Case-Study: How to Increase the Value of Computer Science Projects in Higher Education

Daniel Kadenbach, Carsten Kleiner
University of Applied Sciences and Arts Hannover, Germany
{daniel.kadenbach, carsten.kleiner}@fh-hannover.de

***Abstract*—This paper describes the efforts in the department of computer science of the University of Applied Sciences and Arts Hannover to bring forward and sustainably increase the value of research and teaching projects. The aim is to optimally support these projects which are carried out by students and staff members, using appropriate software tools and virtual environments. The special requirements for such a supportive system, which were gathered and refined over time, are analyzed in detail. These requirements serve as a foundation of our implementation. Its evolution is described in consecutive stages which continuously improve the quality of project support by learning from each stage. Furthermore, challenges and benefits of supportive systems are discussed and a vision for future developments towards a virtual, federated value-creating community is drafted.**

***Index Terms*—cscw, collaboration, project support, virtual communities, education**

I. Introduction

The aim of this paper is to describe ways to bring forward and sustainably increase the value of research and teaching projects in higher education. Bringing projects forward in this context means to support and disburden their implementation. This can be accomplished by facilitating communication, cooperation and the use of shared resources for project teams by means of suitable tools. It is especially important when team members do not always work together at the same place to improve the awareness of the team members. Depending on the project, the supporting application of communication systems like email, mailing lists or instant messengers, of knowledge management systems like content management systems in general, especially wikis and of common repositories, can help to reach this goal.

There are two ways investigated to increase the value of projects: Firstly, by giving them a framework by means of a better and coordinated support which improves their development and can lead to better results. Secondly, by creating an increased awareness of the project not only within the project team and project lifetime but also beyond it. A project should be visible and durably accessible. Such greater visibility also leads to increased possibilities concerning the use of the results of the project, a greater exchange of knowledge and the possibility to learn from each other. Therefore, it increases the value of projects not only for single persons but also for the community and leads to a greater motivation to achieve results of high quality. By means of a lasting availability of those results, the value of single projects increases sustainably. This whole process is illustrated in Figure 1.

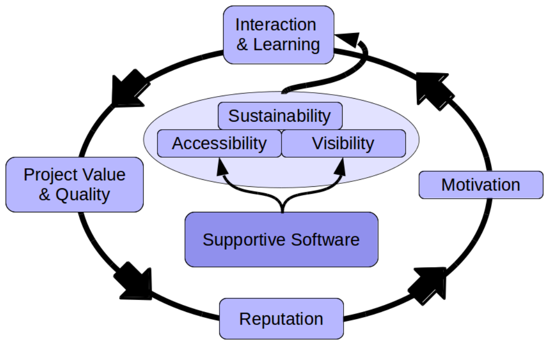


Fig. 1. Potential Positive Effects of Project Support

An ideal supporting environment should not only supply single supporting tools but also integrate them into a complete system and add comprehensive overall functions to benefit from synergistic effects and social mechanisms. Still, supporting systems can also lead to risks for projects: For each project, a reasonable choice of tools needs to be made. The tools chosen should not only depend on the project's topic but also on the experiences and preferences of the team members and the team culture. The acceptance by the team is essential. Inappropriate tools can not only create needless additional expenses but also decrease the motivation of team members endangering the progress. Therefore, such systems should be implemented carefully and their use should be a free choice.

A. Motivation and Aims

We have been motivated to create better and more sustainable support for projects by our observation of the usual development of many previous projects. This development has been characterized by a lack of visibility which led to the fact that brilliant achievements of single project teams even became inaccessible shortly after the end of their project lifetime. In many cases, special supportive tools which had been administrated by the project team itself and had not been accessible for other teams, were used in those projects. Most often, those tools also disappeared after the projects were finished. Thus, the knowledge concerning the support and results of projects has neither been accumulated nor been available for others. How tragic is it if the results of projects are above the ordinary, but those results are not used any further and are not even visible? And how tragic is it if projects are often implemented completely isolated, so that there is no

COLLABORATECOM 2012, October 14-17, Pittsburgh, United States
Copyright © 2012 ICST
DOI 10.4108/icst.collaboratecom.2012.250425

possibility to learn from others, to profit by an exchange of knowledge and ideas or to build on their results?

The main goals concerning the improvement of this situation already arise from those questions:

1) Supporting projects by providing optimal sets of tools
2) Protection of the content of projects and the availability and preservation of access to it – therefore enabling its reuse and further development
3) Improvement of creation of useful documentation, interaction, communication and motivation
4) Encouragement of reasonable and sustainable projects

Implementing a supporting system which fulfills the listed goals poses a great challenge. The goal of this paper is to highlight in which way this challenge has been faced in our department and which experiences we gained, so others may benefit from them. Since the year 2009 we are working on such a system, which is constantly improving and refined and has been used by students and staff members from its start on. Alas, our efforts are restricted because only one staff member with a half-time position may concentrate on this task.

B. Related Work

A general introduction into project support systems can be found in [1]. Different approaches to support projects in higher education are presented in [2] and [3] among others. We are focusing more on the requirements which allow to evaluate different systems and the investigation of different stages of project support.

The design of our system and the selection of its components was heavily influenced by the experiences and research of other authors [4] [5] [6]. So [7] and [8] describe their experiences with project management supportive tools like Trac, source code repositories like Subversion and the use of wikis to support projects for capstone projects. The benefits of the use of wikis to accumulate and preserve knowledge has also already been researched. [9] shows their importance to create an organizational memory. [10] investigate the challenges of the use of wikis in organizations. How to support learning with collaborative environments, which is also an aim of the system described in this paper, is also discussed in [11] and [12].

Furthermore, this paper is based on the experiences of several years of project support at our department, and therefore also on previous results presented in [13], [14] which are extended and refined. The first paper from 2009 describes fundamental aspects and requirements for the conception of project supporting systems using social software mechanisms, whereas the second paper discusses the results of two surveys carried out in the department to incrementally refine the requirements and draft trends. In contrast to that, this paper builds on those results and describes the experiences with an actual implementations of the afore drafted system in consecutive stages. In contrast to that, this paper builds on those results and describes the experiences with an actual implementations of the afore drafted system in consecutive stages.

C. Structure

Section two investigates the requirement analysis for the support of projects in our department. The analysis process in this special environment with its unique difficulties is explained and the results gathered are outlined. Building on section two, sections 3-7 show the evolution of implementation stages which have been carried out and are planned to meet the requirements, summing up our experiences, while section 8 describes alternative approaches. Finally, section 9 summarizes our conclusions and open questions.

II. Requirements for Project Support

Like with many systems in the field of CSCW, a requirement analysis regarding project support has to take special impeding factors into account. Those factors will be explained in the following subsection. They lead to the fact that the requirement analysis can only be a continuing, incremental and iterative process. Following those observations, the concrete methodology and the current results are explained. The requirement analysis can already be seen as as a valuable conclusion because of the difficulties which are connected with its implementation. Still, the results of it in any specific project of course do not have a general character but rather have to be adapted to the existing circumstances of each individual project.

A. Challenges of the Requirement Analysis

Diversity of Projects. Projects often differ significantly concerning aspects such as their complexity (concerning the size of the team, the number of stakeholders and the project lifetime), content, aims, applied technologies, experiences and preferences of the team members, the team spirit, control mechanisms and forms of communication. Project support has to respect those differences. Different projects will need different supportive tools and tools which are perfectly suited to support one project can be a burden for others. Therefore, each project has to have the option to choose supportive tools individually, which just leads to the next limiting factor: the experiences of the project team.

Experiences of Target Group. The fact that every person can only give answers according to her or his horizon of experience and therefore, may fail to discover important aspects, always has to be considered if surveys are conducted among people who will use project support in the future. For example, a person who has not yet gained experience with the work with wikis will hardly be able to make a statement about future perspectives concerning the use of wikis in her or his projects. Therefore it is very important to make students familiar with as many tools as possible. This is the only way in which they can gain the ability to evaluate the value of these tools and decide for or against certain tools for their projects.

Constant Development & Challenging Evaluation. New tools which are suitable for the support of projects are constantly developed as well as there are new versions of existing tools. Additionally, there are always new technologies used to develop projects. Because of those innovations, new promising tools and new versions of existing tools have to be evaluated continuously to be able to assess their value. Beyond that, advantages, disadvantages and challenges which may arise from the use of a tool can not be evaluated easily. It is not enough to analyze, list and test the functional and non-functional features of a tool. In addition to the integration of the tool into the existing team infrastructure and culture, another crucial factor is the acceptance by the individual project teams which can hardly be estimated beforehand. Most often, it can only be evaluated in the course of the application of the tool. In this way, experiences develop. However, those experiences can only be applied to other projects with great carefulness because of the diversity of projects. Still, tools with great potential can be distinguished from tools with less potential and the former can be classified into groups according to the type of project which can benefit the most from them.

B. Methodology

The first requirements had been collected by observing student and staff projects and their special needs. But to create a comprehensive supporting system which should also be able to meet further needs and to improve the offered services for student and staff projects, two surveys have been carried out, which led to incremental improvements of the support. In February 2009 and November 2010, students were asked detailed questions about their experiences and requirements for their projects. The iterative-incremental approach to the requirements analysis can already be recognized in those two surveys. The results of the surveys had to be evaluated carefully because of the described limitations concerning the horizon of experience of the respondents. As the core of the questions had been the same in both surveys, trends could be deduced from the results [14]. Furthermore requirements gathered from experiences of the daily work with project teams were constantly collected. This made it possible to directly react to them by evaluating possible tools and testing the most mature ones directly with the teams in their projects. Additionally, mostly technical requirements came up while designing and implementing supportive functions.

C. Results

Using the aforementioned sources of information, previous research and experiences from projects the following requirements have been identified for a supporting system for departments with focus on software engineering projects. They are divided into functional and non-functional requirements, as shown in Table I and Table II (In these tables the importance of the requirements for our implementation is additionally weighted with three to one bullets. For functional requirements three bullets mean "absolutely necessary", two bullets "very sensible" and one bullet "nice to have"). Even though these requirements might seem obvious, the essence is to integrate them altogether in a single application to utilize their potential and synergetic effects. This has not been done hitherto in a contending way to the knowledge of the authors.

TABLE I
LIST OF FUNCTIONAL REQUIREMENTS

ID	*Functional Requirements*	*Importance*
R1.1	User and Project Administration	● ● ●
R1.2	Project Portal	● ● ●
R1.3	Version Control Systems	● ● ●
R1.4	Knowledge Management	● ● ●
R1.5	Integration and Social Functions	● ● ●
R1.6	Issue Tracking Systems	● ● ●
R1.7	Project Management Support	● ●
R1.8	Continuous Integration Environments	● ●
R1.9	Communication System Support	●
R1.10	Artifact Repositories	●
R1.11	Export Functions	●

TABLE II
LIST OF NON-FUNCTIONAL REQUIREMENTS

ID	*Non-Functional Requirements*	*Importance*
R2.1	User Acceptance	● ● ●
R2.2	Additional Value vs. Additional Effort	● ● ●
R2.3	Teaching & Training	● ● ●
R2.4	User Support	● ●
R2.5	Response Time	● ●
R2.6	Extensibility	● ●
R2.7	Self Administration & Automation	● ●
R2.8	System Safety & Security	● ●
R2.9	Updates	●

1) Functional Requirements:

- **User and Project Administration.**(R1.1) To be able to offer different tools and services user-friendly and to decrease administrative overhead, the system needs a central user management component. Thus, users only need to create one account to use all the different services of the system. Projects with their corresponding teams should also be manageable within this component, so that all necessary settings for the subsystems can be performed automatically. If applicable, existing user directories (e.g. via LDAP) should be integrated.
- **Project Portal.**(R1.2) A central place for all projects is important to ensure the visibility of projects beyond their own boundaries and therefore, to facilitate its sustainability, better interaction and mutual learning. Such a portal should clearly list all projects and teams and should offer links to all project artifacts as well as ways to interact with the projects. It can then be seen as a project portfolio, which can also represent the activities of the department to a broader audience.
- **Version Control Systems.**(R1.3) A version control system can be of great value in many situations. Thus, project members can collectively work with a common file repository, investigate all changes of the files made by themselves and others and even be informed instantly about changes of other users. This can greatly improve the

awareness in the team. Additionally, other users can see changes and the progress of the project early and can give feedback. Even more complex workflows are available if using a sophisticated version control system like Git (for example, one person of the team who reviews all changes before they are committed to the repository to ensure their quality, could be specified).

- **Knowledge Management.**(R1.4) Another crucial aspect in every project is how to deal with the gathered information, experiences and knowledge, whose value is often significant. Without special tools which ensure the visibility of this information even beyond the project lifetime, they often sink into oblivion soon after projects are finished. Knowledge management can greatly improve this situation if suitable tools are chosen and if they are used sensible. They can offer a central place to gather all information where everyone can find, edit and update it, therefore greatly simplifying the documentation process and making it more attractive. It is much more likely that a useful documentation will be created when each team member can directly, immediately and without much effort edit and update her or his own special part of the documentation in a wiki. Additionally, this improves the possibility to receive feedback. Blogs are well suited to document the experiences with and solutions for daily problems in projects, which otherwise would be forgotten soon and would require nearly the same amount of time and effort the next time they arise.
- **Integration and Social Functions.**(R1.5) Bringing all these different functions together in a sensible way requires their integration into a surrounding system which acts as a common interface to the functions. Only then the different functions can be accessed easily, and even more important: Only then synergy effects emerging from the combination of tools can be utilized. Common functions of the surrounding system should not only enable the user to control and access the components but also to search, tag, comment and rate artifacts in all of them.
- **Issue Tracking Systems (ITS).**(R1.6) ITS are ideally for collecting and processing bug reports, since they also offer feedback mechanisms to the person who added the report. They are also useful to acquire new requirements through feature requests and therefore improve the interaction with the project when people outside the project team are allowed to add tickets. ITS can even be used as a tool for project management when tasks are tracked and multiple tasks are combined into milestones.
- **Project Management.**(R1.7) Supporting the project management with tools can be sensible, especially in complex projects. But also smaller projects can benefit if suitable tools are used in the right way. Being able to view all open tasks, their owners and their current state, to bundle tasks into milestones and monitor the progress and problems, to add new requirements and feature requests at a central visible location can greatly increase team awareness. Admittedly, this requires that the chosen system is accepted by the team and used sensible. But in this case, tutors or customers of the project can also easily monitor the project progress and interact with the project in a defined and self-documenting way.
- **Continuous Integration Environments.**(R1.8) Automated, iterative tests of software are a powerful way to develop more stable and secure software right from the beginning of its development, also beyond agile and test-driven development per se. Continuous Integration (CI) Environments reduce the effort to automatically test software and can support the developers even in other ways. Except from automatically checking out the project after a commit to the version control system, a CI server can build the project, run all tests, create code metrics, run static code analysis tools or code standard conformity tests and generate reports with all the gathered data – and all of that completely automatically.
- **Communication System Support.**(R1.9) A mail system is often already part of the school's infrastructure and therefore can easily be used to support asynchronous communication and to be integrated into several tools. For example, most version control systems can be configured to send mails to project participants if some files have been changed automatically, including a summary of the changes and its author. Mailing lists whose communication is automatically archived are a very useful extension. Instant messengers, whose infrastructure is freely available, offer increasingly useful functionality for synchronous communication support. Particularly logging important conversations, so that the team can refer to them, or using possibilities like sending files, sharing desktops, initiating conferences or using shared virtual white boards can improve the communication of distributed working team members. Since all those communication tools are normally already in use by users of the system and use independent infrastructure, this requirement has been weighted low for the central supporting system.
- **Artifact Repositories.**(R1.10) Especially in complex software engineering projects with many dependencies to other libraries, the use of artifact repositories provides advantages. If these libraries can be fetched from central repositories, the dependency handling can be left to special build tools like maven which automatically fetch all required versions for all dependencies. Additionally, artifacts which are uploaded to such repositories are also available to other developers.
- **Export Functions.**(R1.11) Ideally, all project contents should be exportable from the system, so that the users are not bound to it and can use or refine them in other environments or just archive them.

2) Non-Functional Requirements:

- **User Acceptance.**(R2.1) A crucial non-functional requirement is the acceptance by the users of the system. This requirement can hardly be estimated beforehand because it depends on many non-functional requirements as well as on a few factors which are hard to measure like the experience of the team, its team spirit, its preferences, its ability to learn and its enthusiasm.

- **Surplus Values.**(R2.2) It is important that the system and its functions contain an additional value which is clearly recognizable for the users from the beginning on. This additional value has to exceed the additional effort to use the system to a great extend. The effort concerning the training of new users has to be kept at a minimum. This can be reached by means of elaborate, intuitively operable, well-documented tools and teaching. Furthermore, the daily overhead to use the system has to stay minimal compared to the directly recognizable additional value.
- **Teaching and Training.**(R2.3) Teaching and training users of the system should never be neglected. It enables future users to get an idea of it and to use its functions easily. So they better understand the benefits and drawbacks of the functions and know how to use them.
- **User Support.**(R2.4) Errors in the system have to be detected and fixed quickly. The response time to support requests should be kept at a minimum. Proposals of the users concerning changes, extensions and the further development should always be investigated. Therefore it makes sense to use one of the mentioned ticket systems for the system itself. In this way, the support and further development can be shaped transparently.
- **Response Time.**(R2.5) In such a complex system, response times of its services are very important because fast responses are critical for user acceptance. For example, the multitude of different components can easily consume to much RAM if not configured properly, which can lead to too much swapping of the operating system which again can slow down access to services in the range of many seconds per response.
- **Extensibility.**(R2.6) The system has to be oriented towards extensibility due to constantly changing and refining requirements which have to be respected. The integration of new components should be possible with the least possible effort.
- **Self Administration & Automation.**(R2.7) To be able to provide such a complex system at all, automation is needed for every process where it can be implemented. As soon as a relatively small number of users and projects is exceeded, it becomes completely impossible to administer and maintain such a system manually in a satisfyingly way. Self administration should be used wherever possible, because the system and its functions should anyhow be used freely. Users should be able to use all functions through an intuitive interface, creating their user accounts, projects and choosing and activating required functions by themselves for each project utilizing the automation. This is also strictly necessary to provide the service for a great number of users and minimize administration overhead.

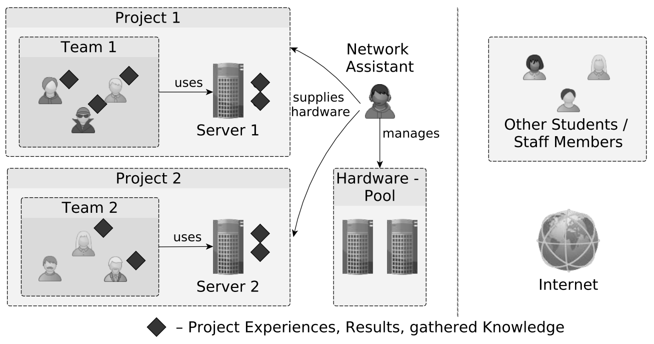


Fig. 2. Overview of Project Support Stage Zero – Only Supply Hardware

- **System Safety and Security.**(R2.8) A system like this one, composed out of a multitude of subsystems, inevitably has to be reviewed and monitored carefully to ensure its safety and security. This is even more valid if the system or some of its functions are exposed to the Internet, which seems essential to offer a good usability. To monitor the system, automatic tools can be used for intrusion detection and system health monitoring. Of course, the operating system and all components have to be updated regularly to the most stable version, which can quickly become a laborious task. Part of the safety and security concept should be a solid, tested backup concept for all user data and also the system itself.
- **Updates.**(R2.9) Every component of the system should be updated regularly to its last stable version to ensure security and offer new functionality.

Just like the requirement analysis, an implementation has to be iterated constantly to sufficiently adapt to the demands of the users and projects and to take current developments and newly available tools into account. Due to the variety of different requirements, it seems nearly impossible to develop a solution from the ground up or to find a single software system which already implements them all. Thus in the implementation, appropriate software components have to be found and sensibly integrated into an overall system to optimally fulfill the requirements. To deal with this huge amount of different requirements for supporting projects, our implementation has passed through several incremental steps in the course of time, using the gathered experiences to fulfill the requirements better. The advantages and disadvantages of these iterations, so-called *stages of project support*, shall be exemplified in the following. The stages are listed chronologically, whereas the potential of the project support increases from stage to stage. Thus this model may also allow to classify the state of project support in other institutions. Every implemented stage generated positive feedback and was used extensively. Succeeding stages compensate shortcomings or deal with challenges of the previous ones to clearly enhance their benefits.

III. Project Support Stage Zero – Only Supply Hardware

A. Description

In this stage, projects are supported by the department only by providing them with hardware, e.g. servers, for their work. The project team then has to administer and use the provided hardware autonomously by installing and configuring operating systems, software and tools as needed. For example, a project team could install a version control system like Subversion on its server, configure it and offer access for the team members to use it.

B. Implementation

The department only has to provide and manage hardware for the projects. Since the employee to which this task is assigned does not have to know anything about project supporting systems, she or he could be a technical assistant from the network or infrastructure team. This stage is shown in Figure 2.

C. Assessment

The advantages and disadvantages are obvious (a plus sign marks an advantage, a minus sign a disadvantage or challenge):

- **+** The administration overhead for the employee of the department is rather small. The provision of hardware like servers can be carried out by a member of the department network team with no further special knowledge about project support. Installations are carried out by the project teams, so that the computers only have to be brought back into their original state after a project finishes.
- **+** The project team has maximum freedom on how to use its resources, for example: which operating system to use and which software and tools to install.
- **–** The usage of the supplied resources is significantly restricted by the horizon of experiences and the administrative competence of the project team members. Students who have never worked with a version control system before will very likely not install one and struggle through all its configuration issues. The time which the project team can offer for administrative tasks also limits the usage of the resources.
- **–** Gained knowledge and results are not collected at a central, sustainable and accessible place, so they will very likely vanish by the end of the project. Additionally, there is no central place where knowledge regarding project support and the administration of tools is gathered. So even if a project team evaluates particularly suitable tools in its project, other teams will probably not benefit from it. Even worse: The project environments and data are very likely to be lost after the lifetime of the projects, when the hardware is returned. The chance to incrementally build on or enhance carried out projects, which is a central aspect of computer science, is lost.
- **–** Access from outside of the department network may be a problem because of the security risks which may originate from self-administered servers.

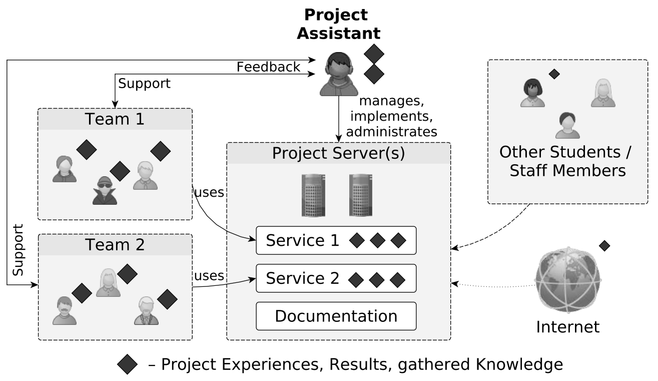


Fig. 3. Overview of Project Support Stage One – Central Project Support by Project Assistant

IV. Project Support Stage One – Central Project Support by Project Assistant

A. Description

In this stage, an employee who is exclusively dedicated to assist projects and who is called project assistant (PA) from now on is introduced. The PA is then able to perform recurring administrative tasks at a central place, relieving the project teams. Thus, she or he is also able to gather deeper knowledge about these tasks and document them accordingly, preserving all experiences.

The PA is responsible for installing and configuring all tools for the project support on central servers of the department. Therefore, the PA is likely to use automation for repetitive tasks, speeding up the process significantly. This stage is illustrated in Figure 3.

B. Implementation

The PA has to install and administer central project servers, which are at least accessible from the department network but preferably also from the Internet. Even if they do not offer a project portal to expose some projects to a greater audience, the access from the Internet will allow the project teams to use their project tools from everywhere. The PA will probably install and configure commonly used tools like version control management software and the required infrastructure for them, so that project teams can be provided with accounts to use the services quickly. The PA also has to keep the project servers, their services and tools up to date and ensure their safety by implementing an automated backup strategy.

Since the PA is the central person for project support, she or he will gather knowledge about CSCW, experiences with available tools and feedback from the users of the project servers over time and should refine all this information and make it available to the users, so that it is not lost like in the previous stage. Therefore, the PA can also assist project teams as a consultant and should offer training courses, so that students can easily learn how to use the supportive tools.

C. Assessment

- **+** Unburdens the project teams who do not have to perform administrative tasks themselves any more and therefore are able to use a far more sophisticated project support environment. This also makes it more attractive for students to try new tools and evaluate their value for their projects, because they do not have the overhead to install and administer them and therefore encourages learning. Additionally, more complex projects are possible since the teams can concentrate on their projects.
- **+** Repetitive tasks can be automated by the PA.
- **+** Services which are administered by the PA are more likely to be accessible also outside of the department network.
- **+** Furthermore, the PA can install special or new tools if project teams need them and document experiences with them, also offering them to other teams if applicable.
- **+** The chance to have sustainable projects increases, because project data is stored at a central well-known place and there is at least one person, the PA, who has an overview of all projects.
- **–** The overhead to manage all the projects increases rapidly with their number and may soon exceed the PA's time. The manual administration of different tools, even with the help of scripts, may not be feasible over time. Often user accounts have to be configured for every tool separately and updating the tools manually is very time consuming.
- **–** Without a central management software, such a system tends to become hard to manage for its users. It is likely that even if the project data of a specific project still exists in the system, it is hard or impossible to find and use or even be aware of it. Therefore, the sustainability and chance to reuse projects is compromised.

It has to be mentioned that even in a stage 1 project environment, it can be sensible to provide special projects with stage 0 support if the project characteristics will profit by it. For example, a technically high experienced team which needs very special supporting software may benefit from dedicated hardware.

V. Project Support Stage Two – Central Management Software with User Management and User Interface

A. Description

In this stage, a central management software is introduced to compensate the disadvantages of the previous stage. This management software also controls a central user database which should be used by as many tools as possible to avoid multiple user accounts for a single user for different tools or projects. The management of the tools is automated by the management software. The functions of the management software are accessible through a user interface (preferably a web interface), so they can be directly used by the end-users and normally do not have to be administered by the PA. In this way, the users are able to manage the most frequently used functions of the system themselves through the user interface and only have to use the help of the PA if they have special requirements or problems. Additionally, the central management software can ideally take over all administrative tasks for common tools.

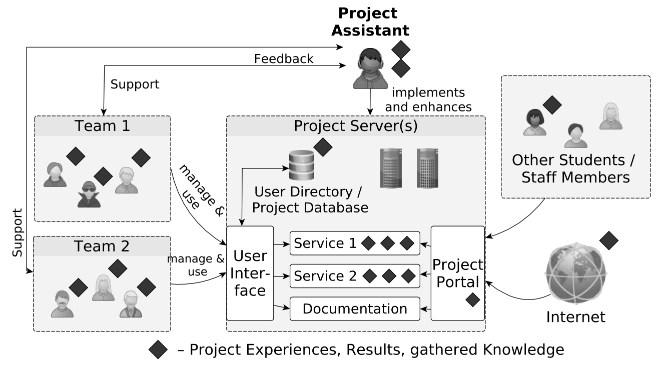


Fig. 4. Overview of Project Support Stage Two – Central Management Software with User Management and User Interface

B. Implementation

Since this stage is the last one we have fully implemented, we will have a closer look at its implementation.

1) Implementation Process: One iteration in the implementation process consists of the following phases:

- *Selection of Appropriate Tools.* Since it is very unlikely that a single tool can satisfy all requirements, a set of appropriate tools for each category has to be chosen using the results of the requirement analysis. Therefore, available tools have to be found, categorized, evaluated and rated, which by itself can be a sophisticated process. In some situations the only reliable method to measure the user acceptance of a tool is to evaluate it in real projects. This especially applies for relatively new tools.
- *Implementation or Improvement of Central User Interface.* The set of tools has to be integrated into an easy to use user interface which offers access to the tools while hiding all administrative details. The user interface has to automate all administrative tasks and maximize the potential of the different tools by integrating them in a sensible, clearly arranged way, improving their interaction and offering additional synergetic social and awareness functions. It therefore has to:
  - – Manage a central user database, where new users can autonomously create accounts which enable them to create projects and manage their teams.
  - – Offer the possibility to instantiate tools for projects. These instantiated tools have to use the central user database and preferably should be accessible via a single sign-on to the system, so the users do not have to supply their credentials multiple times.
  - – Support project life cycles. Since many projects will only last for a few weeks or months, the system has to deal with a lot of projects which lose their maintainers and teams. It therefore should be possible that

TABLE III
REQUIREMENTS AND SELECTED TOOLS

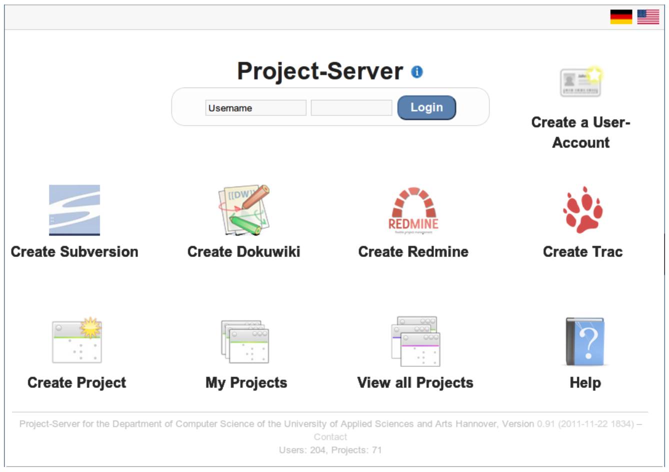
Requirement	*Selected Tool*
User and Project Management / Portal	Apache2, PHP-application MySQL-DB, LDAP
Version Control System	Subversion, Git
Knowledge Management	Wikis: Dokuwiki, Mediawiki CMS: Joomla; Blog: Wordpress
Project Management	Redmine, Trac
Communication Systems	Email
Tracker	Mantis, Bugzilla
Artifact Repositories	Sonatype Nexus
Continuous Integration Server	Jenkins
System Health Monitoring	Munin

an orphaned project can be resumed and continued by other users.
  - Offer additional synergetic functions like a project portal, feedback mechanisms, notifications, commenting and rating functions for projects and artifacts to increase awareness and interaction between users.
- *Awareness & Training.* Beyond increasing the awareness between users who actually use the system and other users, projects and artifacts through the aforementioned mechanisms, it is also very important to increase the awareness of new users to the system. The implementation can only be a success if new users are introduced to the system, motivated to use it and trained so that they are able to assess the value of its functions and to fully utilize them in their projects. The usage of supportive functions should be a part of the curriculum, since it is an important part of current software engineering practices.
- *Gather Feedback & Usage Statistics.* To constantly improve the system, it is important to gather and process direct and indirect user feedback. Direct feedback is acquired by asking users about their experiences, likes and dislikes. Indirect feedback by collecting anonymous usage statistics to investigate how the services are used.

For our concrete implementation, two servers with Ubuntu Linux are used. Table III shows the current selection of supportive tools. In the course of the implementation, two principles have been followed:

- Open-source software is used so the system is free software as well and therefore, available for everyone.
- Self-management of the system is meant to minimize the administrative effort and to let the users decide which functions they want to use on their own. Of course, it is essential that the users are educated appropriately and know how to deal with the different functions.

2) User Interface: The administration of users, projects and supportive functions is the core of the system. With their school's email address, users are able to generate accounts on their own using a web interface. Afterwards, they can create new projects, form teams and activate supporting functions for their projects on their own. The web interface and the functions behind it have been realized with the use of a PHP application. PHP was used for its fast prototyping qualities, to respond to changing requests quickly. At first, user data had been stored in a MySQL database, but meanwhile has been switched to an LDAP directory. Using LDAP has the advantage that many tools already have an interface to LDAP directories and are able to authenticate users through them. See Figure 5 for the welcome-page of the user interface.



Project-Server for the Department of Computer Science of the University of Applied Sciences and Arts Hannover, Version 0.91 (2011-11-22 1834) – Contact
Users: 204, Projects: 71

Fig. 5. Project Server Homepage

3) Evaluation: The following short evaluation shows usage data from March 2011 to July 2012 for this stage, although many projects (about 120) which have to be migrated to the new system have been created on the server before. As can be seen in Figure 6, after the introduction of the system the number of users in the new user management system at first increased only slowly. But then many users registered in October, when new projects were started in courses and the system was offered as a voluntary choice for the students. This shows how important it is to integrate a supportive system into the teaching itself. In this period, about 50 projects have been created. Figure 7 shows the number of subversion and git commits of these projects as well as the number for old projects.

C. Assessment

+ For the first time it is now possible to offer sustainable, accessible and perceptible projects through the central user interface and the gathered user and project data from the central management software. With this data, a project portal can easily be generated as a part of the user interface.

+ The high degree of automation allows a system which is able to scale to high numbers of projects and users.

– The huge amount of user and project data which develops over time may cause a loss of overview. As soon as a user cannot find information easily, the sustainability of all projects in the system is at risk. Therefore, the system has to offer functions to search through all data,

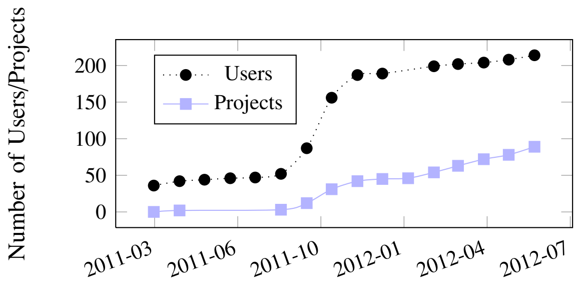


Fig. 6. User and Project Development in Stage Two

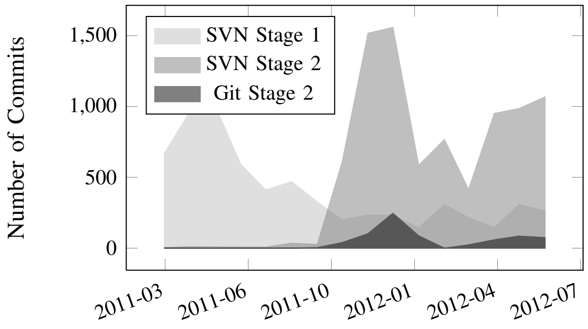


Fig. 7. Subversion- and Git-Commits in Stage One and Two

rate, comment and tag data and even to rate users and their experiences themselves, so that their ratings can be weighted. There is a great need for social functions.

– The development and maintenance efforts to realize the central management software and user interface are very high. A single PA is likely not able to implement and maintain such a system.

VI. Project Support Stage Three – Central User Database and Integration of Functions into Established Framework

A. Description

The aim of stage three tries is to compensate the high development and maintenance efforts of stage two, which exceeded the possibilities of our PA, while preserving its benefits. For that it uses an existing software as the framework of the management software component. These tools are integrated as modules or plugins into the framework where applicable. We are currently in the implementation process of this stage. The structure of this stage is very similar to that of the previous stage shown in Figure 4, only that the user interface and project portal functions are now offered by the integration platform and all services are modules of this platform.

B. Implementation

As an integration platform, Redmine has been chosen. Redmine is an open-source project management software implemented with Ruby on Rails. It already offers many functions, which would otherwise have to be implemented and maintained independently for a central management component as in stage two. These functions include: User and project management functions (where an existing LDAP-directory can also be used as a user-database) with role-based access control, a simple project portal, a issue tracking system, version control system integration, wikis, forums, feeds, email notifications, news, documents, file management, calendar and Gantt charts. Additionally, Redmine is already structured modularly, so that new functions can be integrated as modules or plugins. Every project can also choose which functions it wants to use and therefore is not burdened with unused functions. Admittedly, some functions are missing or need reworking for this use. For example, Redmine does not create Subversion or Git repositories. It can only use existing ones. Therefore this functionality has to be added.

C. Assessment

+ Comprises all benefits from the previous stage.
+ Much of the burden of the implementation is relieved by using Redmine as an integration platform for tools, because it already offers many functions and can be extended easily.
+ Plugins which are developed for Redmine may also be used by everyone else.
– If direct changes to the codebase of Redmine are made, they have to be applied to every new version of Redmine.

VII. Project Support Stage Four – Federation of Multiple Project Environments

A. Description

Ideally project support should not end at institutional borders. Quite the contrary, often the most interesting projects are projects which form their teams beyond institution boundaries. Therefore a system which allows users from different institutions to collaborate easily on projects would be desirable. This stage will introduce federation mechanisms to join multiple project support environments together. This enables their users to collaborate beyond institutional borders, but also respects the demand of each institution to administer their project support system themselves, internally using whatever system suites them best. In this way they are able to respond to special needs of their institution.

B. Implementation

Common interfaces of project environments have to be defined, so that project environments of different institutions can be joined together in a federation. User directories based on LDAP already support federation. Only the project metadata has to be shared between the environments through a well defined interface. For example Redmine already exposes its functions through a REST interface, upon which the federation could be built. But generally every institution could implement their own system, and could be part of the federation as long as it also implements the defined interfaces. This could even be a way to integrate external services like GitHub if they support the interface or a proxy is used.

C. Assessment

+ Such an approach would finally enable different institutions to work together easily, so they are able to jointly reach far higher aims with their projects but also preserving their own project environment domains.

VIII. Project Support Alternatives – Use of Hosting Services

An alternative to administer a project support environment is to use services which offer this functionality partly for free on the Internet (like SourceForge, GitHub and others). This may be the perfect solution for many projects because it relieves the institution from all administrative burdens. But this approach also imposes some restriction on the projects and the project support. Most free services require the projects to be open source projects and only offered tools can be used. Finally, institutional project support environments have the special chance to form a social network in the institution and benefit from this communities.

IX. Conclusion

The aim of this paper is to investigate ways of how to increase the value of computer science projects especially in the field of software engineering, research and teaching to avoid projects which are only carried out for their own sake and without a further value for the community. Most of these ways are based upon a higher visibility, accessibility and sustainability of the project and its results throughout the whole project lifetime and beyond it. This can be the key to more awareness, feedback and interaction with a project.

Students can gain the chance to learn from carried out and running projects. Former projects can be accessed more easily and it can be built on their results to create much greater values in the long range. This has already been done in our department, since there are projects which reuse results from former ones. Admittedly, the so far described functions and implementations of this paper will not be enough. It is not enough to implement the described infrastructure for projects: The infrastructure sure will provide the possibility to manage projects and easily access, edit and update their data in their lifetime and also beyond it, making it possible to reuse their results. But it has to be considered that the multitude of projects, which will eventually be created over time, will complicate a sensible use of their results without further actions. Therefore, the system has to be extended with proper mechanisms to categorize, tag, search for and browse, rate, review and comment projects and even their artifacts weighed by the users' reputation. This could also lead to an increased interaction from which the quality of the projects can benefit.

This is also a prerequisite for students to learn from other projects. They have to have a possibility to judge which projects and artifacts can be used as remarkable examples for e.g. good code quality, well-written documentation, intelligent framework use, or the use of special technologies. Ratings and comments from experienced users can be of inestimable value in this case. A student of architecture who did not examine excellent former examples of architecture in his or her studies is not imaginable. Nevertheless, many software engineering students – future software architects – often do not investigate excellent works in their studies. This situation could be improved with such a system and the possibility to learn from each other could be encouraged. The aim of such a development is a virtual community whose members benefit mutually from their work, creating common values.

References

[1] T. Gross and M. Koch, *Computer-Supported Cooperative Work.* Oldenbourg, 2007.

[2] K. L. Reid and G. V. Wilson, "DrProject: a software project management portal to meet educational needs," in *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education.* New York, NY, USA: ACM, 2007, pp. 317–321.

[3] C. Herrmann, T. Kurpick, and B. Rumpe, "Sselab: A plug-in-based framework for web-based project portals," in *Developing Tools as Plug-ins (TOPI), 2012 2nd Workshop on*, june 2012, pp. 61 –66.

[4] A. Majumdar and S. Krishna, "Social computing implications for technology usage and team interactions in virtual teams," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on*, oct. 2011, pp. 443 –450.

[5] M. Schaal and Y. Eren, "Dynamics of commitment and contribution quality in collaborative communities," in *Collaborative Computing: Networking, Applications and Worksharing, 2007. CollaborateCom 2007. International Conference on*, nov. 2007, pp. 294 –298.

[6] K. Swigger, R. Brazile, B. Harrington, X. Peng, and F. Alpaslan, "Teaching students how to work in global software development environments," in *Collaborative Computing: Networking, Applications and Worksharing, 2006. CollaborateCom 2006. International Conference on*, nov. 2006, pp. 1 –7.

[7] R. Keith Stanfill and Ethan I. Blackwelder, "Adapting Lightweight Source Control and Project Management Software for Use by Multidisciplinary Product Design Teams," in *Proceedings of the National Capstone Design Conference 2010*, 2010.

[8] A. Radermacher, A. Helsene, and D. Knudson, "Improving Capstone Courses with Content Management Systems and Virtualization," in *Proceedings of the National Capstone Design Conference 2010*, 2010, online: http://capstoneconf.org/resources/2010%20Proceedings/index.html.

[9] S. A. Munson, "Motivating and enabling organizational memory with a workgroup wiki," in *Proceedings of the 4th International Symposium on Wikis*, ser. WikiSym '08. New York, NY, USA: ACM, 2008, pp. 18:1–18:5. [Online]. Available: http://doi.acm.org/10.1145/1822258.1822283

[10] J. Grudin and E. S. Poole, "Wikis at work: success factors and challenges for sustainability of enterprise Wikis," in *Proceedings of the 6th International Symposium on Wikis and Open Collaboration*, ser. WikiSym '10. New York, NY, USA: ACM, 2010, pp. 5:1–5:8. [Online]. Available: http://doi.acm.org/10.1145/1832772.1832780

[11] J. T. Langton, T. J. Hickey, and R. Alterman, "Integrating tools and resources: a case study in building educational groupware for collaborative programming," *J. Comput. Small Coll.*, vol. 19, no. 5, pp. 140–153, 2004.

[12] I. Giannoukos, I. Lykourentzou, G. Mpardis, V. Nikolopoulos, V. Loumos, and E. Kayafas, "Collaborative e-learning environments enhanced by wiki technologies," in *PETRA '08: Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments.* New York, NY, USA: ACM, 2008, pp. 1–5.

[13] D. Kadenbach and C. Kleiner, "Benefits and Challenges of Using Collaborative Development Environments with Social Software in Higher Computer Science Education," in *Proceedings of the 3d International Conference on Online Communities and Social Computing: Held as Part of HCI International 2009*, ser. OCSC '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 479–487.

[14] ——, "Recent Trends in Software Support for Online Communities for Teaching and Research Projects in Higher Education," in *Proceedings of the 4th International Conference on Online Communities and Social Computing*, ser. OCSC'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 50–59.