manually transferred to these tools.

As in case study I only paper and white boards were used as task-management tool, H10 is evaluated only in case study II.

The NHST will accepts the hypothesis H10 at a 5% significance level, if the majority of the respondents confirm that applying the STACHUS Protocol Generator simplifies the meeting workflow, as project and task management tools are integrated and the results of the meeting (like new action items) are documented in a protocol and additionally automatically transferred to the project and task management tools (i.e. a new action item is created in the task tool). As the response choices of the corresponding research question Q19 are coded using natural numbers ("yes, definitely" (4), "yes, probably" (3), "no" (2), and "no, even additional work" (1)), the arithmetic mean should be larger than 2,5 to accept H10.

H_0: $\mu \leq 2{,}5$
H_1: $\mu > 2{,}5$

Measure	Responds
$\bar{x}$	2,82
s	0,76
s^2	0,57
n	38

Table 6-19: Mean, Variability and Number of Participants for Q19

The independent one-sample t-test on the data of Q19 yielded t = 2,58, df = 37 , p < 0,01 one-tailed, and a 95% confidence interval ranging from 2,57 to 3,06. Based on these results, the null hypotheses can be rejected in favor of the alternative hypotheses at a 5% significance level.

H11: Protocol generation procedure (STACHUS Protocol Engine and STACHUS Compiler) can be improved by a dynamically extending grammar

The objective was to analyze, if the protocol quality can be improved on average by factoring the project vocabulary into the automated protocol generation process by the STACHUS Protocol Engine.

Several project meetings at Siemens were recorded and analyzed, that included on average 14 meeting phrases. Each phrase consisted of a meeting element (like information, issue or action item) and its description.

To validate this hypothesis, two experiments were done. In the first experiment, the STACHUS Protocol Engine was started with the default vocabularies to create a protocol form the meeting audio stream.

In the second experiment, the STACHUS Compiler was fed with project dependent vocabulary, like names of the team members, special abbreviations, and project domain vocabulary (e.g. methods, tools, etc.), and afterwards the STACHUS Protocol Engine was updated. Then the protocol was generated a second time.

Our hypothesis is that the quality of the generated protocol can be improved by the integration of project vocabulary. This means that the mean quality index (consisting of the number of documented meeting elements, their correctness and completeness) of the second experiment (μ) is higher than the mean of the first experiment (μ_0).

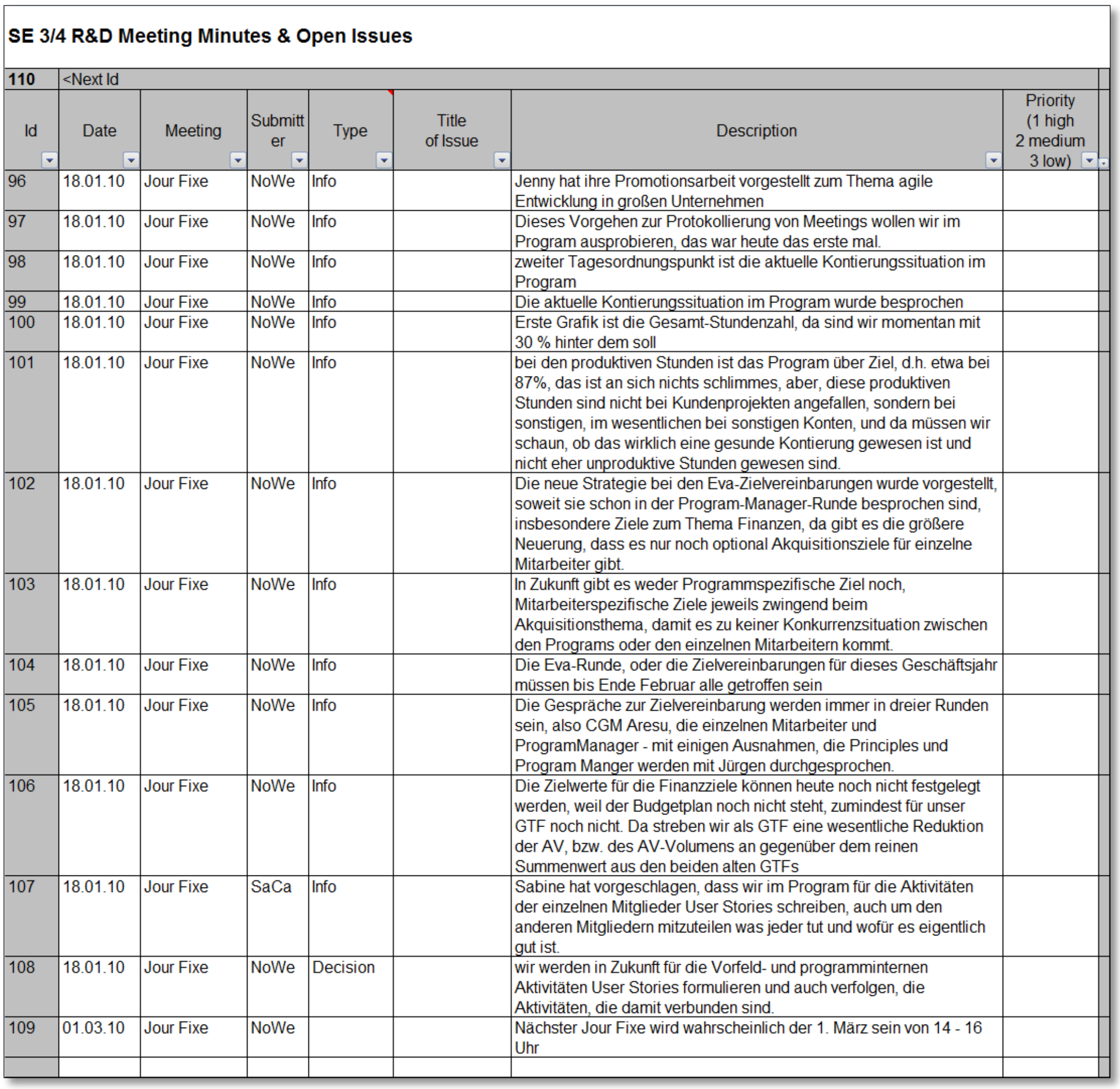
H_0: $\mu \leq \mu_0$

H_1: $\mu > \mu_0$

Measure	# total	Experiment 1	Experiment 2
# meeting elements (phrases)	14 (100%)	12 (85,7%)	14 (100%)
# words total	345 (100%)	202 (58,6%)	280 (81,2%)
Ø words/ phrase	24,6	14,4	20
quality index	100%	59,6%	81,9%

Table 6-20: Results of the experiments regarding improvement of protocol quality

The dependent t-test for paired samples on the data of the two experiments yielded $\bar{d} = 5{,}57$, $s_d = 4{,}38$, $t = 4{,}76$, $df = 13$, and $p < 0{,}00025$. Thus the alternative hypothesis is accepted, the automated protocol generation can be improved by a dynamically extending grammar. An example for an automated protocol, generated during the case study, is shown in Figure 6-1.



SE 3/4 R&D Meeting Minutes & Open Issues

110	<Next Id						
Id	Date	Meeting	Submitter	Type	Title of Issue	Description	Priority (1 high 2 medium 3 low)
96	18.01.10	Jour Fixe	NoWe	Info		Jenny hat ihre Promotionsarbeit vorgestellt zum Thema agile Entwicklung in großen Unternehmen	
97	18.01.10	Jour Fixe	NoWe	Info		Dieses Vorgehen zur Protokollierung von Meetings wollen wir im Program ausprobieren, das war heute das erste mal.	
98	18.01.10	Jour Fixe	NoWe	Info		zweiter Tagesordnungspunkt ist die aktuelle Kontierungssituation im Program	
99	18.01.10	Jour Fixe	NoWe	Info		Die aktuelle Kontierungssituation im Program wurde besprochen	
100	18.01.10	Jour Fixe	NoWe	Info		Erste Grafik ist die Gesamt-Stundenzahl, da sind wir momentan mit 30 % hinter dem soll	
101	18.01.10	Jour Fixe	NoWe	Info		bei den produktiven Stunden ist das Program über Ziel, d.h. etwa bei 87%, das ist an sich nichts schlimmes, aber, diese produktiven Stunden sind nicht bei Kundenprojekten angefallen, sondern bei sonstigen, im wesentlichen bei sonstigen Konten, und da müssen wir schaun, ob das wirklich eine gesunde Kontierung gewesen ist und nicht eher unproduktive Stunden gewesen sind.	
102	18.01.10	Jour Fixe	NoWe	Info		Die neue Strategie bei den Eva-Zielvereinbarungen wurde vorgestellt, soweit sie schon in der Program-Manager-Runde besprochen sind, insbesondere Ziele zum Thema Finanzen, da gibt es die größere Neuerung, dass es nur noch optional Akquisitionsziele für einzelne Mitarbeiter gibt.	
103	18.01.10	Jour Fixe	NoWe	Info		In Zukunft gibt es weder Programmspezifische Ziel noch, Mitarbeiterspezifische Ziele jeweils zwingend beim Akquisitionsthema, damit es zu keiner Konkurrenzsituation zwischen den Programs oder den einzelnen Mitarbeitern kommt.	
104	18.01.10	Jour Fixe	NoWe	Info		Die Eva-Runde, oder die Zielvereinbarungen für dieses Geschäftsjahr müssen bis Ende Februar alle getroffen sein	
105	18.01.10	Jour Fixe	NoWe	Info		Die Gespräche zur Zielvereinbarung werden immer in dreier Runden sein, also CGM Aresu, die einzelnen Mitarbeiter und ProgramManager - mit einigen Ausnahmen, die Principles und Program Manger werden mit Jürgen durchgesprochen.	
106	18.01.10	Jour Fixe	NoWe	Info		Die Zielwerte für die Finanzziele können heute noch nicht festgelegt werden, weil der Budgetplan noch nicht steht, zumindest für unser GTF noch nicht. Da streben wir als GTF eine wesentliche Reduktion der AV, bzw. des AV-Volumens an gegenüber dem reinen Summenwert aus den beiden alten GTFs	
107	18.01.10	Jour Fixe	SaCa	Info		Sabine hat vorgeschlagen, dass wir im Program für die Aktivitäten der einzelnen Mitglieder User Stories schreiben, auch um den anderen Mitgliedern mitzuteilen was jeder tut und wofür es eigentlich gut ist.	
108	18.01.10	Jour Fixe	NoWe	Decision		wir werden in Zukunft für die Vorfeld- und programminternen Aktivitäten User Stories formulieren und auch verfolgen, die Aktivitäten, die damit verbunden sind.	
109	01.03.10	Jour Fixe	NoWe			Nächster Jour Fixe wird wahrscheinlich der 1. März sein von 14 - 16 Uhr	

Figure 6-1: Meeting minutes of the meeting from 18.01.2010 at Siemens, published in MS Excel

Observation

The overall findings of the observations of the second case study support the results of the questionnaire. The participants of the meetings got quickly accustomed to the meeting grammar and accepted the recordings during the meeting. The meeting participants evaluated the STACHUS Protocol Engine as useful on average for their project workdays, and the results show, that the evaluation increases, when the respondents attend many meetings per week (20 h/ week) (see Diagram 6-3).

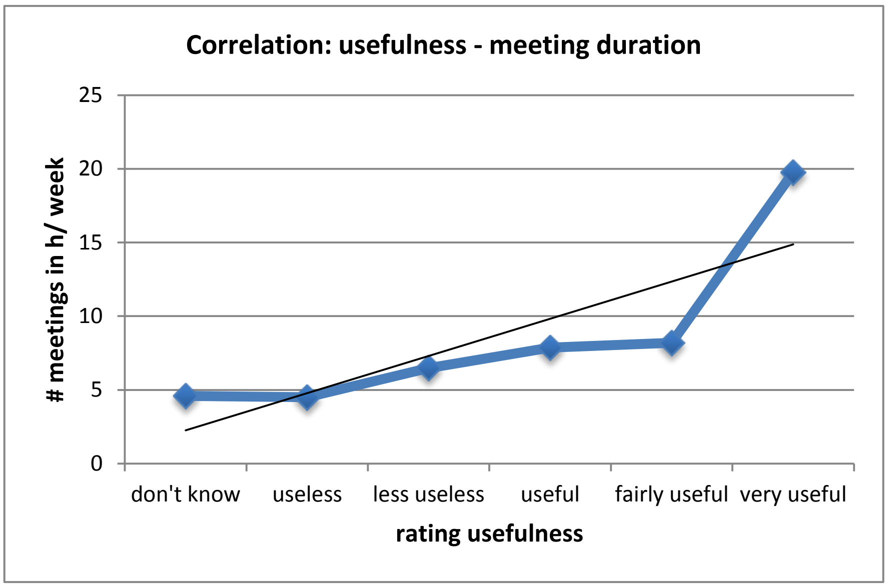


Diagram 6-3: Correlation: evaluated STACHUS usefulness - meetings

The statistical significance from the t-test results rejects the null hypotheses and accepts all evaluated hypotheses (H1 – H5 and H8 – H11). This gives evidence that automatic protocol generation is useful and reduces the manual effort for protocol generation, so the protocols are published faster. Moreover, the automatic protocol generation is assistance in the project work-aday life, compatible also with agile practices and principles.

Validity

The results of this study, such as the effort reduction for protocol generation, confirm the results of the previous study, however based on a larger experimental group with more than 40 participants. This sampling size allows us to be 95% confident in the result, accepting an error of 15% (Fink, 1995). For this reason, we judge the results of this case study with a strong external validity that can be generalized.

6.3.3 Case study III

The third case study was conducted within the DOLLI project (Distributed Online Logistic and Location Infrastructure) at the Technische Universität München (Bruegge, et al., 2009). The DOLLI project consisted of three sub-projects, DOLLI (in winter term 2007), DOLLI2 (2008) and DOLLI3 (2009). The project's goal was to develop a prototype for a real problem posed by a real customer, the Munich Airport. The developers were organized into four to five sub teams in each project (Team Telemetry, Team Interaction, Team Lavis, Team Airtouch, Team Architecture, Team Optimizer, Team Simulation, Team Reasoning, and Team Facility Management).

The case study took place in 2008 and 2009 analyzing teams of the DOLLI2 and DOLLI3 project.

The objective of this case study is to analyze the validation of the hypothesis H12 – protocols generated by the STACHUS Protocol Engine are more correct (complete and less errors) than manually written meeting minutes – to complete the evaluation of the previous case studies. H12 could not be evaluated in the previous case studies, as only in case study III several manual and automated protocols for the same meetings were available to be compared.

During the case study two student teams were observed, Telemetry System (DOLLI2) and Optimizer (DOLLI3), and their meeting protocols were evaluated. The teams were selected based on their experience with agile methods and their cooperativeness and willingness to participate in this experiment. The teams consisted of six and five team members respectively.

During the case study, the weekly team meetings of these two teams were recorded. The students had to write protocols for each meeting, which were afterwards compared to the protocols generated by STACHUS from the stored recordings.

In the following the results of the statistical tests regarding hypothesis H12 are summarized.

H12: The STACHUS Compiler and STACHUS Protocol Engine improve the correctness of protocols

The correctness of a protocol is evaluated as the completeness of a protocol, i.e. if all, or how many discussed meeting elements are documented in the protocol. Additionally, the numbers of typing errors and false information is regarded to estimate the correctness of a protocol.

The evaluation of our questionnaire showed that almost 50% of the respondents are annoyed at incorrect and incomplete protocols. Moreover, 48,9% of the participants of the questionnaire expect a reduction of errors in automatically generated protocols by STACHUS.

To validate the hypothesis H12 the mean completeness of manually written protocols is compared to the mean completeness of protocols generated by the STACHUS Protocol Engine. The hypothesis is tested at a significance level of 5%, i.e. the hypothesis H12 is accepted, STACHUS Compiler and Protocol Engine improve the correctness of protocols compared to the correctness of manually written meeting minutes.

H_0: $\mu_{manual} \geq \mu_{STACHUS}$
H_1: $\mu_{manual} < \mu_{STACHUS}$

Measure	# meeting Elements	# STACHUS	#manual
$\bar{x}$	100% (14,4)	97,8% (14)	49,7% (7)
s	0	3,1	18,8
s^2	0	9,4	353,2

Table 6-21: Mean and variability for the "Completeness-Experiment"

The dependent t-test for paired sample, comparing STACHUS generated and manually written protocols, yielded $\bar{d} = 48{,}1$, $s_d = 19{,}36$, $t = 5{,}55$, $df = 4$, and $p < 0{,}005$ one-tailed. Thus, the null hypothesis can be rejected at a significance level $\alpha = 5\%$.

Additionally, the number of typing errors and false information in the protocols was analyzed.

Measure	# Errors manual	# Errors STACHUS
$\bar{x}$	5,6	2,0
s	5,13	1,58
s^2	26,3	2,5
n	7	7

Table 6-22: Mean and Variability: aggregated typing errors & false information

That is, the mean number of errors and false information is higher in manually written protocols than in meeting minutes generated by the STACHUS Protocol Engine. However, the improvement of automatically generated protocols compared to manually written protocols is not statistically significant.

This case study showed that the number of typing errors and false information can be reduced by STACHUS, when automatically generating protocols compared to manually written meeting minutes. Additionally, and even more important, the improvement by STACHUS of the degree of completeness of discussed meeting elements, like decisions, action items, or issues, was verified with a statistical significance.

The hypothesis H12 is accepted based on the statistical significance from the t-test and gives evidence that the automatic protocol generation not only reduces the manual effort for protocol generation, but also improves the completeness that is, that more/ all discussed meeting elements are documented in the protocol, with less errors.

Conclusion

Evaluations of the questionnaire showed that approximately 90 minutes are spent per meeting for manual meeting post-processing and protocol generation for 15 meetings on average per week. Thus, there is a potential for effort reduction in the projects, which verifies the need for a protocol automating tool like the STACHUS framework.

Our framework was analyzed in three case studies, verifying the hypotheses H1 – H12 with a statistical significance, although with a sampling size of (only) 40 – 50 participants. The observations during the case studies regarding the applicability and quality of the STACHUS protocol generator for externalizing information in formal and informal meetings were confirmed by the statistical tests.

The evaluations (observations, interviews, and questionnaire) showed that the meeting participants got used to the STACHUS grammar and followed the rules during the meetings; however the acceptance of an automated protocol generator could be increased, if the grammar is simplified with fewer restrictions to detect meeting phrases. Nevertheless, the overall opinion and evaluation of the STACHUS framework by the participants is positive; it is seen as a valuable and effective tool for information externalization.

7 Conclusion

Summary and outlook

As, more and more companies are introducing agile methods in their software development lifecycles, especially in large organizations, a dedicated knowledge management is essential for a successful integration of the agile practices and principles in a defined process world. The missing focus of knowledge externalization and knowledge management is a weakness of agile methods. This dissertation addressed the externalization of information from formal and informal meetings and the creation of meeting minutes. Our approach of automated information extraction from meetings tried to address this problem of missing documentation and to enhance knowledge management.

Our collection of twelve hypotheses states that it is possible to follow an agile method in a large organization with defined processes. The agile methods can be enriched by a knowledge externalization approach that is in compliance with the agile philosophy of light-weightness and applicable without additional effort for the team.

We developed a framework for automated protocol generation called STACHUS. As integral constituent of this framework, we employed a self-improving Protocol Compiler and Protocol Engine to generate automated meeting minutes.

Furthermore, we claim that traditional meetings can also benefit from this automated protocol generation approach by effort reduction, faster publication of the meeting minutes, and improvement of the protocol quality.

In the following, we present the main contributions of this work and an outlook on future steps in the field of automated meeting protocols.

7.1 Contribution

The outcome of the dissertation is STACHUS, a framework for *automated protocol generation*. STACHUS was realized as a prototype to show feasibility and to evaluate the automated protocol generation process. In several projects the framework was applied to validate our hypotheses that an automated protocol generation reduces time and effort for the creation of meetings minutes compared to the manual creation, and improves the protocol quality, as the protocols are 'more complete' and contain less errors.

The STACHUS framework supports externalization of information and knowledge by protocol generation and the maintenance of *project and task management tools*. The results of a meeting are automatically documented in a protocol, and can easily be transferred to other tools.

The STACHUS Compiler generates the STACHUS Engine, which creates the protocols from the meeting audio stream. With the STACHUS Engine, our framework for protocol generation is enhanced by a *self-improving component*. Regularly, project context and updates of the vocabulary, system model and created documents are considered by the STACHUS Compiler and integrated in the protocol generation process, to improve the protocol quality. The influence of the STACHUS Compiler on the generated protocols was evaluated and verified in chapter 6.

STACHUS supports the integration and acceptance of *agile meeting methods*, also in a traditional environment. The added knowledge management focuses also beyond the individual project, so projects can learn from the experiences of other projects in a company-wide learning process and the development processes can be improved.

7.2 Future work

We believe that the STACHUS framework can improve the acceptance and introduction of agile practices and principles in the software development lifecycles of large organization with defined process control models.

Several further possibilities of improving the STACHUS framework have been identified, which will be briefly described.

Within the time frame of the research, this dissertation considered and implemented three interfaces to *project and task management tools* (see chapter Appendix II). However a comprehensive introduction and successful application of the STACHUS prototype in all meetings of the workaday life requires the support of further tools, for example ScrumWorks (CollabNet, Inc. , 2010), as one of the most frequently applied tools in agile software development projects, according to the agile survey 2009 (VersionOne, 2010).

Additionally, the *meeting taxonomies* and the *project management vocabularies* (necessary for the detection and externalization of meeting information) have to be extended. The idea is to support further meeting types, for instance Kick-off meetings or Customer Acceptance Test meetings, as well as software development independent meetings, for example budget meetings or director's meetings.

The STACHUS *grammar* has to be analyzed, to integrate possibilities for simplification. Although, the developed grammar was applied during the empirical evaluation procedure and accepted by the meeting participants and valued as easy to learn and to follow during a meeting, simplifications are desirable. So the acceptance and pervasiveness of automated protocol generation and our framework can be increased. For this reason the reduction of grammar rules has to be investigated and the replacement, for example by gestures or movements has to be analyzed. This would allow for instance following scenario: instead of saying "new action item … action item end.", the meeting participant could wave with his right hand or shake his iPhone. So the system is signalized to start recording the meeting sequence, as a part is following which has to be integrated in the protocol.

The STACHUS Compiler currently considers the system model, project vocabulary and existing documents e.g. old protocols to create the *knowledge sources* as part of the STACHUS Protocol Engine. However, further sources could be integrated in the protocol engine generation process, for example text from chat sessions, e-mails, forum entries, or wiki pages that are integral part of the projects.

The STACHUS framework currently creates protocols but does not handle the optimal storage of the generated meeting protocols and how information can effectively and efficiently be found in old protocols. The framework has to be extended by an *indexing and information retrieval* system for an optimal usage of the externalized knowledge.

Information is discussed not only in meetings. Other *sources of information* have to be integrated in a comprehensive knowledge management process. In e-mail conversations, forums, or chats, information is exchanged in a communicative way that has to be considered and integrated in the companywide knowledge management workflow in a future step, to document problems, decisions, and action items, mentioned somewhere else than in a meeting.

Our hypotheses were evaluated with teams of 5 – 12 team members. However, the protocol generation process works also for large teams applying *Meta-Scrum,* published by Sutherland (2005). The individual sub-team meetings and the *scrum of scrums meetings* are an important technique in scaling Scrum to large project teams. This type of meetings allows clusters of teams to discuss their work, focusing especially on areas of overlap and integration. Each scrum of scrum meeting occurs daily with on average seven participants (Schwaber & Beedle, 2002). So we infer from the evaluated meetings to the applicability of STACHUS in large projects that apply Meta-Scrums – however this has to be evaluated in a further case study.

The STACHUS framework is build on existing speech recognition engines. As these engines only recognize speech from well-known users and under defined situations (e.g. no background noise and a very good microphone), we are relying on *improvements in the speech recognition* field. Currently most speech recognizers are user dependent and require trainings in advance. This problem is addressed in our framework by the combination of several speech engines; however improved speech recognition would improve the protocol quality of our system.

The combination of video and audio can be investigated to improve the speech recognition. If a video tool recognizes the meeting participant in the video and detects who is speaking, the right speaker profile of the speech recognition engine can be applied, to analyze the audio stream of this part of conversation and externalize the information.

Concluding, STACHUS and the process of automated meeting minutes generation is work in progress. The contributions described in this dissertation helped to further improve the automated protocol generation and created a reliable approach for knowledge externalization. However, the practical use of the framework also revealed weaknesses and improvement potential, which need to be covered to increase the acceptance and practicability in the project lifecycle.

Appendix I: Grammar

An exemplarily meeting grammar of the iPhone student project

This chapter presents an exemplarily grammar G and its rules, that was applied in the iPhone student project (see chapter 6.2) in the summer term 2009.

A grammar is defined as the ordered quad-tuple:

$$G = (N, T, P, S)$$

That is, the grammar G comprises

- a finite set N of nonterminal symbols,
- a finite set T of terminal symbols, where $N \cap T = \emptyset$
- a finite set P of production rules with a left and right-hand side consisting of a sequence of these symbols: $(N \cup T)^{*}N(N \cup T)^{*} \to (N \cup T)^{*}$
- and a start symbol S, where $S \in N$

To allow an easy, modular, and automatic extensibility of the grammar, it is decomposed into one superior grammar (the iPhone project grammar) and three further parts: the grammar for the project vocabulary, the grammar of the address book, and the grammar that covers the system model and project domain. The details of the grammars are given below.

Grammar: iPhone project

G = (N, T, P, S)

N = {WSE, Txt, TM, Descr, Date, Day, Month, SM, Sign, Letter, Number, Symbol}

T = {*new, end,*
Monday, Tuesday, Wednesday, Thursday, Friday,
January, February, March, April, Mai, June, July, August, September, October, November, December,

a, b, c, d, e, f, g, h, I, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z,
0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

P = {S → *new* WSE Txt WSE *end*
WSE → PM | SE
Txt → Descr | Descr TM Descr | Descr TM Descr Date Descr| Descr Date Descr
Descr → SM | Sign | Descr SM Descr
Date → Number | Day | Month
Day → *Monday* | *Tuesday* | *Wednesday* | *Thursday* | *Friday*
Month → *January* | *February* | *March* | *April* | *Mai* | *June* | *July* | *August* | *September* | *October* | *November* | *December*
Sign → Letter | Number | Letter Sign | Number Sign
Sign → ε
Letter → *a* | *b* | *c* | *d* | *e* | *f* | *g* | *h* | *i* | *j* | *k* | *l* | *m* | *n* | *o* | *p* | *q* | *r* | *s* | *t* | *u* | *v* | *w* | *x* | *y* | *z*
Number → *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9*
}

The project vocabulary comprises, beside classical project management vocabulary (PM) like 'decision' or 'action item', in addition the elements of Scrum (SE), like 'impediment' or 'Sprint Backlog'.

Grammar: project vocabulary

G = (N, T, P, S)

N = {PM, SE, AI, D, I, P, Req, FR, NFR, Info, BI, SBI, Status}

T = {*Action Item, Task, ToDo, Issue, Problem, Impediment, Decision, Proposal, Suggestion, Idea, Requirement, functional Requirement, non-functional Requirement, Info, Information, Backlog Item, Sprint Backlog Item, Status, User Story}*

P = {S → PM | SE
PM → AI, D, I, P, Req, FR, NFR, Date, Info, Status
SE → BI, SBI
AI → *Action Item* | *Task* | *ToDo* | *User Story*
I → *Issue* | *Problem* | *Impediment*
D → *Decision*
P → *Proposal, Suggestion, Idea*
Req → *Requirement*
FR → *functional Requirement*
NFR → *non-functional Requirement*
Info → *Info* | *Information*

BI → *Backlog Item*
SBI → *Sprint Backlog Item*
Status → *Status*
}

In the address book, all team members, customers, and stakeholders modeled.

Grammar: address book

G = (N, T, P, S)

N = {TM}

T = {*Yuliya, Nikolay, Nik, Petromil, Yonata, Mahmoodreza, Mahmood}*

P = {S → TM
TM → *Yuliya* | *Nikolay* | *Nik* | *Petromil* | *Yonata* | *Mahmoodreza* | *Mahmood* | ε
}

Grammar: system model

G = (N, T, P, S)

N = {SM, Meeting, Speech, Architecture}

T = {*meeting, participant, location, date, room, agenda, protocol, minutes, summary, information, retrieval, discussion, topic, minutes taker, facilitator, speech, recognition, conversation, Sphinx, Naturally Speaking, MacSpeech, keyword, training, user independent architecture, framework, backend, server, connection, interface, mobile, iPhone, smart phone, Cocoa, upload, download, synchronize, picture, map, google, list*
}

P = {S → SM
SM → Meeting | Speech | Architecture
Meeting → *meeting* | *participant* | *location* | *date* | *room* | *agenda* | *protocol* | *minutes* | *summary* | *information* | *retrieval* | *discussion* | *topic* | *minutes taker* | *facilitator*
Speech → *speech* | *recognition* | *conversation* | *Sphinx* | *Naturally Speaking* | *MacSpeech* | *keyword* | *training* | *user independent*
Architecture → *architecture* | *framework* | *backend* | *server* | *connection* | *interface* |

```
                    mobile | iPhone | smart phone | Cocoa | upload |  download |
                    synchronize | picture | map | google | list
}
```

The system model can become very large. Hence, the system model described above is only an excerpt of some of the most important and often spoken words of the iPhone project.

Base on the defined grammar and its rules, for instance, following typical phrases can be spoken and recognized during a meeting:

- New action item for Yonata: create a manual. Action item end.
- New action item for Nik: prepare a suggestion for a database connection by 15.01.2009. Action item end.
- New issue: Problems with the network connection. Issue End.
- New info: we will have a strategy meeting in September. Info end.
- New decision: our annual team excursion will be a bike tour this year. Decision end.
- New Backlog Item: progress bar for data entries. Backlog Item end.

Appendix II: Prototype

Implementation of the STACHUS framework

This chapter presents a brief description of the realization of two prototypes: one for formal meetings in a meeting room and one iPod/iPhone-application for informal and ad-hoc meetings.

A.II.1 Realization of a prototype for formal meetings

The STACHUS Protocol Engine was realized in a prototype to evaluate the feasibility of an automated protocol generator and validate our hypotheses (see chapter 6.1). The results of the evaluation are presented in chapter 6.

The framework for protocol generation is modularly designed, i.e. the speech recognition engines, as well as the generated reports and documents, and the linked project and task management tools respectively, can be exchanged at runtime. The STACHUS framework was exemplarily realized supporting Excel file outputs, XML and text files, as especially Excel is still one of the most common project management tools (VersionOne, 2009). Additionally, to support the distinct character of a project, individual project management tools are interlinked with the STACHUS framework. The first prototype implements Nuance's speech recognition engine Dragon NaturallySpeaking in combination with manual transcriptions. The prototype is constructed for agile Scrum meetings and common status meetings, i.e. the STACHUS Analyzer and grammar are aligned for those meetings.

A.II.1.1 Libraries and Packages

The prototype is written in Java 1.5. For the creation of the different file formats existing libraries where integrated, which will be briefly described in the following.

Apache POI - the Java API for Microsoft Documents (The Apache Software Foundation, 2002)

Apache's POI project enables the creation and maintenance of Java APIs for manipulating various file formats based upon the Office Open XML standards (OOXML) and Microsoft's OLE 2 Compound Document format (OLE2). OLE2 files include most Microsoft Office files such as xls, doc, and ppt. For each MS Office application exists a component module that provides a common high level Java API, including the Java implementation of Excel files that is applied in our prototype. Thus functionality is provided to create, modify, read and write xls spreadsheets, so new protocols can be generated in Excel and existing worksheets be modified, to continue existing protocols.

The Java API for XML Processing (*JAXP*) enables applications to parse, transform, validate and query XML documents using an API that is independent of a particular XML processor implementation. It provides the capability of validating and parsing XML documents.

In the STACHUS framework following packages are reused:

javax.xml.parsers	provides classes allowing the processing of XML documents.
javax.xml.transform	defines the generic APIs for processing transformation instructions, and performing a transformation from source to result.

A.II.1.2 Interfaces

The STACHUS framework allocates two interfaces, one to the speech recognition engine and one to project and task management tools, which discussed in the following.

The speech recognition engine is integrated by the STACHUS Protocol Generator Compiler during the framework generation process in the STACHUS Protocol Engine. So the speech recognition engine can easily be exchanged with a new version, or additional engines can be integrated to parallelize the speech recognition process, when the STACHUS Protocol Engine is updated and new or additional knowledge sources are integrated. Hence, the STACHUS framework is constructed in such a way, that it is independent from the individual speech recognition engine, as the recognized phrases are published on the blackboard.

The second interface of the STACHUS Engine is the connection to tools for project, task and document management. Information, extracted in the meeting can be handed in several ways, that is, if a new task for instance was identified and assigned in a meeting, it can be documented in a text protocol, added to an individual task list, or send and included in a team specific project management tool, like XPlanner (Codehaus Foundation, 2006) or VersionOne (VersionOne, Inc, 2010) for agile teams. The user can decide in advance of each meeting, if the information will be transferred to a tool – realized with a strategy. To show feasibility, our first prototype provides interfaces to following tools: VersionOne, XPlanner, and Unicase.

VersionOne is one of the leading project planning and management tool designed specifically for agile software development, enabling the agile methodologies Scrum, XP, DSDM and Agile UP. The tool is relied on by over 10,000 teams around the world and has more than 70,000 users (VersionOne, Inc, 2010).

Our prototype provides a connection to the VersionOne Integration Platform, an open-source toolkit for building applications that integrate with the VersionOne Application. An external application or integration can interact with the VersionOne server using the Core API (VersionOne API), which is the foundation for all integrations. This API, available on any VersionOne instance, provides secure read/write access to all data stored in the VersionOne system. Through the Data API – part of the Core API – it is possible to query for simple or complex sets of information, update the information, and execute system-defined operations. Additionally, several sample applications are provided by VersionOne, for example the V1 Task Manager. This is a task tray utility that allows viewing and updating key properties of tasks, tests, defects and stories. The utility also allows taking ownership of items as well as taking effort and closing items. Another helpful sample application for the STACHUS framework it the Create Entity sample that is able to create Stories, Defects, Themes, Goals, Requests, and Issues in a specific project. With it, connection and data transfer between our framework and the VersionOne tool are realized, e.g. the creation of new issues, a status update, or assignment of a task, discussed in a meeting and handled in VersionOne.

XPlanner (Codehaus Foundation, 2006) is a web-based project planning and tracking tool for agile development teams in particular for eXtreme Programming (XP). XPlanner is an open-source project, implemented using Java, JSP, and Struts, Hibernate and MySQL supporting integration in the STACHUS framework with directly manipulating the data on the data base. The main features of XPlanner are a simple planning model, virtual note cards, user stories, and tasks. Additionally metrics can be generated (team velocity, individual hours, etc.) and charts for iteration velocity (like Scrum Burndown Charts).

The *UNICASE* system integrates models from different development activities, such as requirements, use cases, UML models, schedules, bugs, and feature models into a unified model. The UNICASE client allows viewing and editing these models in a textual, tabular and diagramming visualization. The models are stored and versioned on a server. Client and server are easily extensible to support integrating new models into the unified model. The open-source project UNICASE is based on the Eclipse platform including the Eclipse Modeling Framework (EMF) and Graphical Modeling Framework (GMF). Its modular and open interfaces allow a connection establishment out of the STACHUS framework and make the creation and modification of action items, issues or bugs (discussed in a meeting) possible, as well as the storage of meeting minutes, for instance.

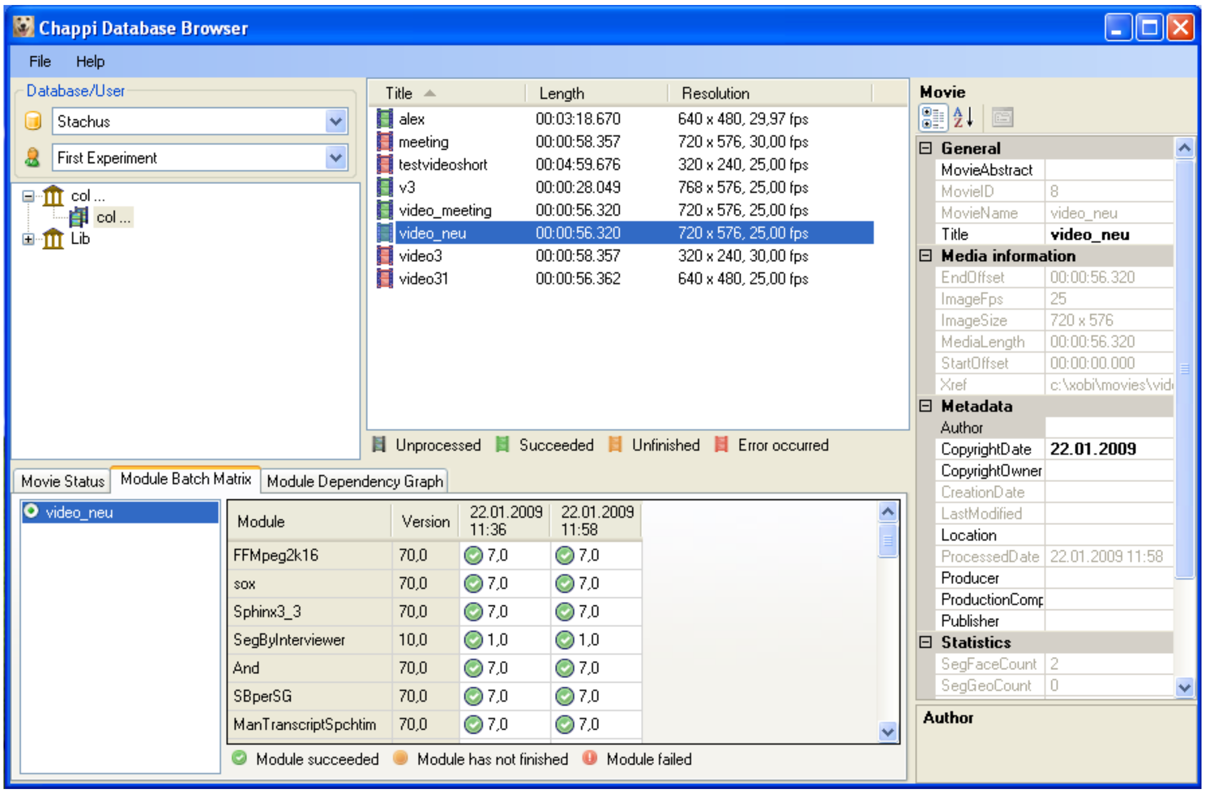
In future, the STACHUS framework will be extended supporting further project management tools (like MS Project (Microsoft, 2010), Scrumworks (CollabNet, Inc. , 2010), etc.) or additional document formats (e.g. .doc).

A.II.1.3 Linking audio and video

The first version of the STACHUS prototype used only the audio stream of a meeting. However, if a video is recorded of the meeting, the visual information can be reused, too. This is done for example by linking the location in the audio with the information in the protocol. If somebody couldn't attend a meeting and now is reading the meeting minutes, he can jump to the situation, where an action item for instance was assigned to him, by following a link from the meeting minutes to the position in the video. So he can watch the part of the video that is important to him, without loosing time while searching through the whole video. The feasibility of these ideas is analyzed within our prototype with the Informedia Project, introduced in chapter 4.

The overarching goal of the Informedia initiatives is to achieve machine understanding of video and film media, including all aspects of search, retrieval, visualization and summarization in both contemporaneous and archival content collections. The developed technology combines speech, image and natural language understanding to automatically transcribe, segment and index linear video for intelligent search and image retrieval. Utilizing this system, indexing and retrieval of key words in video files is integrated in the STACHUS prototype. All information of the Informedia project is written to an (Oracle) database, which allows reading out the needed information and reusing them during the report generation.

Following the procedure of integrating meetings in the Informedia system is briefly described. First a new meeting video is imported into the Informedia system, followed by the execution of this video (speech recognition, segmentation by speaker, indexing, etc.). This step is illustrated in Figure A.II.1 and Figure A.II.2.



Title	Length	Resolution
alex	00:03:18.670	640 x 480, 29,97 fps
meeting	00:00:58.357	720 x 576, 30,00 fps
testvideoshort	00:04:59.676	320 x 240, 25,00 fps
v3	00:00:28.049	768 x 576, 25,00 fps
video_meeting	00:00:56.320	720 x 576, 25,00 fps
video_neu	00:00:56.320	720 x 576, 25,00 fps
video3	00:00:58.357	320 x 240, 25,00 fps
video31	00:00:56.362	640 x 480, 25,00 fps

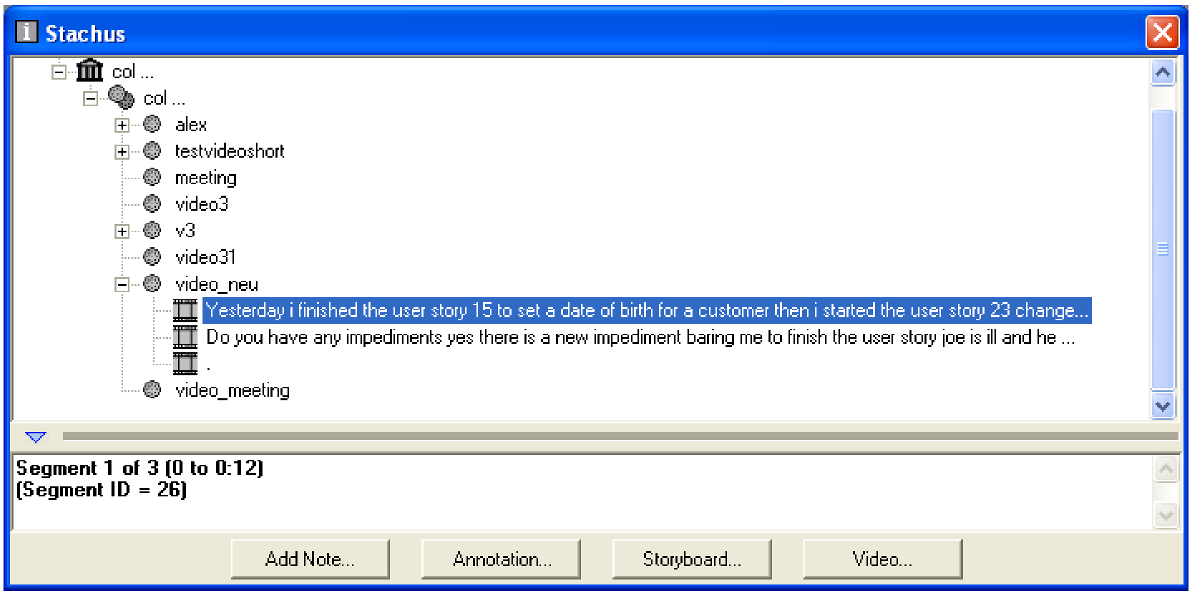
🟩 Unprocessed 🟩 Succeeded 🟧 Unfinished 🟥 Error occurred

General	
MovieAbstract	
MovieID	8
MovieName	video_neu
Title	**video_neu**
Media information	
EndOffset	00:00:56.320
ImageFps	25
ImageSize	720 x 576
MediaLength	00:00:56.320
StartOffset	00:00:00.000
Xref	c:\xobi\movies\vid
Metadata	
Author	
CopyrightDate	**22.01.2009**
CopyrightOwner	
CreationDate	
LastModified	
Location	
ProcessedDate	22.01.2009 11:58
Producer	
ProductionComp	
Publisher	
Statistics	
SegFaceCount	2
SegGeoCount	0

Module	Version	22.01.2009 11:36	22.01.2009 11:58
FFMpeg2k16	70,0	✓ 7,0	✓ 7,0
sox	70,0	✓ 7,0	✓ 7,0
Sphinx3_3	70,0	✓ 7,0	✓ 7,0
SegByInterviewer	10,0	✓ 1,0	✓ 1,0
And	70,0	✓ 7,0	✓ 7,0
SBperSG	70,0	✓ 7,0	✓ 7,0
ManTranscriptSpchtim	70,0	✓ 7,0	✓ 7,0

✓ Module succeeded ● Module has not finished ● Module failed

Figure A.II.1: Informedia: Database Browser (import and execution of a new meeting video)



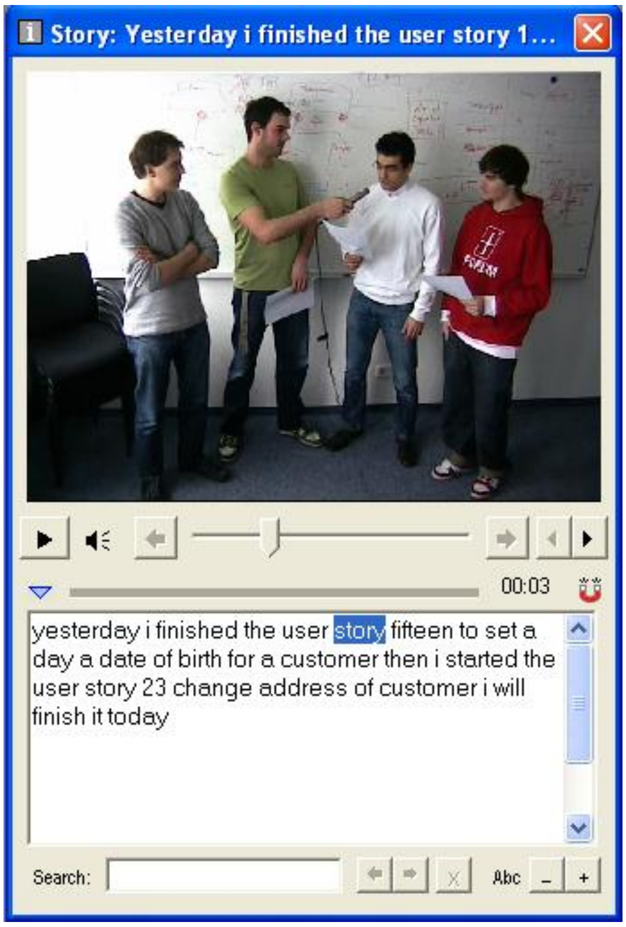
```
col ...
  col ...
    alex
    testvideoshort
    meeting
    video3
    v3
    video31
    video_neu
      Yesterday i finished the user story 15 to set a date of birth for a customer then i started the user story 23 change...
      Do you have any impediments yes there is a new impediment baring me to finish the user story joe is ill and he ...
      .
    video_meeting
```

Segment 1 of 3 (0 to 0:12)
(Segment ID = 26)

Figure A.II.2: Result of the Informedia batch process

The result of this procedure, are listed in a meeting browser, as illustrated in Figure A.II.3, – video sequences of meeting sections separated by speaker and the corresponding text.

Behind each section are its details with the video and corresponding text sequence. As a video player is integrated, the video can be replayed and additionally the currently spoken text is highlighted. Thus, linking the indexed text in the Informedia project with the created or updated action item for instance in the generated report would allow a meeting participant to review the details in a follow up process.



yesterday i finished the user story fifteen to set a day a date of birth for a customer then i started the user story 23 change address of customer i will finish it today

Figure A.II.3: Informedia: Player for meeting section with text highlighting

A.II.2. Prototype: mobile version

During the iPhone practical course in the summer term 2009, a mobile version of the first prototype, the iPhone App “iMOnTrack”, was developed by five students. This ad-hoc conferencing tool was designed and implemented in cooperation with Siemens Enterprise Communications (SEN). Following the realized functionality is described and screenshot of the final prototype are presented. Details of the prototype and its realization can be found on the project website (SEN-iPhone Team, 2009).

Requirement of SEN was that a meeting participant – the user – can start a meeting, pause and stop it with the iPhone app and send the audio to our server, which implements the STACHUS Protocol Engine.

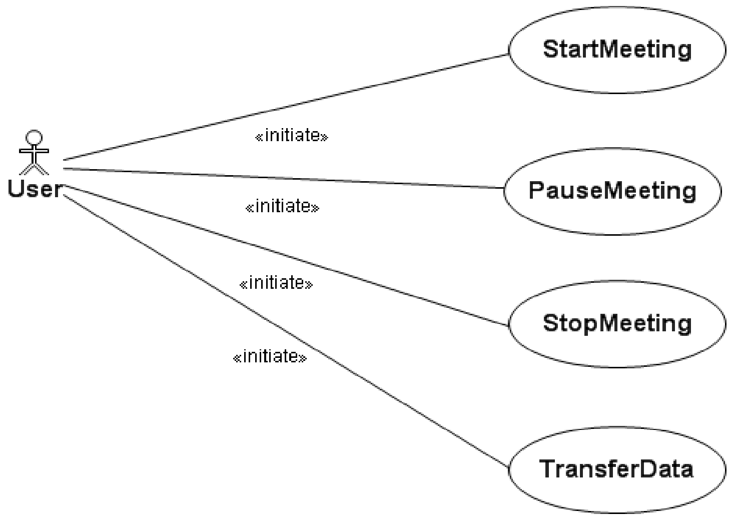


Figure A.II.4: iPhone app – meeting recording (UML Use Case Diagram)

In addition, the user should be able to attach documents to the meeting, regard these attachments and replay the meeting audio. Moreover, a meeting summary should be provided to the user, containing assigned action items, decisions, issues, or new information. Furthermore, it should be possible to search for meetings, share and delete them, and edit the meeting details, like participants, or the description, as summarized in Figure A.II.5.

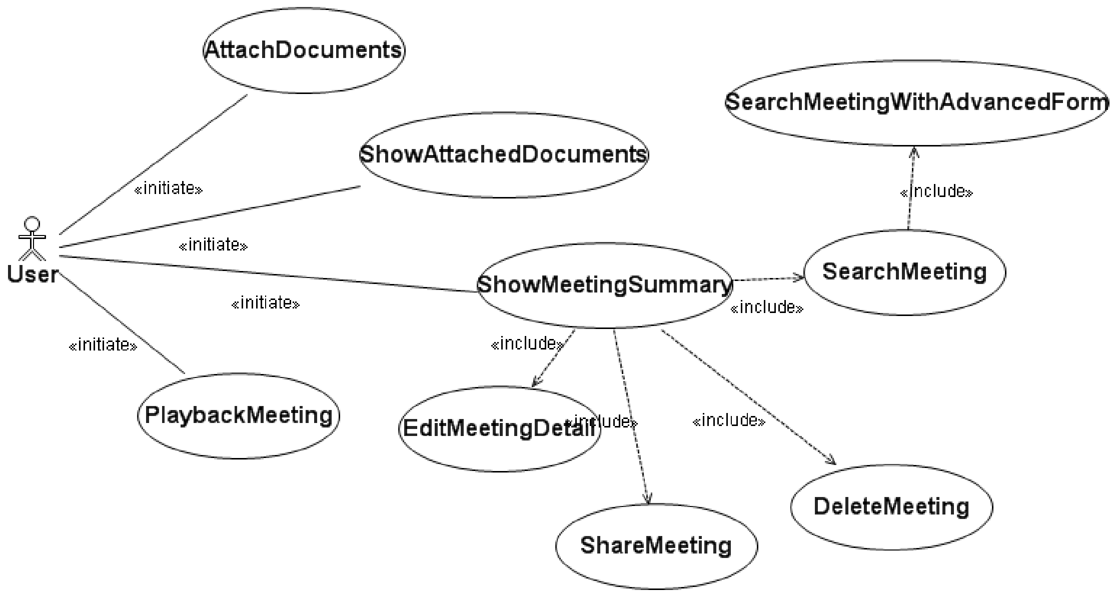


Figure A.II.5: iPhone app - additional functionality

The iPod/Phone prototype implements the STACHUS Audio Recorder and a protocol viewer. The STACHUS Protocol Compiler, as well as STACHUS Analyzer and STACHUS Protocol Generator are on a backend server (called Speech Recognition Server in Figure A.II.6). That is, components of the first prototype are reused on the backend server that analyzes the audio stream, extracts meeting elements like action items and prepares them as a protocol, which can be shown on the iPhone. iPhone app and the backend server communicate via the iMOnTrack Application Server.

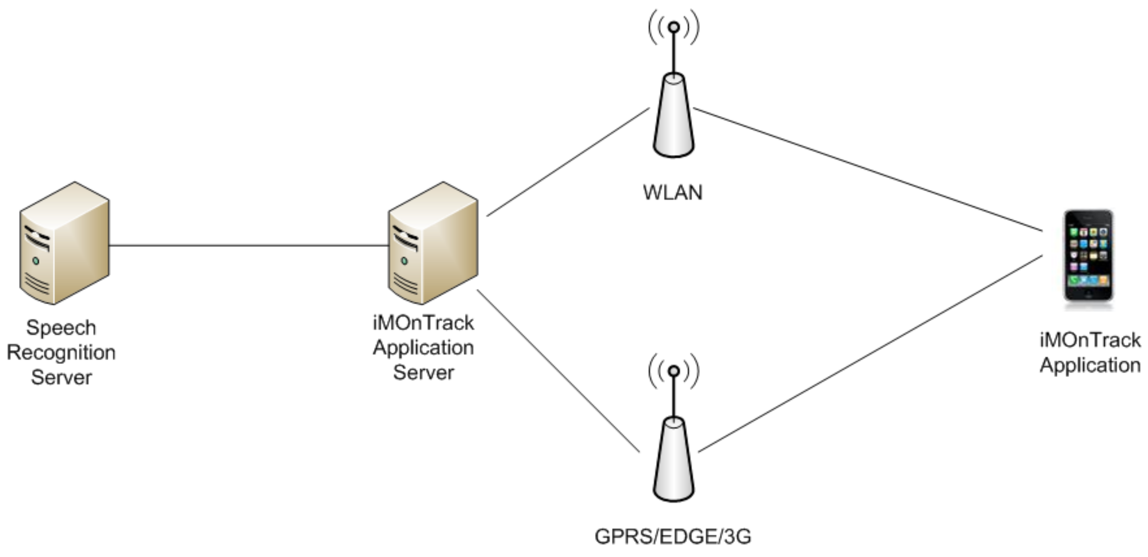


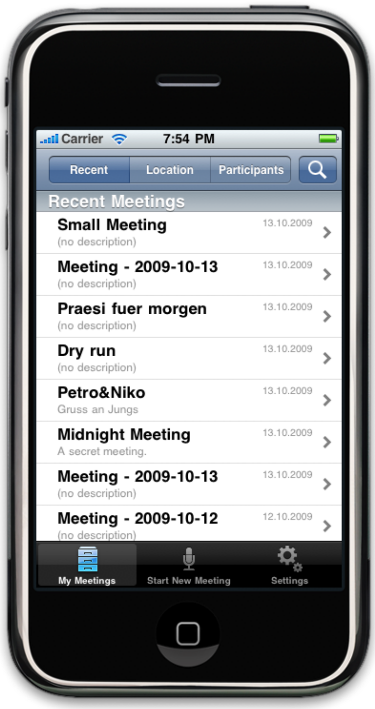
Figure A.II.6: overview of iPhone and server connection

The result of the development process is illustrated in the following screenshots of the iMOnTrack iPhone application (Figure A.II.7 - Figure A.II.16).

All recorded meetings are listed under "My Meetings", chronological, or separated by participant or location. Additionally, meetings can be search for. "Start New Meeting" allows the user to record a new meeting, attach documents and administrate the participants. Under "Settings" the user can change the connected application server, as well as user name and password.

If a recorded meeting is selected from the list (or map) its details are shown, including the participants, the meeting summary, and details of the meeting like the duration. Moreover, the selected meeting audio can be replayed.

The iPhone app was integrated in the weekly meeting workflow of the students and thus evaluated already during the development process by the team members. The evaluation is described in detail in chapter 6.



Carrier 7:54 PM

Recent | Location | Participants

Recent Meetings

Small Meeting 13.10.2009
(no description)

Meeting - 2009-10-13 13.10.2009
(no description)

Praesi fuer morgen 13.10.2009
(no description)

Dry run 13.10.2009
(no description)

Petro&Niko 13.10.2009
Gruss an Jungs

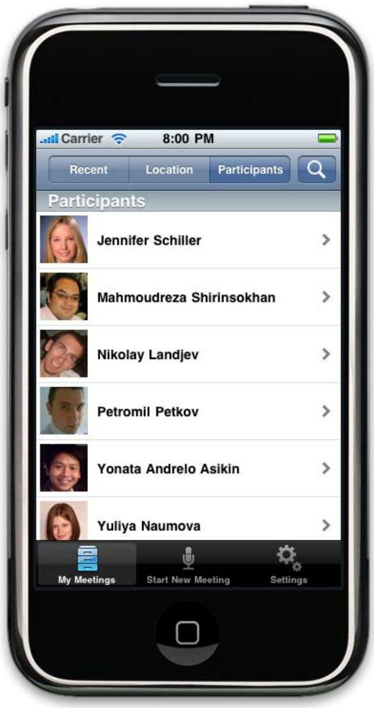
Midnight Meeting 13.10.2009
A secret meeting.

Meeting - 2009-10-13 13.10.2009
(no description)

Meeting - 2009-10-12 12.10.2009
(no description)

My Meetings | Start New Meeting | Settings

Figure A.II.7: Meeting list



Carrier 8:00 PM

Recent | Location | Participants

Participants

Jennifer Schiller

Mahmoudreza Shirinsokhan

Nikolay Landjev

Petromil Petkov

Yonata Andrelo Asikin

Yuliya Naumova

My Meetings | Start New Meeting | Settings

Figure A.II.8: Meetings by participant

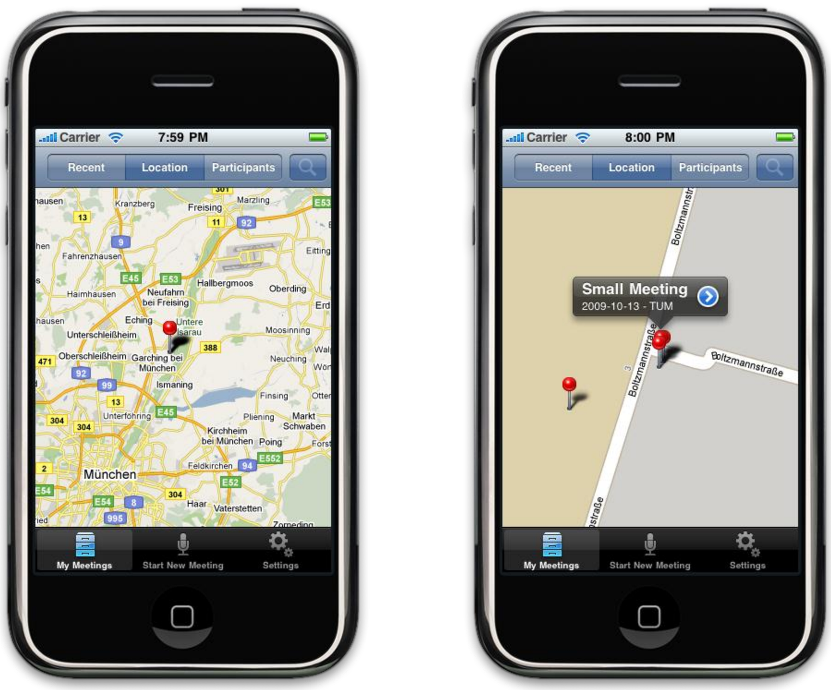
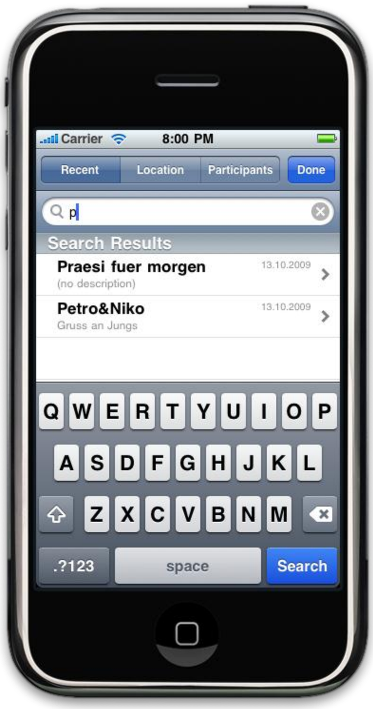
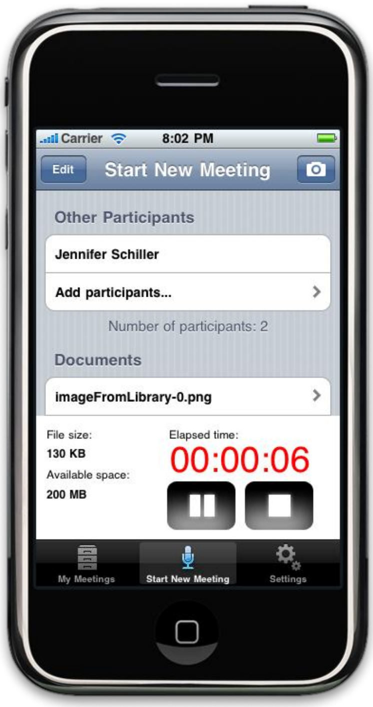


Figure A.II.9: Meetings by location



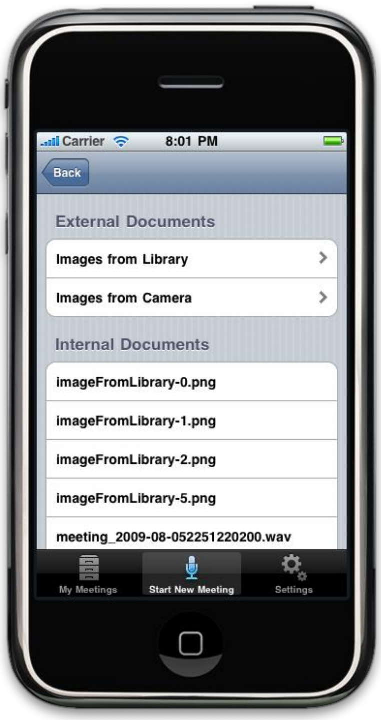
Praesi fuer morgen 13.10.2009
(no description)
Petro&Niko 13.10.2009
Gruss an Jungs

Figure A.II.10: search for meetings



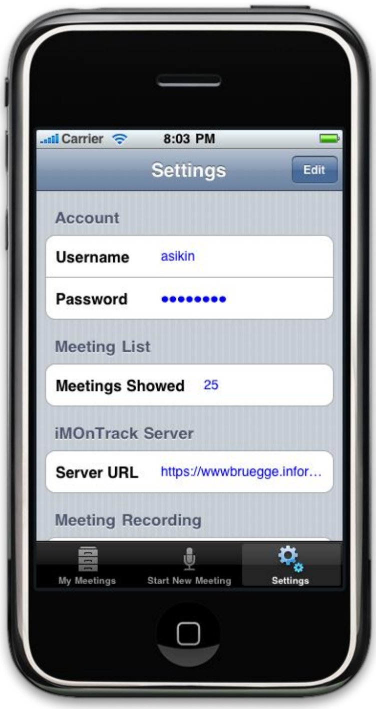
Other Participants
Jennifer Schiller
Add participants...
Number of participants: 2
Documents
imageFromLibrary-0.png
File size: 130 KB
Available space: 200 MB
Elapsed time: 00:00:06

Figure A.II.11: meeting recording



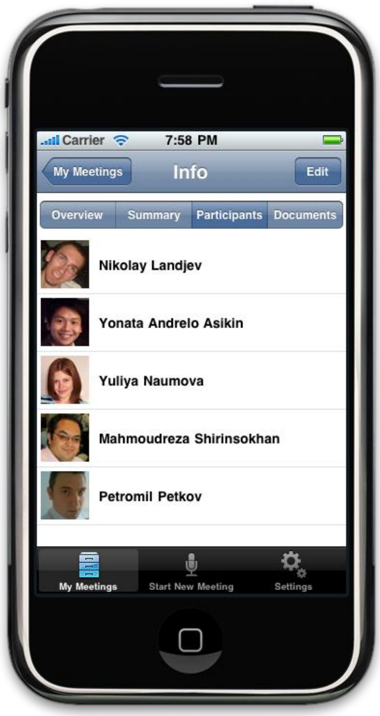
External Documents
Images from Library
Images from Camera
Internal Documents
imageFromLibrary-0.png
imageFromLibrary-1.png
imageFromLibrary-2.png
imageFromLibrary-5.png
meeting_2009-08-052251220200.wav

Figure A.II.12: Attach documents to meeting



Account
Username asikin
Password ••••••••
Meeting List
Meetings Showed 25
iMOnTrack Server
Server URL https://wwwbruegge.infor...
Meeting Recording

Figure A.II.133: Settings dialog



Nikolay Landjev

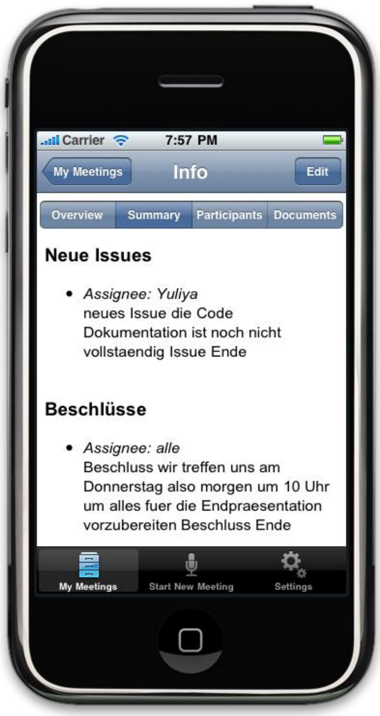
Yonata Andrelo Asikin

Yuliya Naumova

Mahmoudreza Shirinsokhan

Petromil Petkov

Figure A.II.14: Participants of the meeting



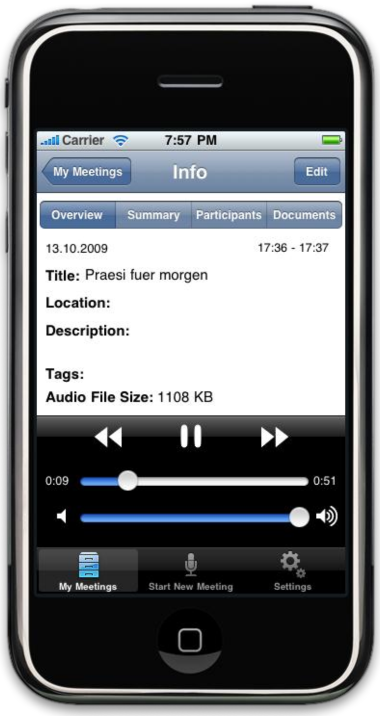
Neue Issues

- *Assignee: Yuliya*
  neues Issue die Code Dokumentation ist noch nicht vollstaendig Issue Ende

Beschlüsse

- *Assignee: alle*
  Beschluss wir treffen uns am Donnerstag also morgen um 10 Uhr um alles fuer die Endpraesentation vorzubereiten Beschluss Ende

Figure A.II.15: Meeting summary



13.10.2009 17:36 - 17:37

Title: Praesi fuer morgen

Location:

Description:

Tags:

Audio File Size: 1108 KB

Figure A.II.16: Meeting details and integrated Player

Appendix III: Empirical research

Empirical research methods

This chapter gives a brief overview of the empirical research methods, applied during the evaluation of our hypotheses (see chapter 6).

Empirical methods focus on collection data and interpreting conclusions out of it. Following, the methods for observation and measurment and describing and analyszing the gained data, which were applied during our case studies, are described.

Empirical research methods can be classified into qualitative and quantitative methods. Quantitative research means that the data exist in a numerical or graphic form, and qualitative research means that the data exist in a narrative form (spoken words, recorded conversations) or a pictorial form. (Rosnow & Rosenthal, 2008, p.75) This distinction is not unambiguous, however, because it is always possible to figure out ways to enumerate and graph aspects of qualitative data (e.g. using judges as counters of events, or having a computer systematically decompose written messages). The two classes are not mutually exclusive, as it is possible to use quantitative and qualitative methods in the same study (e.g. interviewing some participants in a rigorously quantified lab experiment).

Qualitative research is used to understand, study, and explain social and cultural phenomena. It involves the use of qualitative data sources such as surveys, interviews and questionnaires, the observation of participant, and the researcher's impressions and reactions. Qualitative data is any non-numerical information, represented as words and pictures. Qualitative analysis methods are designed to analyze qualitative data. These methods tend to be used when it is necessary to evaluate and understand end user perspectives of a situation; they allow the identification of human-related aspects such as motivation, thinking, attitudes, values, and satisfaction with a product.

Quantitative research methods work on the measurable properties and employ mathematical models and theories for the investigation of hypothesis. Examples of quantitative methods include survey methods, laboratory experiments, formal methods and numerical methods such as mathematical modeling. They produce statistical results by counting features, activities, and measuring certain values of interest.

Quantitative methods are best suited to statistically evaluate a hypothesis that can be translated to a quantifiable value. For example: "teams using protocol automation receive the meeting minutes faster than those teams not using it". The time to create a protocol and how long it takes until it is published can be measured, giving the possibility to support or disprove the hypothesis.

Techniques to *collect data* include among others interviews, observational techniques such as participant observation, and questionnaires.

Interviewing is a qualitative data query technique that asks questions to the interviewee. The questions are used to collect opinions or impressions about the activities. They are sometimes used in combination with participant observations where they serve to clarify things that happened or where said during an observation, for example to elicit impressions of a meeting or to collect information on relevant events that were not observed.

Participant observation refers to a technique, where a group or community is studied from within by recording behavior as it occurs. The data is systematically and unobtrusively collected. This does not mean that the observer takes part in the activities. It means only that the observer is visibly present and is collecting data. During a *judgment study*, an observer (judge) scales, sorts, or rates certain variables (e.g. observable behavior). As the observation techniques alone often are of limited use, they should be complemented with other qualitative techniques such as think aloud protocols and field notes.

In order to perform some type of quantitative or statistical analysis, the qualitative data can be quantified. A commonly used technique called coding extracts values for quantitative variables from qualitative data (collected from observations or interviews).

A *questionnaire* is a research instrument consisting of a series of questions to gather information from respondents. The questions can be distinguished between open-ended and closed-ended questions. An open-ended question asks the respondent to formulate his own answer, whereas in a closed-ended question the respondent picks an answer from a given number of options. The response options for a closed-ended question should be exhaustive and mutually exclusive. The parameters of a hypothesis can be measured by posing closed questions. However, as not all of the measureable parameters have concrete measurement standards, the response choices are modeled using a fictitious scale, for instance with the response choices *very useful*, *fairly useful*, *useful*, *less useless*, *useless*, *don't know*. For evaluation, the response choices of the research questions are coded using natural numbers. Coding starts from the lowest response choice, that is, "useless" is coded with 1. The other response choices are coded in ascending order with 2, 3, 4, and 5 for "very useful".

The collected data is analyzed and evaluated by *statistical tests*. Beside describing data and measuring relationships, researchers are usually interested in making comparisons using statistical test, such as t-tests.

Null hypothesis significance testing (NHST) uses statistics and probabilities to evaluate null hypotheses and determine the significance level α and probability (p value), to evaluate if for example a difference between two means might be due to chance (Rosnow & Rosenthal, 2008, p. 271). In this context a Type I error (significance level) implies that the decision maker mistakenly rejected the null hypothesis (H0) when it is, in fact, true and should not have been rejected.

A t-test is a test of statistical significance that examines the difference between two independent means against the background of the within-group variability. The larger the difference between the means, and/ or the smaller the within-group variability for any given size of study, the greater will be the value of t. Because large t values are associated with differences between means that are more statistically significant, researchers generally prefer larger t values. That is, larger t values have a lower level of probability (the p value or alpha) and, in turn, allow researchers to reject the null hypothesis that there is no difference between the means. When the results are not statistical significant, the evidence is termed to be anecdotal evidence.

The .05 alpha, indicating the statistical significance, is seen by many scientists as a good “fail-safe” standard because it is convenient (most statistical tables show 5% values) and stringent enough to protect us from too often concluding that the null hypothesis is false when it is actually true.

Appendix IV: Questionnaire

Questions and response choices

1) Gender:

   - o female
   - o male

2) How old are you?

   ____

3) What is your profession?

   - o Software Developer
   - o Product Manager
   - o Project Manager
   - o Consultant
   - o Student
   - o Other: ________________________

4a) For how many years do you already work in your profession?

   - o < 1 year
   - o 1 - 3 years
   - o 4 - 5 years
   - o 6 - 10 years
   - o More than 10 years

4b) How many semesters have you already completed in your current course of study?

- o 1 - 2 semesters
- o 3 - 4 semesters
- o 5 - 6 semesters
- o 7 - 8 semesters
- o 9 - 10 semesters
- o > 11 semesters

5) Do you have any experience with agile methods (Scrum, XP, FDD, DSDM, …)?

- o Yes, applied
- o Yes, but only read/ heard
- o No

6) How many hours do you work on average per week?

____

7a) On how many meetings do you participate on average per week?

	Number
up to 30 minutes	
30 minutes - 1 h	
1 - 2 h	
2 - 4 h	
4 - 8 h	

7b) How many informal (unplanned) meetings do you have approximately per week?

- o < 3 meetings
- o 3 - 5 meetings
- o 6 - 10 meetings
- o 11 - 15 meetings
- o 15 - 20 meetings
- o > 20 informal meetings per week

8) How many time do you spend approximately for writing protocols?

Meeting duration	<= 10 min	30 min	45 min	1 h	1,5 h	>2 h	not specified
up to 30 min	O	O	O	O	O	O	O
30 min - 1 h	O	O	O	O	O	O	O
1 - 2 h	O	O	O	O	O	O	O
2 - 4 h	O	O	O	O	O	O	O
4 - 8 h	O	O	O	O	O	O	O

9) How many time do you spend approximately for post-processing of meetings, beside writing a protocol?

Meeting duration	<= 10 min	30 min	45 min	1 h	1,5 h	>2 h	not specified
up to 30 min	O	O	O	O	O	O	O
30 min - 1 h	O	O	O	O	O	O	O
1 - 2 h	O	O	O	O	O	O	O
2 - 4 h	O	O	O	O	O	O	O
4 - 8 h	O	O	O	O	O	O	O

10) How long does it approximately take, until a protocol is finished and published by the protocol writer?

- o 1 hour
- o Half a day
- o 1 day
- o Up to 3 days
- o 1 week
- o 2 weeks
- o 1 month

11) When are the protocols usually published?

- o Early
- o In time
- o Just before the next meeting
- o Too late

12) Did you ever get annoyed at incomplete or incorrect protocols?

- o Often
- o Sometimes
- o Seldom
- o Never

13) In which form do you write and publish your protocols?

- ☐ In Powerpoint
- ☐ In Word
- ☐ In Excel
- ☐ Per e-mail
- ☐ Other: ________________________

14) How do you manage your tasks?

- ☐ Not at all
- ☐ On paper
- ☐ Electronically, with the following tool: ________________________

15) How helpful is the automation of the protocol generation process for you?

- o Very helpful
- o Fairly helpful
- o Helpful
- o Less helpful
- o Useless
- o Don't know

16) How useful is the iPod application for you to remember the results also from short meetings?

- o Very useful
- o Fairly useful
- o Useful
- o Less useless
- o Useless
- o Don't know

17) Do you believe that the automation of the protocol generation would save time in your project workaday life?

- o Yes, definitely
- o Yes, probably
- o No
- o No, even additional work
- o Don’t know

18) Do you believe that, due to the automation of the protocol generation, there are less errors in the protocols?

- o Yes, much less
- o Yes, less
- o No
- o No, even more
- o Don’t know

19) Do you believe that the automation of the protocol generation and integration with your tools (e.g. with Excel, To-Do-Lists) would be useful and an assistance for you?

- o Yes, definitely
- o Yes, probably
- o No
- o No, even additional work
- o Don’t know

20) Is the iPod-application easy to handle?

- o Yes
- o Fair
- o No
- o Don’t know

21) Do you believe that the protocol automation is helpful for agile projects in a traditional environment, especially regarding knowledge management?

- o Yes
- o Partly
- o No
- o Don't know

22) How useful is the iPod-application for you to cope with the information overload from meetings?

- o Very useful
- o Fairly useful
- o Useful
- o Less useless
- o Useless
- o Don't know

23a) Would it have any effects on the communication behavior, if the conversation of a meeting is recorded?

- o Yes, very much
- o Yes, a little bit
- o No

23b) If yes, in what way?

____

24a) Would you have any concerns, if meetings are recorded?

- o Yes
- o No

24b) If yes, to what extent?

____

25) Would you have any concerns, if meetings are recorded, but deleted after the processing?

- o Yes
- o No

26) Would you have any concerns, if meetings are recorded and project-internally stored (accessible only for meeting participants)?

- o Yes
- o No

27) How difficult is the grammar to learn and abide by its rules?

- o Very easy
- o Easy
- o Difficult
- o Very difficult
- o Don't know

28) What are your requirements for a system that automates the protocol generation?

____

29) What do you like regarding the idea of meeting support and automatic protocol generation?

____

30) What don't you like regarding the system?

____

31) What should the system provide additionally to be even more useful for you?

____

Bibliography

Agile Alliance. (2010). *Agile Alliance*. Retrieved 03 16, 2010, from http://agilealliance.com/

agilecollab. (2008, 01 04). *Agile Introduction for Dummies*. Retrieved 03 17, 2010, from WATERFALL vs. AGILE METHODOLOGY: http://agileintro.wordpress.com/2008/01/04/waterfall-vs-agile-methodology/

Alavi, M., & Leidner, D. E. (2001, March). Review: Knowledge Mamagement and Knowledge Management Systems: Conceptual Foundations and Research Issues. *MIS Quarterly , Vol. 25* (No. 1), pp. pp. 107-136.

Alexandersson, J., & Poller, P. (1998). Towards multilingual protocol generation for spontaneous speech dialogues. *In Proceedings of the INLG'98*, (pp. 198-207). Niagara-on-the-lake.

amazon.com. (2005, 11 02). *Amazons Mechanical Turk*. Retrieved 01 25, 2010, from https://www.mturk.com/mturk/welcome

Ambler, S. W. (2006). *Agile Best Practice: Document Late*. Retrieved 04 20, 2010, from http://www.agilemodeling.com/essays/documentLate.htm

Ambler, S. W. (2006). *Agile/Lean Documentation: Strategies for Agile Software Development*. Retrieved 04 20, 2010, from http://www.agilemodeling.com/essays/agileDocumentation.htm

Ambler, S. W. (2009). *Communication on Agile Software Projects*. Retrieved 04 07, 2010, from http://www.agilemodeling.com/essays/communication.htm

Ambler, S. W. (2006). *Survey Says: Agile Works in Practice.* Retrieved 04 15, 2010, from http://www.it-smc.com/Articles/Survey%20Says%20-%20Agile%20Works%20in%20Practice.pdf

Ambler, S. W. (2009). *Why Agile Software Development Techniques Work: Improved Feedback*. Retrieved 04 15, 2010, from http://www.ambysoft.com/essays/whyAgileWorksFeedback.html

Aone, C., Okurowski, M. E., Gorlinsky, J., & Larsen, B. (1999). A trainable summarizer with knowledge acquired from robust nlp techniques. *Advances in Automatic Text Summarization* (pp. 71-80). MIT Press.

Arons, B. (1994). Pitch-based emphasis detection for segmenting speech. *In Proceedings of the ICSLP'94*, (pp. 1931-1934).

Arons, B. (1997, March). SpeechSkimmer: A system for interactively skimming recorded speech. *ACM Transactions on Computer Human Interaction , 1*, pp. 3-38.

Barzilay, R., & Elhadad, M. (1997). Using lexical chains for text summarization. *In Proceedings ISTS'97*, (pp. 10-17).

Baxendale, P. (1958). Machine-made index for technical literature - an experiment. *IBM Journal of Research Development , 2* (4), pp. 354-361.

Beck, K., & al., e. (2001). *Manifesto for Agile Software Development*. Retrieved 11 27, 2009, from http://agilemanifesto.org/

Beck, K., & Andres, C. (2004). *Extreme Programming Explained: Embrache Change, 2nd Edition.* Amsterdam: Addison-Wesley Longman.

Berteig, M. (2006, 09 19). *Agile Advice*. Retrieved 04 15, 2010, from Agile Challenges - A Good List of the Common Problems with Agile Methods: http://www.agileadvice.com/archives/2006/09/agile_challenge.html

Bleek, W.-G., & Wolf, H. (2008). *Agile Softwareentwicklung: Werte, Konzepte und Methoden.* Heidelberg: Dpunkt Verlag.

Bortz, J., & Döring, N. (2005). *Forschungsmethoden und Evaluation* (3. Auflage Ausg.). Heidelberg: Springer Medizin Verlag.

Bridle, J., & Brown, M. (1979). Connected Word Recognition Using Whole Word Templates. *Proc. Inst. Acoust. Autumn Conf.*, (pp. 25-28).

Brooks, F. P. (1987). No Silver Bullet Essence and Accidents of Software Engineering. *Computer , no. 4*, pp. 10-19.

Brown, E. W., Srinivasan, S., Coden, A., Ponceleon, D., Cooper, J. W., & Amir, A. (2001). Toward speech as a knowledge resource. *IMB Systems Journal* , pp. 985 - 1001.

Brown, F., Pietra, V., Pietra, S., & Mercer, R. (1993). The mathematics of statstical machine translation: parameter estimation. *Comput. Linguist. , 19* (2), pp. 263-311.

Bruegge, B., Reiss, M., & Schiller, J. (2009). Agile Principles in Academic Education: A Case Study. *Sixth International Conference on Information Technology: New Generations,* (pp. 1684-1686). Las Vegas, Nevada: itng.

Brügge, B., & Dutoit, A. H. (2004). *Objektorientierte Softwaretechnik mit UML, Entwurfsmustern und Java.* München: Person Education Deutschland.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., et al. (2005). Learning to rank using gradient descent. *In ICML'05: Proceedings of the 22nd international conference on Machine learning* (pp. 89-96). New York, NY: ACM.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1998). *Pattern-orientierte Software-Architektur.* Addison-Wesley.

Canditt, S., & Russwurm, W. (2008). *The First CMMI-based Appraisal in an Agile Environment at Siemens AG.* Retrieved 11 30, 2009, from http://www.sei.cmu.edu/library/assets/Canditt08.pdf

Carbonell, J., Geng, Y., & Goldstein, J. (1997). Automated query-relevant summarization and diversity-based reranking. *In Proceedings of the IJCAI'97 workshop on AI and digital libraries*, (pp. 9-14). Nagoya.

Carnegie Mellon University. (1991). *Capability Maturity Model Integration (CMMI).* Retrieved 11 30, 2009, from http://www.sei.cmu.edu/cmmi/

Chang, S., Chen, W., Meng, H., Sundaram, H., & Zhong, D. (1997). VideoQ: An Automated Content Based Video Search System Using Visual Cues. *Multimedia* (pp. 313-324). Seattle, WA: Proceedings, ACM.

Charette, R. (2002). Foundations of Lean Development: The Lean Development Manager's Guide. *The Foundations Series on Risk Management* .

Chen, F. R., & Withgott, M. (1992). The use of emphasis to automatically summarize a spoken disourse. *In Proceedings of the ICASSP'92*, (pp. 229-232). San Francisco, CA.

Chow, Y., Dunham, M., Kimball, O., Krasner, M., Kubala, G., Makhoul, J., et al. (1987). BBYLOS: The BBN Continuous Speech Recognition System. *Proc. ICASSP 87*, (pp. 89-92).

Christel, M., Stevens, S., & Wactlar, H. (1994). Informedia Digital Video Library. *In Proceedings of the Second ACM International Conference on Multimedia* (pp. 480-481). New York: ACM.

Chung, M.-W., & Drummond, B. (2009). Agile at Yahoo! From the Trenches. *2009 Agile Conference* (pp. 113-118). IEEE Computer Society.

CMMI Product Team. (2006, August). *CMMI® for Development, Version 1.2.* Retrieved 11 30, 2009, from http://www.sei.cmu.edu/reports/06tr008.pdf

CMU-Speech. (1995). *Carnegie Mellon University - Speech at CMU.* Retrieved 04 20, 2010, from http://www.speech.cs.cmu.edu/speech/

Cockburn, A. (2002). *Agile Software Development.* Boston, MA: Pearson Education, Inc.

Cockburn, A. (2005). *Crystal Clear: A Human-Powered Methodology for Small Teams.* Amsterdam: Addison-Wesley Longman.

Codehaus Foundation. (2006). *XPlanner*. Retrieved 05 28, 2010, from http://xplanner.org/

Coden, A., & Brown, E. (2001). Speech Transcript Analysis for Automatic Search. *Hawaii International Conference on System Science.* Maui, HI: Proceedings.

Cohn, M., & Ford, D. (2003, 06). Introducing an Agile Process to an Organization. *Computer , vol. 36* (issue 6), pp. 74-78.

CollabNet, Inc. . (2010). *ScrumWorks*. Retrieved 05 30, 2010, from http://www.danube.com/scrumworks

Conroy, J. M., & O'leary, D. P. (2001). Text summarization via hidden markov models. *In Proceedings of SIGIR'01*, (pp. 406-407). New York, NY.

Corrall, S. (1999, 02 03). *Knowledge Management: Are We in the Knowledge Management Business?* Retrieved 04 10, 2010, from http://www.ariadne.ac.uk/issue18/knowledge-mgt/

Das, D., & Martins, A. F. (2007, 11 21). *A Survey on Automated Text Summarization.* Retrieved 04 20, 2010, from www.cs.cmu.edu/~nasmith/LS2/das-martins.07.pdf

Daumé III, H., & Marcu, D. (2002). A noisy-channel model for document compression. *In Proceedings of the Conference of the Association of Computational Linguistics (ACL 2002)* (pp. 449 - 456). Philadelphia, PA: Association for Computational Linguistics .

Daumé III, H., & Marcu, D. (2004). A tree-prosition kernel for document compression. *In Proceedings of the Fourth Document Understanding Conference (DUC 2004).* Boston, MA.

Davenport, T. H., & Prusak, L. (1998). *Working Knowledge: How Organizations Manage What They Know.* Boston, MA: Harvard Business School Press.

Davies, K., Biddulph, R., & Balashek, S. (1952). Automatic Speech Recognition of Spoken Digits. *J. Acoust. Soc. Am. , No. 6*, pp. 637-642.

Denes, P. (1959). The Design and Operation of the Mechanical Speech Recognizer at University College London. *J. British Inst. Radio Engr.* , pp. 211-229.

Dijkstra, E. W. (1972, October). The humble programmer. (A. Press, Ed.) *Communications of the ACM , vol. 15* (no. 10), pp. 859-866.

Doyle, M., & Straus, D. (1976). *How To Make Meetings Work.* New York: Jove Books.

Drucker, P. F. (1998, 05 10). *Management's New Paradigms.* Retrieved 04 20, 2010, from http://www.forbes.com/forbes/1998/1005/6207152a.html

Drucker, P. F. (2006; Revised edition (1966)). *The Effective Executive: The Definitive Guide to Getting the Right Things Done.* Harper Paperbacks.

DSDM Consortium. (2009). *DSDM.* Retrieved 09 11, 2009, from www.dsdm.org

Eckstein, J. (2004). *Agile Software Development in the Large: Diving into the Deep.* New York: Dorset House.

Edelman, J., & Crain, M. B. (1994). *The Tao of Negotiation.* New York: HarperCollins Publishers.

Edmundson, H. P. (1969). New methods in automatic extracting. *Journal of the ACM , 16* (2), pp. 264-285.

El Emam, K., & Koru, A. G. (2008, September/October). A Replicated Survey of IT Software Project Failures. *IEEE Software* , pp. pp. 84-90.

EML Research gGmbH. (2005, 04 11). *Gesprächsprotokolle auf Knopfdruck?* . Retrieved 04 29, 2010, from http://www.eml-r.org/english/press/p5_presspart.php?we_objectID=236

Erman, L. D., Hayes-Roth, F., Lesser, V. R., & Reddy, R. D. (1980, 06). The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. (ACM, Ed.) *Computer Surveys , No. 2.*

Ferguson, J. (1980). *Hidden Markov Models for Speech.* Princeton, NJ: IDA.

Fink, A. (1995). *How To Sample in Surveys.* Thousand Oaks, California: SAGE Publications.

Forgie, J., & Forgie, C. (1959). Results Obtained From a Vowel REcognition Computer Program. *J. Acoust. Soc. Am. , No. 11*, pp. 1480-1489.

Fry, D. (1959). Theoretical Aspects of Mechanical Speech Recognition. *J. British Inst. Radio Engr.* , pp. 211-229.

Garfolo, J. S., Voorhees, E. M., Stanford, V. M., & Sparck Jones, K. (1997). TREC-6 1997 spoken document retrieval track overview and results. *In Proceedings of the 1997 TREC-6 Conference*, (pp. 83-91). Gaithersburg, MD.

Garofolo, J. S., Voorhees, E. M., Auzanne, C. G., & Stanford, V. M. (1999). Spoken document retrieval: 1998 evaluation and investigation of new metrics. *In Proceedings of the ESCA workshop: Accessing information in spoken audio*, (pp. 1-7). Cambridge.

Glass, R. L. (1998). *In the Beginning: Recollections of Software Pioneers.* IEEE Computer Society Press.

Glazer, H., Dalton, J., Anderson, D., Konrad, M., & Shrum, S. (2008). *CMMI® or Agile: Why Not Embrace Both!* Software Engineering Process Management.

Godfrey, J. J., Holliman, E. C., & McDaniel, J. (1992). SWITCHBOARD: telephone speech corpus for research and development. *In Proceedings of the ICASSP'92*, (pp. 517-520).

Goldman, S. L., Nagel, R. N., & Preiss, K. (1995). *Agile Competitors and Virtual Organizations.* New York: Van Nostrand Reinhold.

Google. (2010). *google*. Retrieved 04 24, 2010, from http://www.google.de/

Hastreiter, D. G., Roberts, S. R., & Mathis, D. C. (2009, 01). Leuchtfeuer entzünden: Einführung von Enterprise Scrum bei der Allianz Deutschland AG. *OBJEKTspektrum* , pp. 54-61.

Hauptmann, A., & Smith, M. A. (1995). Text, Speech and Vision for Video Segmentation: The Informedia Project. *AAAI Fall Symposium, Computational Models for Integrating Language and Vision.*

Hauptmann, A., & Witbrock, M. J. (1997). Informedia: News-on-Demand Multimedia Information Acquisition and Retrieval. *Intelligent Multimedia Information Retrieval* (pp. 213-239). AAAI Press.

Highsmith, J. A. (1999). *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems.* New York: Dorset House Publishing Co Inc.

Highsmith, J. (2002). *Agile Software Development Ecosystems.* Boston, MA: Addison-Wesley.

Highsmith, J. (2002, October). What Is Agile Software Development? *CrossTalk The Journal of Deense Software Engineering* , pp. 4 - 9.

Highsmith, J., & Cockburn, A. (2001, September). Agile Software Develpment: The Business of Innovation. *Computer - Software Management* , pp. 120 - 122.

Hirschberg, J., Whittaker, S., Hindle, D., Pereira, F., & Shinghal, A. (1999). Finding information in audio: A new paradigm for audio browsing/ retrieval. *In Proceedings of the ESCA workshop: Accessing information in spoken audio*, (pp. 117-122). Cambridge.

Hori, C., & Furui, S. (2000). Automatic speech summarization based on word significance and linguistic likelihood. *In Proceedings of ICASSP'00* , (pp. 1579-1582). Istanbul.

Hruschka, P., Rupp, C., & Starke, G. (2009). *Agility kompakt.* Heidelber: Spektrum.

Humphrey, W. S. (1988). Characterizing the Software Process: A Maturity Framework. (I. C. Press, Ed.) *IEEE Software , vol. 5* (issue 2), pp. 73 - 79 .

Hwang, M., Rosenfeld, R., Thayer, E., Mosur, R., Chase, L., Weide, R., et al. (1994). Improving Speech Recognition Performance via Phone-Dependent VQ. *In Proceedings of ICASSP-94*, (pp. 549-552).

ISO. (2009). *International Standardization for Organization*. Retrieved 11 30, 2009, from http://www.iso.org/iso/home.htm

Itakura, F. (1975). Minimum Prediction Residual Applied to Speech Recognition. *IEEE Trans. Acoustics, Speech, Singal Proc.*, (pp. 67-72).

Jelinek, F. (1985). The Development of an Experimental Discrete Dictation Recognizer. *Proc. IEEE*, (pp. 1616-1624).

Jelinek, F., Bahl, L., & Mercer, R. (1975). Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech. *IEEE Trans. Information Theory*, (pp. 250-256).

Jones, G. J., Foote, J. T., Jones, K. S., & Young, S. J. (1995). video Mail Retrieval: The Effect of Word Spotting Accuracy on Precision. *International Conference on Acoustics, Speech, and Signal Processing* (pp. 309-312). Detroit, MI: Proceedings, IEEE.

Jones, G., Foote, J., Jones, K., & Young, S. (1996). Retrieving Spoken Documents by Combining Multiple Index Sources. *International Conference on Research and Development in Information Retrieval* (pp. 30-38). Zurich: Proceedings, ACM SIGIR.

Kameyama, M., & Arima, I. (1984). Coping with aboutness complexity in information extraction from spoken dialogues. *In Proceedings of the ICSLP 94*, (pp. 87-90). Yokohama.

Kameyama, M., Kawai, G., & Arima, I. (1996). A real-time system for summarizing human-human spontaneous spoken dialogues. *In Proceedings of the ICSLP'96*, (pp. 681-684).

Kaminski, S. (2006). *Communication Models*. Retrieved 04 07, 2010, from http://www.shkaminski.com/Classes/Handouts/Communication%20Models.htm#SchrammsInteractiveModel1954

kathana. (2010). *Sinhala speech recognition system.* Retrieved 05 25, 2010, from kathana.googlecode.com/files/Project%20Praposal.pdf

Kayser, T. A. (1995). *Mining Group Gold: How to Cash in on the Collaborative Brain Power of a Group* (2 ed.). Irwin Professional Publishing.

Kerzner, H. (2006). *Project Management - A Systems Approach to Planning, Scheduling, and Controlling* (9th ed. ed.). Hoboken, New Jersey: John Wiley & Sons, Inc.

Knight, K., & Marcu, D. (2000). Statistical-based summarization - step one: Sentence compression. *In AAAI/IAAI* (pp. 703-710). Austin, TX: MIT Press.

Kunz, W., & Rittel, H. W. (1970). *Issues as Elements of Information Systems, Working paper No. 131.* Heidelberg: Studiengruppe für Systemforschung.

Kupiec, J., Pedersen, J., & Chen, F. (1995). A trainable document summarizer. *In Proceedings SIGIR '95*, (pp. 68-73). New York, NY.

Larman, C. (2004). *Agile and Iterative Development: A Manager's Guide.* Boston: Addison-Wesley.

Laufer, H. (2009). *Sprint-Meetings statt Marathon-Sitzungen: Besprechungen effizient organisieren und leiten.* Offenbach: GABAL-Verlag GmbH.

Lavie, A., Waibel, A., Levin, L., Finke, M., Gates, D., Gavaldà, M., et al. (1997). Janus III: Speech-to-speech translation in multiple languages. *In IEEE International Conference on Acoustics, Speech and Signal Processing*, (pp. 99-102). Munich.

LDC, Linguistic Data Consortium. (1999). Treebank-3: CD-ROM containting databases of disfluency annotated Switchboard transcripts.

Lee, C., & Rabiner, L. (1989). A Frame Synchronous Network Search Algorithm for Connected Word Recognition. *IEEE Trans. Acoustics, Speech, Signal Proc.*, (pp. 1649-1658).

Lee, C., Rabiner, L., Pieraccini, R., & Wilpon, J. (1990). Acoustic Modeling for Large Vocabulary Speech Recognition. *Computer Speech and Language* , pp. 127-165.

Lee, K., Hon, H., & Reddy, D. (1990). An Overview of the SPHINX Speech Recognition System. *IEEE Trans. Acoustics, Speech, Signal Proc.*, (pp. 600-610).

Leffingwell, D. (2007). *Scaling Software Agility: Best Practices for Large Enterprises.* Amsterdam: Addison-Wesley Longman.

Lewis, B. (1997, 01 08). *The Importance of Meeting Minutes*. Retrieved 04 15, 2010, from http://web.mit.edu/brlewis/www/minutes/index.html

Lighthouse Documentation. (2009, 12 02). *lighthousedocumentation.* Retrieved 04 10, 2010, from http://www.lighthousedocumentation.com/BestPracticesSample1.PDF

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In S. S. Marie-Francine Moens (Ed.), *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop* (pp. 74-81). Barcelona: Association for Computational Linguistics.

Lin, C.-Y. (1999). Training a selction function for extraction. *In Proceedings of CIKM'99*, (pp. 55-62). New York, NY.

Lin, C.-Y., & Hovy, E. (1997). Identifying topics by position. *In Proceedings of the Fifth conference on Applied natural language processing*, (pp. 283-290). San Francisco, CA.

Lippmann, R. (1987). An Introduction to Computing with Neural Nets. *IEEE ASSP Mag.*, (pp. 4-22).

Longstreet, D. (2008). *The Agile Method and Other Fairy Tales.* Retrieved 11 30, 2009, from http://www.softwaremetrics.com/Agile/Agile%20Method%20and%20Other%20Fairy%20Tales.pdf

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research Development* , pp. 159-165.

Marcu, D. (1998a). Improving summarization through rhetorical parsing tuning. *In Proceedings of The Sixth Workshop on Very Large Corpora*, (pp. 206-215). Montreal.

Marcu, D. (1998b). The rhetorical parsing, summarization, and generation of natural language texts. PhD thesis, University of Toronto. Adviser-Graeme Hirst.

Martin, T., Nelson, A., & Zadell, H. (1964). Speech Recognition by Feature Abstraction Techniques. *Tech. Report AL-TDR-64-176, Air Force Avionics Lab* .

Mauldin, M. (1989). *Information Retrieval by Text Skimming.* Ph.D. Thesis, Carnegie Mellon University. August 1989. Revised edition published as "Conceptual Information Retrieval: A Case Study in Adaptive Partial Parsing, Kluwer Press, September 1991.

McAdam, R., & McCreedy, S. (2000, September). A critique of knowledge management: Using a social constructionist model. *New Technology, Work and Employment , Vol. 15* (No. 2), pp. pp. 155-168.

McKeown, K., & Radev, D. (1995). Generating summaries of multiple news articles. *In Proceedings of SIGIR'95*, (pp. 74-82). Seattle, Washington.

Microsoft. (2010). *Bing*. Retrieved 04 24, 2010, from http://www.bing.com/

Microsoft. (2010). *Project 2010*. Retrieved 05 30, 2010, from http://www.microsoft.com/project/en/us/default.aspx

Miller, G. A. (1995). Wordnet: a lexical database for english. *Commun. ACM , 38* (11), pp. 39-41.

Mountain Goat Software. (2005). *Planning Poker*. Retrieved 11 30, 2009, from http://www.planningpoker.com/

MRT PLM Group Europe. (2007, 09 07). *Product Evolution Process @ Siemens.* Retrieved 05 30, 2010, from www.mrtplm.de/downloads/success_prodoma_siemens_de.pdf

Müller, C., Bahrs, J., & Gronau, N. (2005, 11 28). Considering the Knowledge Factor in Agile Software Development. *Journal of Universal Knowledge Management* , pp. 128-147.

Myers, C., & Rabiner, L. (1981). A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition. *IEEE Trans. Acoustic, Speech, Singal Proc.*, (pp. 284-297).

Nagata, K., Kato, Y., & Chiba, S. (1963). Spoken Digit Recognizer for Japanese Language. *NEC Res. Develop. , No. 6*.

Natural Speech Communication Ltd. (2005, 07). *Key-Word Spotting - The Base Technology for Speech Analytics.* Retrieved 05 27, 2010, from www.crmxchange.com/whitepapers/pdf/NSC-kws.pdf

Nebulon Pty. Ltd. (2002). *Feature Driven Development*. Retrieved 11 30, 2009, from http://www.featuredrivendevelopment.com/

Nenkova, A. (2005). Automatic text summarization of newswire: Lessons learned from the document understanding conference. *In Proceedings of AAAI 2005*, (pp. 1436-1441). Pittsburgh.

Niekrasz, J., & Purver, M. (2005). A Multimodal Discourse Ontology for Meeting Understanding. *Proceedings of MLMI'05. LNCS* (pp. 162-173). Springer Verlag.

NIST. (2002, 07 16). *Document Understanding Conferences*. Retrieved 04 24, 2010, from http://duc.nist.gov/

NIST. (2000, 08 01). *Text REtrieval Conference*. Retrieved 04 24, 2010, from http://trec.nist.gov/

Nonaka, I. (1994, February). A Dynamic Theory of Organizational Knowledge Creation. *Organization Science* , pp. pp. 14-37.

Nonaka, I., & Takeuchi, H. (1995). *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation.* New York: Oxford University Press .

Ogunnaike, B. A., & Ray, W. H. (1994). *Process Dynamics, Modeling and Control.* New York: Oxford University Press.

Olson, H., & Belar, H. (1956). Phonetic Typewriter. *J. Acoust. Soc. Am. , No. 6*, pp. 1072-1081.

Ono, K., Sumita, K., & Miike, S. (1994). Abstract generation based on rhetorical structure extraction. *In Proceedings of Coling'94*, (pp. 344-348). Morristown, NJ.

Osborne, M. (2002). Using maximum entropy for sentence extraction. *In Proceedings of the ACL'02 Workshop on Automatic Summarization*, (pp. 1-8). Morristown, NJ.

Owen, H. (1997). *Open space technology: a user's guide.* San Francisco, CA: Berrett-Koehler Publishers.

Oxford Corpus. (2005). *Oxford Dictionary of English.* (C. Soanes, & A. Stevenson, Eds.) Oxford University Press.

Oxford University Press. (2010). *Communication*. Retrieved 05 22, 2010, from http://www.oxforddictionaries.com/definition/communication?view=uk

Paul, D. (1989). The Lincoln Robust Continuous Speech Recognizer. *Proc. ICASSP 89*, (pp. 449-452). Glasgow.

Pawar, R. V., Kajave, P. P., & Mali, S. N. (2005, 12). *Speaker Identification using Neural Networks.* Retrieved 05 28, 2010, from www.waset.org/journals/waset/v12/v12-7.pdf

Pfleeger, S. L., & Atlee, J. M. (2009). *Software Engineering: Theory and Practice.* Upper Saddle River, NJ: Prentice Hall.

Pichler, R. (2008). *Scrum - Agiles Projektmanagement erfolgreich einsetzen.* Heidelberg: dpunkt.verlag GmbH.

Polanyi, M. (1966). *The Tacit Dimension.* Chicago: University of Chicago Press.

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit.* Upper Saddle Rier, NJ: Addison-Wesley.

Prager, J., Brown, E., Coden, A., & Radev, D. (2000). Question-Answering by Predictive Annotation. *Conference on Research and Development in Information Retieval* (pp. 184-191). Athens: Procedings, ACM SIGIR .

Rabiner, L. (1989, 02). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceeding on the IEEE* , pp. 257-286.

Rabiner, L. (1993). *Fundamentals of Speech Recognition.* Englewood Cliffs, NJ: Prentice Hall.

Rabiner, L., Levinson, S., Rosenberg, A., & Wilpon, J. (1979). Speaker Independent Recognition of Isolated Words Using Clustering Techniques. *IEEE Trans. Acoustics, Speech, Signal Proc.*, (pp. 336-349).

Radev, D. R., Hovy, E., & McKeown, K. (2002). Introduction to the special issue on summarization. *Computational Linguistics , 28* (4), pp. 399-408.

Randell, B. (2001, August 10). *The 1968/69 NATO Software Engineering Reports.* Retrieved 04 14, 2010, from http://homepages.cs.ncl.ac.uk/brian.randell/NATO/NATOReports/index.html

Ravin, Y., Wacholder, N., & Choi, M. (1997). Disambiguation of Names in Text. *Conference on Applied Natural Language Processing* (pp. 202-208). Washington, DC: Procedings, ACL.

Reddy, D. (1966). *An Approach to Computer Speech Recognition by Direct Analysis of Speech Wave.* Computer Science Dept., Stanford Univ: Tech. Report No. C549.

Reithinger, N., Kipp, M., Engel, R., & Alexandersson, J. (2000). Summarizing multilingual spoken negotiation dialogues. *In Proceedings of the 38th Conference of the Association for Computational Linguistics*, (pp. 310-317). Hongkong.

Romano, N. C., & Nunamaker, J. F. (2001). Meeting analysis: Findings from research and practice. *34th Hawaii International Conference on System Sciences* (p. 1072). IEEE Computer Society Press .

Rosnow, R. L., & Rosenthal, R. (2008). *Beginning Behavioral Research* (6. Edition ed.). New Yersey: Pearson Education - Prentice Hall.

Rudnicky, A. (1995). Language Modeling with Limited Domain Data. *In Proceeding of the 1995 ARPA Workshop on Spoken Language Technology*, (pp. 66-69).

Rundle, P. J., & Dewar, R. G. (2006). Using return on investment to compare agile and plan-driven practices in undergraduate group projects. *International Conference on Software Engineering. Proceedings of the 28th international conference on Software engineering* (pp. 649 - 654). ACM.

Rüping, A. (2003). *Agile Documentation.* West Sussex: John Wiley & Sons Ltd.

Sakai, T., & Doshita, S. (1962). The Phonetic Typewriter, Information Processing 1962. *Proc. IFIP Congress.* Munich.

Sakoe, H. (1979). Two Level DP Matching - A Dynamic Programming Based Pattern Matching Algorithm for Connected Word Recognition. *IEEE Trans. Acoustics, Speech, Signal Proc.*, (pp. 588-595).

Sakoe, H., & Chiba, S. (1978). Dynamic Programming Algorithm Optimization of Spoken Word Recognition. *IEEE Trans. Acoustics, Speech, Singal Proc.*, (pp. 43-49).

Saon, G., Zweig, G., Huang, J., Kingsbury, B., & Mangu, L. (2001). Evoluation of the Performance of Autmatic Speech Recognition Algorithms in Transcribing Conversational Telephone Speech. *Instrumentation and Measruement Technology Conference.* Budapest: Proceedings.

Schein, E. H. (2004). The Role of the Founder in Creating Organizational Culture. In J. T. Wren, D. A. Hicks, & T. L. Price, *Modern Classics on Leadership* (pp. 443 - 458). Northampton, MA: Edward Elgar Publishing, Inc.

Schiller, J. (2009). Modern Meeting Management and Information Retrieval. *OOPSLA 2009.* Orlando, Florida: ACM.

Schiller, J. (2008). Word Spotting in Scrum Meetings. *19th International Conference on Database and Expert Systems Application, DEXA* (pp. 125 - 129). Turin: IEEE.

Schiller, J., & Canditt, S. (2008, 03). Das Ganze sehen: Auf dem Weg zum agilen Unternehmen . *OBJEKTspektrum* .

Schmelzer, H. J., & Sesselmann, W. (2007). *Geschäftsprozessmanagement in der Praxis. Kunden zufrieden stellen, Produktivität steigern, Wert erhöhen.* Hanser Fachbuch.

Schramm, W. (1961). *How Communication Works.* Urbana, Ill: The University of Illinois Press.

Schuh, P. (2005). *Integrating Agile Development in the Real World.* Hingham, MA: Charles River Media.

Schwaber, C. (2005). *Corporate IT Leads the Second Wve of Agile Adoption.* Forrester Research Inc. report.

Schwaber, K. (1995). SCRUM Development Process. *OOPSLA'95 Workshop on Business Object Design and Implementation* .

Schwaber, K., & Beedle, M. (2002). *Agile Software Development with Scrum.* Upper Saddle River, NJ: Prentice Hall.

Schwaber, K., & Beedle, M. (2002). *Agile Software Development with Scrum.* Upper Saddle River, NJ: Prentice-Hall.

Selman, B., Levesque, H., & Mitchelle, D. (1992). A new method for solving hard satisfiability problems. *In Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, (pp. 440-446). San Jose, CA.

SEN-iPhone Team. (2009, 07 08). *iphone09sen*. Retrieved 05 20, 2010, from https://teambruegge.informatik.tu-muenchen.de/groups/iphone09sen/

Shannon, C. E. (1948). *A Mathematical Theory of Communication.*

Slinger, M., & Broderick, S. (2008). *The Software Project Manager's Bridge to Agility.* Amsterdam: Addison-Wesley Longman.

Sommerville, I. (2002, 10 01). *Software Engineering.* Retrieved 04 20, 2010, from http://www.comp.dit.ie/rlawlor/Soft_Eng/Sommerville/ch01%20-%20Software%20v2.ppt

Standish Group. (2009, 04 23). *CHAOS Summary 2009.* Retrieved 12 15, 2009, from http://www1.standishgroup.com/newsroom/chaos_2009.php

Stifelman, L. J. (1995). A discourse analysis approach to structured speech. *In AAAI'95 Spring Symposium on Empirical Methods in Disourse Interpretation and Generation*, (pp. 162-167). Stanford, CA.

Sutherland, J. (2005, June 03). *Future of Scrum: Support for Parallel Pipelining of Sprints in Complex Projects.* Retrieved 11 30, 2009, from http://jeffsutherland.com/scrum/Sutherland2005FutureofScrum20050603.pdf

Suzuku, J., & Nakata, K. (1961). Recognition of Japanese Vowels - Preliminary to the Recognition of Speech. *J. Radio Res. Lab* , pp. 193-212.

Svore, K., Vanderwende, L., & Burges, C. (2007). Enhancing single-document summarization by combining RankNet and third-party sources. *In Proceedings of the EMNLP-CoNLL* (pp. 448-457). Prague: Association for Computational Linguistics.

Takeuchi, H., & Nonaka, I. (1986, 01). The New New Product Development Game. *Harvard Business Review* , pp. 1-11.

Tan, S., Teo, H.-H., Tan, B., & Wei, K.-K. (1998). Developing a Preliminary Framework for Knowledge Management in Organizations. *Proceeding of the Fourth Americas Conference on Information Systems (AMCIS)*, (pp. pp. 629-631). Baltimore.

Tappert, C., Dixon, N., Rabinowitz, A., & Chapman, W. (1971). *Automatic Recognition of Continuous Speech Utilizing Dynamic Segmentation, Dual Classification, Sequential Decoding and Error Recovery.* Rome, NY, Tech. Report TR-71-146: Rome Air Dev. Cen.

Thompson, M. P., & Walsham, G. (2004, July). Placing Knowledge Management in Context. *Journal of Management Studies , Vol. 41* (No. 5), pp. pp. 725-747.

Toffler, A. (1970). *Future shock.* Bantam.

U.S. Food and Drug Administration. (2009). *U.S. Food and Drug Administration*. Retrieved 11 30, 2009, from http://www.fda.gov/

Valenza, R., Robinson, T., Hickey, M., & Tucker, R. (1999). Summarisation of spoken audio through information extraction. *In Proceedings of the ESCA workshop: Accessing information in spoken audio*, (pp. 111-116). Cambridge.

Van Schooenderwoert, N. (2005, July 25). *The Essence of Agile - Empirical Process vs. Defined Process*. Retrieved 05 10, 2010, from http://www.agilerules.com/viewaaa.phtml?document=Vol%2002%20Issue%2008-%20The%20Essence%20of%20Agile%20-%20Empirical%20Process%20vs.%20Defined%20Process

Velichko, V., & Zagoruyko, N. (1970, 06). Automatic Recogniton of 200 Words. *Int. J. Man-Machine Studies* , p. 223.

VersionOne. (2009). *State of Agile Development Survey 2009.* Retrieved 04 03, 2010, from http://pm.versionone.com/StateOfAgileSurvey.html

VersionOne. (2010). *VersionOne Agile Survey*. Retrieved 04 15, 2010, from http://www.versionone.com/Resources/Whitepapers.asp

VersionOne, Inc. (2010). *VersionOne*. Retrieved 05 15, 2010, from http://versionone.com/

Vintsyuk, T. (1968, 01/02). Speech Discrimination by Dynamic Programming. *Kibernetika , No. 2*, pp. 81-88.

Wactlar, H., Christel, M., Gong, Y., & Hauptmann, A. (1999, February). Lessons Learned from Bilding a Terabyte Digital Video Library. *IEEE Computer* .

Waibel, A., Bernardin, K., & Wölfel, M. (2007). Computer-Supported Human-Human Multilingual Communication. In *50 Years of Artificial Intelligence* (pp. 271-287). Berlin / Heidelberg: Springer.

Waibel, A., Bett, M., & Finke, M. (1998). Meeting Browser: Tracking and Summarizing Meetings. Lansdowne, VA: Proceedins, DARPA Broadcast News Transcription and Understanding Workshop.

Waibel, A., Bett, M., Metze, F., Ries, K., Schaaf, T., Schultz, T., et al. (2001). Advances In Automatic Meeting Record Creation And Access. *Acoustics, Speech, and Signal Processing, 2001 Vol 1. 2001 IEEE International Conference* (pp. 597-600). Salt Lake City, UT: icassp.

Weibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. (1989). Phoneme Recognition Using Time-Delay Neural Networks. *IEEE Trans. Acoustics, Speech, Signal Proc.*, (pp. 393-404).

Weintraub, M., & et al. (1989). Linguistic Constraints in Hidden Markov Model Based Speech Recognition. *Proc. ICASSP 89*, (pp. 699-702). Glasgow.

Whittaker, S., Hirschberg, J., Choi, J., Hindle, D., Pereira, F., & Singhal, A. (1999). SCAN: Designing and evaluating user interfaces to support retrieval from speech archives. *In Proceedings of the 22nd ACM-SIGIR International Conference on Research and Development in Information Retrieval*, (pp. 26-33). Berkeley, CA.

Wikimedia Foundation. (2001). *Wikipedia*. Retrieved 04 24, 2010, from http://wikipedia.org/

Williams, L., & Cockburn, A. (2003, June). Agile Software Development: It's about Feedback and Change. *Computer, vol. 36, no. 6* , pp. 39-43.

Williams, M. (2008). *The Principles of Project Management.* SitePoint Pty. Ltd.

Witbrock, M., & Mittal, V. (1999). Ultra-summarization (poster acstract): a stasticial approach to generating highly condensed non-extractive summaries. *In Proceedings of SIGIR'99*, (pp. 315-316). New York, NY.

Yegge, S. (2006, September 27). *Good Agile, Bad Agile*. Retrieved 11 30, 2009, from http://steve-yegge.blogspot.com/2006/09/good-agile-bad-agile_27.html

Zechner, K. (2001). *Automatic Summarization of Spoken Dialogues in Unrestricted Domains.* Carnegie Mellon University: Ph.D. thesis. Language Technologies Institute, School of Computer Science.

Zechner, K. (2002, January 14). Summarization of Spoken Language. *Speech Technology Expert eZine* , pp. 1-14.

Zechner, K., & Waibel, A. (2000). DIASUMM: Flexible summarization of spontaneous dialogues in unrestricted domains. *In Proceedings of COLING-2000*, (pp. 968-974). Saarbruecken.

Zhang, H., Low, C., & Smoliar, S. (1995, March). Video parsing and indexing of compressed. *Multimedia Tools and Applications* , pp. 89-111.

Zue, V., Glass, J., Phillips, M., & Seneff, S. (1989). The MIT Summit Speech Recognition System: A Progress Report. *Proc. DARPA Speech and Natural Language Workshop*, (pp. 179-189).