KOI SCHOOL
Towards the Next Level of Communication, Organization and Integration in Education

Jonas Schulte, Reinhard Keil, Dominik Klaholt and Jannis Sauer
Heinz-Nixdorf-Institute, University of Paderborn, Fuerstenallee 11, 33102 Paderborn, Germany

Keywords: CSCW, Education, Integration, Knowledge, Organization, School, SOA.

Abstract: Information technology plays a central role in early school education of pupils, since the information society is continuously changing and developing. Facebook, Twitter, iPhone, Flickr, blogs, YouTube, and Google Earth are just some of the new technologies that bombard us from all directions since the genesis of the Web 2.0. Although there is a broad offer of educational software, yet an integrated solution for continuous support does not exist. In this paper, we present KOI, a collaborative school management system, that combines typical administration tasks with document management and social networking features. In our opinion, modern school software has to comprise both, educational software and those new technologies that are particularly popular among pupils. Thereby, private interests (e.g. social networking or instant messaging) and those functions supporting the schooling can be combined.

1 INTRODUCTION

An increasingly cross-linked digital world changed the society and especially the young generation that grew up with smartphones, the Web 2.0, and a mass of interactive media in everyday life. The MacArthur Foundation launched a five-year, $50 million digital media and learning initiative in 2006 to help determine how digital technologies are changing the way young people learn, play, socialize, and participate in civic life. Especially the increasing participation of young people in online worlds (Kafai et al., 2007), and the read/write culture of Web 2.0 is the subject of ongoing discussion and debate in educational circles (Ito et al., 2008; Jenkins, 2008). The willingness of young people to participate as prolific consumers and producers of digital content is high as never before. For example, the Pew Internet & American Life project identified in a study from 2005 that more than one-half of all teens have created media content, and about one-third of all teens who use the Internet have shared content they produced (Lenhart, 2005), as videos on YouTube or pictures via Flickr. Young people change more and more to participatory cultures where media, and in particular visual and social media, play an increasingly important role. Moreover, a shift in their relationship to the media can be recognized; from being an audience and users to becoming participants and creators as well (Svoen, 2007).

For this participatory cultures video and music mashups and remixes of existing content are the way of profiling and personalizing the new digital world. However, beside the desire to emphasize themselves, young people act more collaboratively and cooperatively than before. The fast growing number of blogs, forums, and newsgroups points up these changes. Furthermore, most Web 2.0 platforms require a so-called critical mass to become successful. In our opinion today's education has to include bit by bit the range and variety of internet-based tools, platforms, and practices adopted by young learners. Modern school software must not be an isolated application for administration, but an open infrastructure allowing the integration and adaption of those tools and platforms the pupils use to support technology-mediated learning in schools. The success of Apple's iTunes U project, with more than 600 participating universities and more than 250,000 lectures, presentations, videos, readings, and podcasts from all over the world, make clear that an easy online access to teaching materials is desirable (Rugg, 2009).

Nevertheless, fancy Web 2.0 tools did not find the way in everyday school life, even though these technologies have the potential to facilitate teachers

Schulte J., Keil R., Klaholt D. and Sauer J..
KOI SCHOOL - Towards the Next Level of Communication, Organization and Integration in Education.
DOI: 10.5220/0003469800430052
In *Proceedings of the 13th International Conference on Enterprise Information Systems* (ICEIS-2011), pages 43-52
ISBN: 978-989-8425-56-0

in performing their daily work. There are numerous possibilities for the teacher to interact with the pupils and a systematic and efficient storage and retrieval of their students' class materials using web-based school management tools and Web 2.0 technologies. Current school IT-infrastructures mainly focus on educational software to support solely the training of pupils. Moreover, there are cumbersome administrative task a teacher is often faced to especially when he or she wants to differ from the original plan. An example is an unheralded room reservation for audio-visual display. First, the teacher has to check which room is equipped with the necessary hardware. Second, a reservation has to be done and finally the pupils has to be informed about the room change.

In this paper we work out the requirements on a modern IT platform for early school education. Especially, the social changes and technical evolutions, mentioned above, have to be considered when planning a new system to support processes in education. The focus is on setting up a platform that supports teachers in doing administrative tasks and in addition allows pupils to interact and discuss in their usual manner. We present KOI school, a web-based application supporting communication, organization, and integration tasks in everyday school life. The implementation of KOI bases on WasabiBeans, a framework for setting up collaborative learning and working environments. Another strength of WasabiBeans is to act as an integration platform. Thanks to the microkernel architecture and flexible interfaces it is quite easy to couple different applications with the framework working as an integration layer.

This paper is structured as follows. We start with a research about typical e-learning systems in section *Related Work* and follow up with a requirements analysis in section *Knowledge Management in Everyday School Life*. Based on the requirement analysis we present our results and particularly the KOI school application. We close the paper with a discussion about our results and possible points of contact for future work.

2 RELATED WORK

On the one hand there are a few open source software solutions for cooperative working and data exchange like wiki systems (MediaWiki, Foswiki, etc.), e-learning platforms like Moodle, and also concepts where groupware goes to school by adapting BSCW to the classroom. Wikis are one of many Web 2.0 components that can be used to enhance the learning process (Stahl, 2004; Ebersbach et al., 2008). A wiki is a web communication and collaboration tool that can be used to engage students in learning with others within a collaborative environment (Parker and Chao, 2007). Wikis are fully editable websites with options to read or add content (Augar et al., 2004). Moodle is a free and open-source e-learning software platform, extendable with serveral modules for text editing, for document management, wikis, and communication. All these systems support a multi user environment with cooperative data collection or cooperative data exchange in terms of documents, images, papers, links etc. and a few simple features to communicate and interact together like message boards or email systems (Dougiamas and Taylor, 2003).

On the other hand there are indeed no open source, but plenty of commercial software solutions to manage the everyday school life. This means these software solutions manage room planning, appointments e.g. for exams, class schedules or pupils and their address data or marks. However, a solution to fill the gap between those applications for supporting cooperation among pupils and teachers and those for performing administrative tasks is missing. For example a global system, which supports the teacher to book or to change a classroom, marks this classroom as occupied for other users and informs the pupils about the classroom change. Specialized and isolated applications are suitable for single task, yet an execution of continuous workflows as required for a modern school IT infrastructure is not possible.

3 KNOWLEDGE MANAGEMENT IN EVERYDAY SCHOOL LIFE

In several countries pupils are grouped as classes in primary school and high school. The concept behind doing so is the Knowledge Building Community (KBC), broadly defined as an environment that encourages pupils into knowledge creating culture in addition to their competencies in certain subjects (Bereiter and Scardamalia, 2003). Usually these classes have their own classroom. Only lessons, like physics or sports that require special equipment take place in different rooms. Beside compulsory subjects the pupils may choose between optional subjects. Accordingly, those pupils who attend a certain optional subject may not belong to the same class. Thus, new groups for optional subjects are formed. Due to this fact the flexible composition of participants to groups is an important requirement on school software.

3.1 Virtualization of Classrooms

The establishment of a digital reproduction of the real school environment helps pupils to orient themselves. Hence, our demand is to combine the idea of a room-based virtual world with basic document management and social networking functionalities to support processes in everyday school life. Rooms function not only as social meeting places and centers of virtual learning community, but also as a collective external memory, providing media functions as a basis for cooperative learning (Hampel and Keil, 2001). Particularly with regard to cooperative work environments, virtual knowledge spaces facilitate users to aware each other as well as to structure documents freely and follow up changes via a notification mechanism based on events.

The basic concept of virtual knowledge spaces includes *room*, *container*, and *document* objects to structure data. In addition every of these so-called data objects can be annotated with *attribute* objects. Attributes are supposed to be real objects rather than simple strings. Managing awareness in virtual knowledge spaces is of particular importance for the user interaction, teamwork, as well as computer-aided learning and collaboration (Prinz, 2001; Prinz et al., 2008). Therefore, every room contains a list of users that are present in the current room. Finally rooms can contain ”exits”, links to other rooms, but they may also contain rooms themselves, so that it is possible to create hierarchical structures of virtual knowledge spaces.

The idea is to create a digital world that allows users to meet, interact, and learn with each other. Thereby, a school represents a network of various virtual knowledge spaces that makes it possible to appreciate the actions of cooperative partners. Such an environment that allows users to be aware of the actions of cooperative partners is the prerequisite for unambiguous interaction and collaboration. For example the cognition of modifications on certain documents, attendees within the same virtual space, and the participants of a chat discussion are important events to support the cooperative working process.

3.2 WasabiBeans – A Service-oriented Framework

WasabiBeans is a service-oriented framework that supports the assembling of cooperative work and learning environments (Schulte et al., 2008). The basic concept is a micro kernel architecture with different layers and so-called modules to implement new functions. Figure 1 gives an overview of the framework’s architecture. The core model (internal core) implements the concept of virtual knowledge spaces as introduced above in section *Virtualization of Classrooms*. The persistency layer bases on the JPA[1] and Hibernate for Object-relational mapping (ORM). Thereby, the mapping of the data model to a relational database system is realized, beyond an easy replacement of the concrete database system is possible. WasabiBeans offers different interfaces for interaction. Firstly, local services, these services are only accessible from modules and those applications that are deployed on the same JBoss Application Server (JBoss AS) and run within the same Java Virtual Machine (JVM) as the WasabiBeans framework itself. Secondly, remote services, these services are accessible via web services and Remote Method Invocation (RMI) as shown in figure 1.

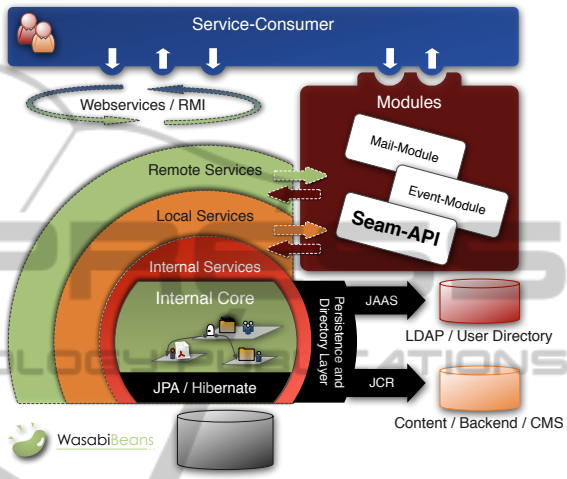

Figure 1: Architecture Overview of the WasabiBeans Framework.

In particular cooperative work and learning environments are characterized by high user interaction. Users may hold the membership in different groups and exchange documents among each other intensely. The high interaction and the complex allocation to groups requires a flexible and highly adjustable right management. WasabiBeans offers a right management with a variety of features that support these requirements. As an example WasabiBeans implements the concept of right inheritance to define permissions rapidly. For a detailed explanation of WasabiBeans’ right management we refer to (Schulte et al., 2009).

[1]Java Persistence API, online available: http://java.sun.com/developer/technicalArticles/J2EE/jpa/

4 RESULTS

The previous sections motivated our goal to create a school-management system, that combines typical administration tasks with document management and social networking features. Further we suggested that the concept of virtual knowledge spaces is well-suited to help reach that goal. As a consequence the WasabiBeans-Framework being an implementation of the concept of virtual knowledge spaces is used as the base of our intended school management system. In this section we will present the KOI school-management systems that combines typical administration tasks with document management and social networking features. Further we suggested that the concept of virtual knowledge spaces is well-suited to help reach that goal. As a consequence the WasabiBeans framework being an implementation of the concept of virtual knowledge spaces is used as the base of our intended school-management system. In this section we will present the current state of our progress to create the intended school-management system as a web application named KOI using state of the art technology like the Seam framework. First we will give information about the Seam API that enables a comfortable use of the WasabiBeans framework within a Seam framework based web application. Then we will present KOI and its main features itself and point out how these features are based upon the WasabiBeans framework. The last paragraph of this section deals with performance issues of the multi-user web application KOI.

4.1 The Seam API – Rapid Seam-based Development

The KOI school management system, described in the next section in detail, is a web-based Seam application. JBoss Seam is also known as the Seam framework or simply Seam. Seam is a web application framework and was invented by Gavin King, a lead developer of JBoss. It simplifies the interaction between view and business logic. The view is represented by JavaServer Faces (JSF) an in terms of the KOI system with Facelets, an alternative view-handler technology for the JSF framework replacing the JavaServer Pages (JSP). The business logic is represented by Enterprise JavaBeans 3.0 (EJB3). The framework also expands the concept of contexts. Each Seam component exist within a context. To afford more flexibility with respect to life cycle definition of web applications, Seam extends the stateful SessionBeans of EJB3 by providing seven different scope types. These scope types enable the developer to control the runtime of a SessionBean from a simple request via a conversation or a session through to complete runtime of application. This offers an efficient way to store data for more then one request. For example the default Seam context, a conversation, can span multiple pages and usually spans the whole business flow, from start to finish. Another example is the session context, which captures all actions of a user until he logs out or gets a session timeout (e.g. when closing the browser); even though using the back-button of the browser. Beside the possibility to configure Seam components using annotations, as known from EJB3, Seam introduces the concept of *bijection*. A concept, taken from Spring's *dependency injection* feature where objects can be in-jected from or out-jected to assigned variables using the `@In` respectively the `@Out` annotation. This is a simple solution to share data between Seam components (Yuan et al., 2009).

On the one hand the data storage and exchange within the Seam framework is realized, as it based on EJB3, in an usual object-oriented manner. On the other hand WasabiBeans is consistently designed as a service-oriented architecture (SOA), hence these two different approaches of the frameworks do not match to each other. More precisely, the design of WasabiBeans does not offer objects to overlying applications, instead it offers services whose output and input data are Data Transfer Objects (DTOs). To support an efficient way of data exchange between Seam and WasabiBeans, it was necessary to develop a solution that provides the WasabiBeans' objects in an object-oriented way for further processing within the Seam framework. The implementation of such an application programming interface (API) can be realized either within the application itself (in this case the KOI system) or as a component of the WasabiBeans framework. For a better reutilization of the API it was integrated into the WasabiBeans framework, since other applications based on JBoss Seam can be implemented in the object-oriented manner too. Figure 1 shows the classification of the developed application programming interface called Seam API between JBoss Seam and the WasabiBeans framework.

Another relevant aspect for the development of an API between JBoss Seam and the WasabiBeans framework is resulting from the functionality of KOI. The WasabiBeans framework offers a set of structures like documents, attributes, links etc. To project a message or an appointment in KOI by the given structures of the WasabiBeans framework, a relation of this structures is necessary. A message for example is represented by a document, its properties like subject or

sender are represented by attributes, which are connected with the document. The distribution of messages is realized by links. To project these complex structures, Seam API was extended with these structures. With it Seam API is not only an object-oriented mapping of WasabiBeans framework, but also an extension of the structures of the WasabiBeans framework.

4.2 KOI – School Management System

KOI is a web application for schools that offers administrative, organizational, and social networking functions for the students, the teachers, and the general staff. Though it still has to be considered as a prototype, it already meets many of the requirements presented in section *Knowledge Management in Everyday School Life*. As stated in the previous section KOI is implemented with JBoss Seam and uses the WasabiBeans framework via our Seam API. In this section we will present concrete functions KOI provides and we will show how KOI makes explicit use of the concept of virtual knowledge spaces.

4.2.1 Entering the User's Room

As described in section *The Seam API - Rapid Seam-based Development*, the view-handler, which is an interface to the user of the web application, consists of xhtml-files which can be rendered as web-pages and thus can be displayed in a user's browser. Each virtual knowledge space of KOI is associated with one or more of these xhtml-files. The idea is that when a user browses through the web application he moves through virtual knowledge spaces. Upon entering a virtual knowledge space one of the associated xhtml-files is rendered and displayed to the user. A user can see a virtual knowledge space from different perspectives, so more than one xhtml-file can be associated with a virtual knowledge space. For instance in KOI each user has his own room that represents the user's personal area where he can store his data like personal information, documents or appointments. Now when a user enters his own room he might have a look at his documents or at the list of his friends - his documents on the one hand and the list of his friends on the other hand are two different perspectives on his room. When a user enters another user's room, then that user's profile page is displayed - yet again another perspective of a user's room. Of course a user is able to specify what other users are allowed to see when they enter his room.

4.2.2 Breadcrumb-trail and Multiple Tabs

To help the user orient himself within the application KOI provides a breadcrumb-trail. The breadcrumb-trail reflects that the user moves through virtual knowledge spaces, that he can have different perspectives on one place, and that he always resides in one place. Furthermore KOI supports independent multi-tabbed browsing. So, in contrast to a classical web application, a user can browse through KOI within multiple tabs independently at the same time (and may reside in a different virtual knowledge space in each tab). As long as no persistent data is changed, the tabs do not influence each other (e.g. each tab has its own breadcrumb-trail).

4.2.3 Virtual and Real Spaces

Virtual spaces like the rooms of users must not be confused with real spaces like the classrooms of a school that are also represented within KOI. KOI enables administrative users to manage information about a real classroom, e.g. the name, the floor, and especially the equipment of the classroom. This information is stored within a virtual room that represents the real classroom within KOI. The information stored about a real classroom can also include appointments at which the classroom is occupied by a class or a learning group.

4.2.4 Documents and Appointments

Each user can manage personal documents and appointments in his room. Personal documents are stored within a container (the document-container) which is located in the user's room. A user can create, edit, rename, delete, download and upload documents in the document-container. Furthermore a user is able to structure his documents by creating arbitrary many levels of sub-containers within the document-container - analogous to folders within a file-system. Personal appointments are also stored within a container (the appointment-container) which is located in the user's room. KOI allows to create, edit and delete two kinds of appointments: Appointments that take place exactly once (e.g. a doctor's appointment) and appointments that take place repeatedly (e.g. weekly classes or annual feasts). Internally an appointment is represented by a document that is extended with special attributes by the Seam API. An important feature of appointments within KOI is that they can be connected to documents and classrooms. So documents that are of importance for a specific appointment are perceived when looking up the appointment. Classrooms can be booked as meeting points for an

appointment if the user creating the appointment is appropriately authorized.

4.2.5 Groups

Users of KOI can be organized in groups, which are created and administered by authorized users like teachers (group-administrators). Each group has its own room, which represents the area exclusive to the group (analogous to a user's room). Within the area of a group, its members, documents, appointments, and sub-groups can be managed. Documents and Appointments are stored analogously to the way they are stored within a user's room. However, managing documents and appointments of a group has an additional level of complexity: multiple members of the group who access the documents and appointments. Therefore group-administrators can assign certain rights to each member of the group. Figure 2 shows the web page on which a group-administrator can manage the members of the group and can assign certain rights to these members by use of toggle-icons behind each user's name. So one member can be allowed to create appointments whereas another member can only read them. Corresponding general rights can be assigned concerning the documents of the group. Moreover there are even more fine grained rights possible for the documents (which can be set on the web-page for managing the documents of the group): For each document an administrative user can specify which member of the group may read or edit the document. The authorization mechanisms used by KOI are based upon the authorization mechanisms provided by the WasabiBeans framework (see (Schulte et al., 2009)). Another capability of a goup-administrator (who, of course, can also promote other members of the group to be a group-administrator) is to manage the subgroups of the group. Sub-groups again have their own room and their exclusive area provides the same functions as the exclusive area of the parent-group. So arbitrary many levels of subgroups can be created.

4.2.6 Asynchronous Communication

Users of KOI can be members of numerous groups and they can keep a list of friends. To further foster communities KOI supports both asynchronous and synchronous communication. The asynchronous communication is accomplished by a message service that provides each user with the components he knows from his regular email client: an inbox, a folder for sent messages, a trash basket, and an address book. Each of these components is represented by a container that is located in the user's room. The address book-container contains documents in which the information about contacts is stored. The inbox-, sent-messages-, and the trash-basket-container contain links to messages. A message is internally represented by a document that is extended with special attributes by the Seam API. Messages can be sent, replied to, forwarded, or deleted. An important detail of the concept behind the message service is based upon the above mentioned links to messages. In contrast to a classical email-system in which numerous copies of one message are created the KOI message service stores each message only once. Therefore messages are not directly stored within the inbox-, sent-messages, or trash-basket-containers. Instead there exists a special container for each user that contains all the messages the user creates (due to sending, replying or forwarding) and that is located outside the user's room. So the inbox-, sent-messages, and trash-basket-containers contain links that point to the messages in the special containers.

4.2.7 Synchronous Communication

The synchronous communication within KOI is accomplished by a chat. The chat allows a user to directly communicate with a friend or with the members of a group he belongs to. When two friends are chatting, they are able to invite other friends. So new contacts who are invited by others can be met and each conversation can be turned into having multiple participants. Moreover a user can participate in multiple conversations at the same time. Each conversation is shown in a separate tab. Furthermore, KOI allows you to start the conversation even though the desired conversation partner(s) is (are) still offline. Once they appear online, they will be able to recognize that someone is waiting for them to participate in a conversation. Another important feature of the chat is that already ended conversations can be resumed at a later time, so that references to an earlier discussion are simplified. As a consequence each conversation must be persisted by KOI. To achieve this the virtual knowledge space representing the overall chatroom contains a dialogue-container for each combination of users and for each group. These dialogue-containers contain a session-container for each conversation the combination of users or the group associated with the dialogue-container ever had. Finally the session-containers contain the exchanged chat-messages, which are internally represented by documents.

4.3 Performance Optimization

With the progress of the implementation of KOI, its performance was decreasing significantly. The cause

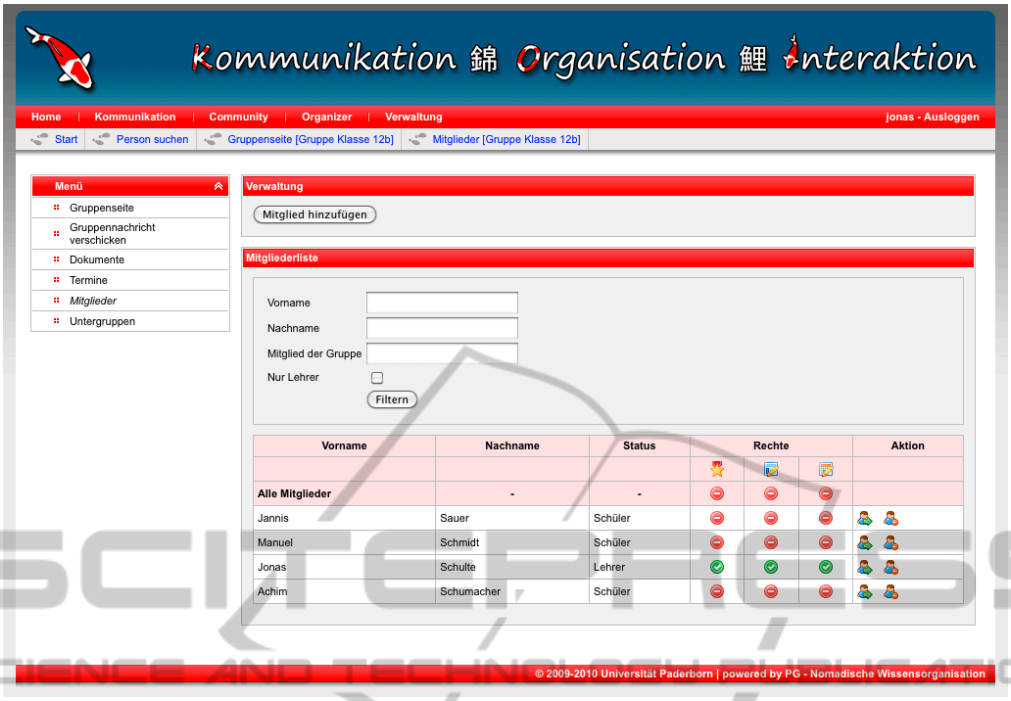

Figure 2: Group Management in KOI School.

of the decrease can be illustrated in particular by the example of users profile page. The users profile page at KOI provides information about a registered user in KOI like his name, address, hobbies, profile image etc.. If the users profile page is requested, the following happens: The users profile page facelet is requesting information in dependence of the requesting user at the corresponding Seam component called BackingBean. The Seam component is requesting at Seam API and Seam API calls each single service at WasabiBeans framework. For each information or property a service must be called at WasabiBeans framework. The complete workflow of getting information by a facelet from WasabiBeans framework is illustrated in Figure 3.

Furthermore it is possible while passing JSF lifecycle, that the facelet is rendering multiple times (Ed Burns, 2010). As a result of that previously invoked services of WasabiBeans framework are invoked once more. Hence it is possible that a simple facelet generates several hundred database requests by WasabiBeans framework. For example the users profile page generates over 200 object and method invocations. If users profile page is called more than 1000 database request are generated, because WasabiBeans framework does not only request information, but also checks weather authorization for requested information exist. Due to this it is possible that more than 10 seconds elapse, before the page is visible in the browser. This is definitively a exclusion criterion for a respectable web application that wants to be taken seriously.

To solve this problem, it is possible to use Seams concept of contexts with its seven scope types by storing data over more than one request. This can happen either directly by the BackingBeans for each facelet according to requirements or uniformly and centralized at one place. The decision was made on a uniformly and centralized approach which extends the Seam API by integrating a cache called Seam Cache.

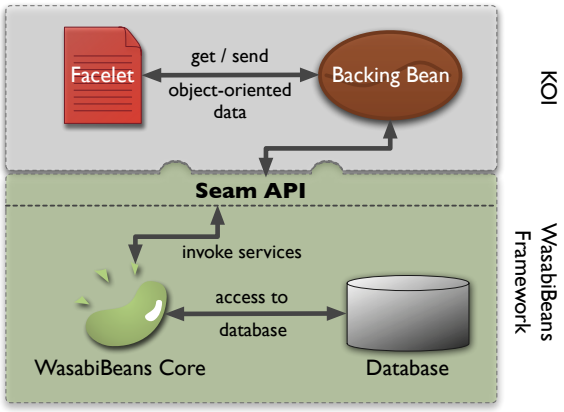

Figure 3: Workflow of getting information by a facelet from WasabiBeans Framework.

Requests by facelets over the BackingBeans do not longer invoke methods at Seam API in a direct way. The Seam Cache whether checks for each request the requested data are stored in the cache. If requested data are stored, the cache supplies the data. Thereby no access to a service of WasabiBeans framework takes place. Because of this there is no access to the database as well. The cache is realized by a Java HashMap with a key-value store. A key consists of following parameters:

- Username of requestor
- Name of the requested service
- Name of the invoked method of the service
- Commited parameters of the method

Depending on the invoked method, for instance a write operation, an invalidation of the corresponding cache entry occurs. Figure 4 shows the new data handling with Seam Cache by cache miss and cache hit. At point 1 in the Figure 4 KOI tries to request data for instance a document from the Seam API and at point 2 the Seam API checks if requested data exists in to the Seam Cache. At point 3 the Seam Cache answers the Seam API, that he requested data does not exist in to the Seam Cache at this time (cache miss). Therefore at point 4 the Seam API starts to request the data from the WasabiBeans Core and consequently from the underlying database. At point 5 the WasabiBeans Core replies the requested data back to the Seam API. Next at the same time the Seam API stores the data at point 6 in to the Seam Cache and sends data to KOI at point 7. If KOI requests the same data again the Seam API requests to Seam Cache, but don't have to request data from WasabiBeans Core and consequently from the underlying database, because the Seam Cache supplies the requested data to Seam API and Seam API to KOI (cache hit).
The Seam Cache solves especially the problem of multiple facelet rendering. For the example of the user profile page it means that the first invocation by facelet via BackingBean to Seam API and services of WasabiBeans framework stores all necessary data in the cache. In case of repeated rendering of the facelet, all data is loaded by cache and not again by invocation of services of the WasabiBeans framework. If a modification of the users profile page data occurs, for instance the address, only this cache entry becomes invalid. If for instance address is requested again, Seam API loads only this value into the cache again. On the basis of Seam Cache the performance of KOI could increase distinctly, so that no page of KOI loads longer than 2 seconds on an average.

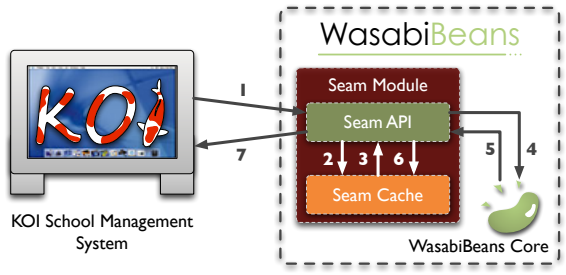

Figure 4: Workflow of the new data handling with Seam Cache (scenario cache miss).

4.4 Usage Scenarios in School

KOI's connection and interaction between several workflows simplifies everyday school life. In the following some exemplary usage scenarios in school, demonstrate the range of functions in KOI.

4.4.1 Classroom Planning

A teacher wants to show a movie at his lesson next day. He needs a classroom with some technical environment to show the movie. KOI enables him to search free and suitable classrooms with a beamer or a television and a minimum number of seats. After selection of a classroom he books it. But now it is necessary to inform his pupils about the changed classroom. No Problem, because the teacher has the possibility to inform his pupils by KOI's message system. He writes a message to his class, represented in KOI by a group, and every member of the group receives a message which informs about the changed classroom. Additionally it is possible to connect an appointment with classroom planning and backwards. That means the teacher has the possibility to create a group appointment and each member of this group, commonly the pupils in the class, gets an own appointment in his schedule.

4.4.2 Submit an Essay

If a teacher wants to enable online submissions of documents, essays or home work until a specific date, KOI supports him with its upload function and powerful right management given by WasabiBeans framework. It is possible to create a specific folder for online submissions at groups document management system and set an allowance to upload documents at this folder. If submission date expired the teacher removes the allowance to upload documents at this folder. An easier way to remove the allowance to upload documents at a specific date are time rights. The time rights with a start date and end date are able to

set an allowance between this interval, consequently the allowance does not exist after end date and no additional user interaction is necessary.

4.4.3 Scheduling

Scheduling especially the connection between group appointments is a helpful organization option in KOI. Appointments of a group are represented as individual appointments of a user. But if a group appointment is changed, the individual appointment of a user is changed too. Commonly a user is in several classes or courses represented in KOI by a group. Each class or course has a teacher which is able to set group appointments. If a teacher changes the appointment of an examination the individual user appointment of this examination is changed too. But the user has the option to modify his individual examination appointment in terms of a copy.

5 DISCUSSION

The WasabiBeans framework implemented as a service-oriented architecture supports the integration of external services, yet KOI school does not exceedingly benefit from this background. Future enhancements of KOI will make use of the underlying architecture. We are currently working on a Facebook connection that allows pupils to import their Facebook profile, rather than specifying and updating their user profile within KOI school application itself. The integration or establishing a connection to recent Web 2.0 services like MySpace or Flickr are imaginable and may be raise the popularity among young teenagers. In particular, turning KOI into a central access point that accomplishes various popular services meets exactly the idea of KOI to be a combination of popular Web features with educational software features. Thus, the acceptance of the system among all participants (teachers and pupils) may be increased.

The KOI system already stores information about real classrooms within the system. However, this connection between the virtual and the real world could be extended. For instance, the equipment of classrooms could be tagged with RFID chips to enable an automatic recognition of all relevant devices within a classroom. So the information stored within the virtual classrooms in KOI could automatically be updated, especially in the case when a device (e.g. a television) is moved from one room to another.

Staying with the idea of automatic recognition of the real world environment, the detection of physical devices makes different approaches possible. As presented in (Paulos and Goodman, 2004), each pupil may equipped with a Jabberwocky to detect the location of pupils with similar interests, lectures, etc. Furthermore, more and more pupils have a mobile device, that is equipped with bluetooth or WiFi interfaces allowing locate the current position of a pupil within the school. By recognizing the current location of a pupil, the devices may exchange automatically information about hobbies and interests in case the pupils are willing to participate and adjust the settings of their devices accordingly.

6 CONCLUSIONS

In this paper we presented KOI, a web-based application to support collaboration and administrative tasks in schooling at the same time. Our main contribution is that our approach allows an integration of manifold applications or services to build up a complex IT infrastructure without media disruption between the single systems. As a result a continuous assistance in schooling can be realized. Moreover, we identified the Web 2.0 developments to be a challenge for state of the art developments in schooling. The possibility to integrate these services and fancy applications is absolutely necessary to address the pupils wishes and their understanding of modern communication.

We started our research with the intent to offer a digital representation of a real school to support the end-users to get along with the digital system. For that reason we used the concept of virtual knowledge spaces and mapped actual classrooms or rather the physical structure of a school onto a digital data model. An important task was to enable the participants to be aware of each other, since awareness is the prerequisite for smooth collaboration and reduce misunderstandings. Therefore, the KOI school-management system offers numerous features to support the interaction among participants. An example is the chat component with its conversation history or the flexible message system of KOI. For our implementation we used the WasabiBeans framework, since its microkernel architecture allows an easy adjustment and the implementation of extensions to fulfill changing needs. First, we implemented a Seam API for WasabiBeans to allow straight forward implementations of Seam-based applications that build on the framework. A second result of our research is the KOI school management system that combines administrative tasks in schooling, as well as functions to support the communication among participants and the structuring of teaching materials. The practical usage of KOI revealed some performance problems

with the Seam API, thus we implemented a Seam cache to meet this disadvantage.

A unique feature of KOI school is the flexible group management. This allows not only administrators to create and organize groups, but also end-users, like pupils, to create and manage their private learning groups. In addition, users can be member of different groups and groups may contain other groups. Thereby a sub-group relation can be realized and self-administration as well as self-organization can be fostered.

To sum up, KOI school is an application for collaboration and offers numerous features that are necessary in everyday school life, like room booking, calendar management, and document management. At the same time it is very flexible and adjustable to suffice future requirements. The open architecture, strengthened by using the WasabiBeans framework, enables the integration of various Web 2.0 features and thus complies with the demand of the young learning generation.

REFERENCES

Augar, N., Raitman, R., and Zhou, W. (2004). Teaching and learning online with wikis. In Atkinson, R., McBeath, C., and Jonas-Dwyer, D., editors, *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference*, pages 95–104, Perth.

Bereiter, C. and Scardamalia, M. (2003). Learning to work creatively with knowledge. In *Powerful learning environments: Unraveling basic components and dimensions*, pages 55–68. Oxford, UK: Elsevier Science, E. De Corte and L. Verschaffel and N. Entwistle and J. van Merriënboer.

Dougiamas, M. and Taylor, P. C. (2003). Moodle: Using learning communities to create an open source course management system. *World Conference on Educational Multimedia, Hypermedia and Telecommunications*.

Ebersbach, A., Glaser, M., Heigl, R., and Warta, A. (2008). *Wiki: Kooperation im Web*. Springer, Berlin, 2. edition.

Ed Burns, Chris Schalk, N. G. (2010). *JavaServer Faces 2.0: The Complete Reference*. McGraw Hill.

Hampel, T. and Keil, R. (2001). steam: structuring information in team-distributed knowledge management in cooperative learning environments. In *Journal on Educational Resources in Computing (JERIC)*, volume 1, New York, NY, USA. ACM.

Ito, M., Horst, H., Bittanti, M., Boyd, D., Herr-Stephenson, B., Lange, P. G., Pascoe, C. J., and Robinson, L. (2008). Living and learning with new media: Summary of findings from the digital youth project. Technical report, John D. and Catherine T. MacArthur Foundation.

Jenkins, H. (2008). Confronting the challenges of participatory culture: Media education for the 21st century.

Kafai, Y. B., Fields, D. A., and Cook, M. (2007). Your second selves: Avatar designs and identity play in a teen virtual world. In *Proceedings of DIGRA 2007*.

Lenhart, A. (2005). Teen content creators and consumers. Technical report, Pew Research Center.

Parker, K. and Chao, J. (2007). Wiki as a teaching tool. *Interdisciplinary Journal of Knowledge and Learning Objects*, 3:57–72.

Paulos, E. and Goodman, E. (2004). The familiar stranger: anxiety, comfort, and play in public places. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 223–230, New York, NY, USA. ACM.

Prinz, W. (2001). Awareness. *CSCW-Kompendium: Lehr- und Handbuch zur computerunterstützten Gruppenarbeit, Springer, Heidelberg et al*.

Prinz, W., Hinrichs, E., and Kireyev, I. (2008). Anticipative awareness in a groupware system. In *Proceedings of the 8th International Conference on the Design of Cooperative Systems (COOP '08)*.

Rugg, B. M. (2009). Getting itunes u at ithaca college up and running! In Farally-Semerad, G., McRitchie, K. J., and Rugg, E., editors, *SIGUCCS*, pages 275–282. ACM.

Schulte, J., Döpke, I., Keil, R., Stark, K., and Eder, J. (2009). Enhanced security management for flexible and dynamic cooperative environments. In *Collaborative Computing: Networking, Applications and Worksharing, 2009. CollaborateCom 2009. 5th International Conference on*, pages 1–10.

Schulte, J., Hampel, T., Bopp, T., and Hinn, R. (2008). Wasabi beans – soa for collaborative learning and working systems. In *DEST '08: Proceedings of the Second IEEE International Conference on Digital Ecosystem and Technologies*, pages 177–183, Phitsanulok, Thailand. IEEE Computer Society.

Stahl, G. (2004). Groupware goes to school: adapting bscw to the classroom. *IJCAT*, 19(3-4):162–174.

Svoen, B. (2007). Consumers, participants, and creators: young people's diverse use of television and new media. *Comput. Entertain.*, 5(2):5.

Yuan, M. J., Orshalick, J., and Heute, T. (2009). *Seam Framework - Experience The Evolution of Java EE*. Prentice Hall.