PILS: Advanced Instant Messaging in e-Learning Based on an Open Implementation

Mirko Fetter, Tom Gross
Faculty of Media
Bauhaus-University Weimar
99423 Weimar, Germany
<firstname.lastname>(at)medien.uni-weimar.de

***Abstract*—In e-learning platforms the social interaction between students and teachers as well as among students is important for effective and efficient learning. Therefore, e-learning platforms need technological support for online presence and communication. In this paper we present the Presence in Learning Spaces (PILS) infrastructure, which provides advanced concepts for mutual presence information and seamless communication between online users based on an open implementation. Its novel concepts support course-specific mutual presence and communication, mutual communication awareness, and e-learning–specific online states. Its open implementation leverages remote control through course and user administration services, presence and communication control services, and data and inferencing services. The PILS infrastructure is part of the Adaptive Learning Spaces (ALS) platform, developed in the Adaptive Learning Spaces (ALS) EU-project.**

***Keywords*—Computer-Supported Cooperative Work; Instant Messaging; Distributed System; Open Implementation.**

I. INTRODUCTION

The social interaction between students and teachers as well as among students is important for effective and efficient learning. Therefore, e-learning platforms need technological support online communication [36].

Simply using separate communication tools parallel to the e-learning application does not account for the interwoven activities that are typical for user switching between their primary task, in our case learning, and their secondary task, in our case communication for mutual exchange and help [14]. Furthermore, learning is a highly focused task for the students and so—although the connection between the users is important and can be beneficial for their success—the cost of interruption is high [25]. And so disruption outweighs the benefits of mutual availability [20].

In this paper we present the Presence in Learning Spaces (PILS) infrastructure, which provides advanced concepts for mutual presence information and seamless communication between students and teachers and among students based on an open implementation. It can be used like existing instant messaging systems (e.g., ICQ [22], Skype [35])—with standard presence information and online text, audio, and video chat capabilities. It provides novel concepts for course-specific mutual presence and communication, mutual conversation awareness, and e-learning specific online states. Furthermore, it is based on an open implementation leveraging remote control through course and user administration services, presence and communication control services, and data and inferencing services.

The PILS infrastructure is part of the Adaptive Learning Spaces (ALS) platform, developed in the Adaptive Learning Spaces (ALS) EU-project. The ALS platform provides teachers and students with advanced support for learning management with adaptive delivery of hypermedia content, adaptation authoring, as well as adaptive presence and communication [2].

In the next section we present the concepts for the advanced instant messaging and the open implementation. We describe the implementation of PILS. We then illustrate the user interaction in a scenario. We provide an overview of related work. Finally, we draw conclusions.

II. CONCEPT

Learning spaces enable distributed students to develop knowledge together in a shared activity. In contrast to real-world learning where people physically meet in a classroom and discuss the material or can ask questions, learning spaces need to provide special mechanisms to support this information exchange. Stahl [36] emphasises the importance of collaboration in computer-supported cooperative learning—therefore, it should provide technology that stimulates and sustains interaction among students. Gutwin [19] argues that systems for collaborative learning should support awareness to foster the positive effects of group learning. This is where the concepts behind the PILS infrastructure bring to bear. PILS is designed to support awareness and to offer novel means of communication and interaction in distributed learning, as we explain in the following in more depth.

A. Advanced Instant Messaging

Providing users with awareness information is a key requisite for computer-supported cooperative work systems [17]. Knowing about the presence, the activities, and availability of peer users is positive for the cohesion and effectiveness of a group of learners as well as for the result of each individual. Awareness information can generate an understanding about what other users are currently doing [12], can help to estimate if it is a good time to establish communication [7], or foster the feeling of being connected and establish a shared sense of community [13]. So, PILS offers basic awareness about the presence of other users via online states.

Supporting asynchronous and synchronous communication in distance learning is beneficial for social presence, learning process and outcome, and the flexibility in terms of anytime and anywhere learning. This is particularly the case for learning techniques that go beyond pure memory and comprehension [6]. So, PILS offers dyadic text, audio, and video chat. As in learning scenarios teamwork is often appropriate, PILS also supports multi-user text chats among course members.

Communication has from a learners' perspective a trade-off between contacting fellow learners to go through the material together, and deeply concentrating on a learning task without. PILS introduces three central concepts for advanced instant messaging tailored to adaptive learning spaces in order to provide the user with means to deal with that trade-off;

The first concept are of *course-specific* online states and communication. Each user can be a member of one or more courses. For each course a learner is subscribed to in the learning platform, PILS provides a separate buddy list showing only the participants of this course. Accordingly, if users are subscribed to more than one course, they can switch between the different courses, to see their online fellow students for each course. A specific online state for each course gives the users the freedom to adapt their presence and availability settings according to their communication needs per course. Formally the model can be formulated as follows: Every user has a set of registered courses (R) and a set of presence relations between the courses and online states (P); and every course has one or more users.

Users: $U = \{u_i \mid i \in N\}$

Courses: $C = \{c_j \mid j \in N\}$

Online states: $O = \{o_k \mid k \in N\}$

R, P are sets:

$\forall u \in U: u = \{R, P\}$

where $R = \{c_1, \ldots, c_n\}$

$\wedge\ P = \{p_{11}, \ldots, p_{nk}\} \mid p_{ij} = c_i \times o_j$

where $c_i \in C \wedge R \subseteq C,\ o_j \in O,\ R \cong P,$

$\forall c_i \in R\ \exists\ o_j \in O$

$\forall c \in C: c = \{u_1, \ldots, u_n\} \mid u_i \in U,\ c \neq \emptyset$

The second concept we introduce in PILS is *communication awareness*. Communication awareness helps the initiator of a conversation to see if the person to be contacted is engaged in other conversations. While in the real world people typically engage in a limited number of parallel conversations, they can have multiple parallel text chats online [29]. Although technically possible, in a learning context, where the cognitive load is usually high on the first level task, a user may still not want to handle too many parallel conversations as a background task. So, the PILSClient indicates in the conversation status, information whether the designated callee is currently in a conversation and over which channel: text, audio, or video. Since users can have multiple text chats in parallel, an additional number indicates how many simultaneous text chats are currently open for the callee.

The third concept are the *e-learning–specific online states* ReadyToHelp and RequiringHelp, which are available besides the other online states that most instant messaging systems offer today (i.e., available, unavailable, away, and invisible). ReadyToHelp indicates that a user is an expert in a specific course and is available to assist fellow learners. RequiringHelp signifies that a user has open questions in a specific course and looks for help from fellow students. The particular course for which a user sets one of the two states provides the adequate context to conclude on what topic the user is willing to help or the help is needed. In the current version of PILS, the users can set the online states themselves. When the system is stronger integrated into the adaptive e-learning platform, the e-learning management system can analyse the performance of the student in a specific course and then automatically adapt the online states.

B. Open Implementation

As discussed earlier PILS is developed to be fully integrated in an adaptive learning space. For this purpose PILS provides concepts for extensible remote controlling. The broad range of services offered can be divided into three categories: course and user administration services, presence and communication control services, and data and inference query services.

Course and User Administration Services

In PILS, remote administration is a prerequisite for the seamless integration into e-learning environments. This way the system can ensure that there is only a single point of administration and so minimise the effort for the user. When a new user is registered or an existing user registers for a new course or leaves a course in the e-learning platform, this can be put straight through to the PILSServer. PILS automatically adapts the system to the new conditions; new users are created, existing users are added to or removed from courses. The users can immediately log into PILS and have all enrolled courses equipped with all fellow students present to start communication.

Some examples of methods are: `registerUser`, `unregisterUser`, `addUserToCourse`, `removeUserFrom-Course`.

Presence and Communication Control Services

In PILS, services for controlling PILSClients at runtime are important for automatically adapting the presence and communication. This can, for instance, be used to automatically log a user into PILS from within the e-learning platform. Also, the online states of users can be set remotely. This helps to align the settings from the e-learning platform to the settings in PILS. If the user is working on a specific course in the e-learning platform, the online state of the user can be adjusted accordingly—for instance, to available for that course and to unavailable for the other courses of the user. Another possibility is to initiate multi-user text chats among a group of students from outside. This can be initiated automatically by an adaptation mechanism in the e-learning platform or manually by a teacher who wants the students to collaborate on a specific topic.

Some examples of methods are: `setUserOnlineStatus`, `openMultiUserTextChat`, `sendMessageToMultiUserTextChat`, `closeMultiUser-TextChat`.

Data and Inference Query Services

The third category of methods for remote calls in PILS includes methods that allow querying of usage data. These methods provide detailed communication metadata that can be used in an adaptive learning space in order to adapt the system to the communication patterns of individual users or a course. PILS continuously collects data from the users' activities (e.g., login and logout, changes to online states, online conversations). These data are then provided to the adaptive learning space. PILS makes data available that is specific to the communication of single users, of dyads of users, and of courses by continuously capturing and inferring on data and building and maintaining a model of the communication patterns. Typical data that are of interest to an adaptive learning space and can be retrieved from PILS are the following. The start and end times, duration and frequency of conversations of any specified dyad. Activity maps of single user or a specific course on hourly or daily basis can reveal rhythmic patterns of usage. Statistics on initiators of conversations, about reciprocity of conversations, about the average response time for communication requests and so on can be inferred. With this input the adaptive learning space can not only recommend a student who has the knowledge to help a fellow student but can also estimate how likely that student will answer and how likely the student will be met online. This can significantly increase the value of the recommendation of an expert.

PILS offers a range of methods for data and inference query. Some examples of methods related to individual users are: `getUserOnlineStatus`, `analyseStateUsage`, `analyseUserDaytimeActivity`, `analyseUserActivity`, `getUsersConversationPartners`, `getFirstResponse-Time`, `getResponseBehaviour`, `getAverageConversa-tionTime`. Some examples of methods related to conversations and messages are: `getStart-TimeOfConversation`, `getEndTimeOfConversation`, `getAverageDurationOfConversation`, `getFrequency-OfConversation`, `getInitiatorOfConversation`, `get-ReciprocityOfConversation`, `getConversationRes-triction`, `getAverageResponseLength`, `parseForKey-word`. Some messages specifically related to communication channels: `getActiveCourses`, `get-ChannelsOfCourse`, `getConversationChannels`, `ana-lyseChannelUsage`, `analyseAllCourseCommunication`. Several methods include aspects of the courses; some methods explicitly related to courses are: `analyse-CourseDaytimeActivity`, `analyseCourseActivity`.

III. IMPLEMENTATION

In the following we give an overview of the architecture of PILS and illustrate its orchestration of subsystems and components. The PILSClient and the PILSServer are implemented in Java SE 5.0 and optimised for Mac OS X and Windows XP. Figure 1 provides an overview of the structure of the PILSServer and PILSClientand their subsystems and components.

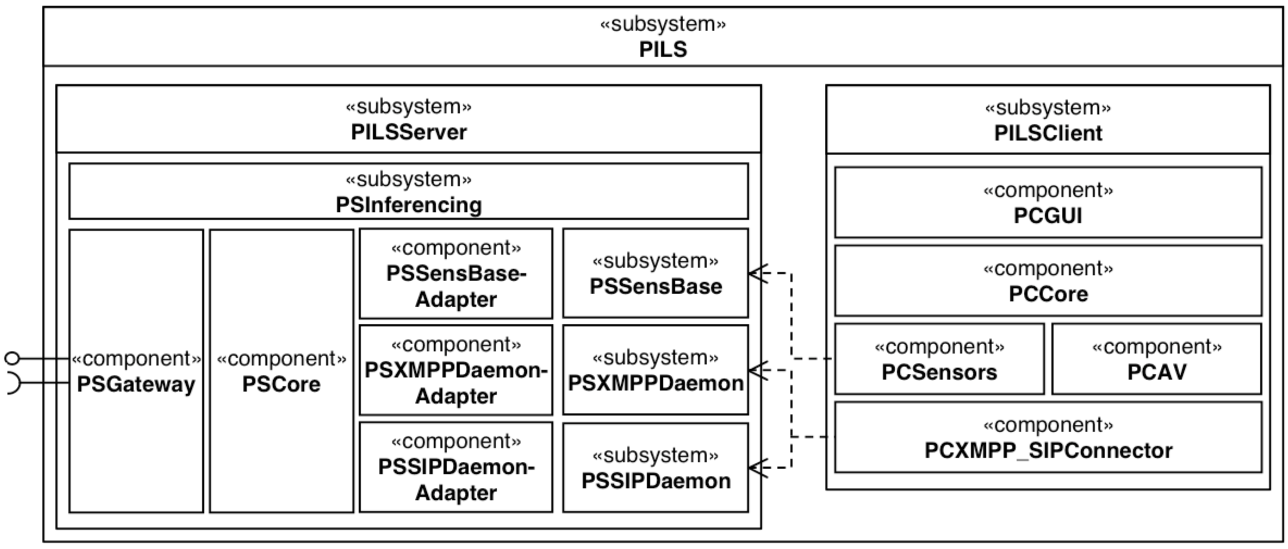


Figure 1. PILS component diagram consisting of PILSServer and PILSClient.

A. *PILSClient*

In this section we describe the implementation of the two core concepts for instant messaging—text, audio, and video chat as well as awareness—and the core concept of the open implementation. The PILSClient is realised as standalone Java application, and can be deployed and started with a single click over the network directly out of an e-learning platform via Java Web Start Technology.

Core Concepts for Instant Messaging

The implementation of the PILSClient and its *text, audio, and video chat* are based on different protocol and implementations for text and for audio and video. In order to enable dyadic and multi-user text chats the eXtensible Messaging and Presence Protocol (XMPP) [32] based on the Smack API library [24] is used. For initiating audio and video chats, calls are handled by the Session Initiation Protocol SIP [31] based on the Java API for SIP Signalling (JAIN SIP) [30].

When a user initiates a text, audio, or video call in the graphical user interface of the `PCGUI`, this call is forwarded as a method call to the `PCCore`. The `PCCore` then forwards the call to the `PSXMPP_SIPConnector`. For text chat no initialisation is needed and text messages are simply sent via XMPP. For audio and video chat calls are translated into SIP calls. The SIP calls are routed via the PILSServer to the respective other PILSClient as invitations. After the other PILSClient has responded with an ok, the initiating PILSClient sends an acknowledge and the session starts. The audio and video signals are then delivered between two clients by the `PCAV` via the Real-time Transport Protocol RTP [34].

Since the PILSClient supports audio and video communication both on Windows XP and on Mac OS X as well as cross-platform, specific treatment of RTP in the `PCAV` is necessary. A solution where JMF can be used for capturing and streaming on Windows and QuickTime for capturing and streaming on Mac OS X was not feasible—in cross-platform video conferencing the two frameworks could not decode the streams of the respective other system. Therefore, on Windows XP RTP streaming and audio and video capturing are handled by the Java Media Framework JMF [37]. On Mac OS X the JMF has no access to native methods that allow capturing audio and video data from microphones and cameras. Therefore, here the processing in the `PCAV` is the following: the `QTAudioPullBufferDataSource` and the `QTVideoPull-BufferDataSource` implement a JMF `PullBufferData-Source` interface, and the QTAudioPullBufferStream and `QTVideoPullBufferStream` implement a JMF `Pull-BufferStream` interface for audio and for video capturing access microphone and camera, and provide JMF compatible data to the `RTPManager`. The Java Sound API is used to sample audio from the microphone and QuickTime for Java [5] for receiving video data from the camera. The data is then transformed to comply with the requirements of JMF and is accessible through the two `PullBufferStreams` `read()` methods. This way also JMF's `RTPManager` can handle sending and receiving audio and video streams.

The implementation of the PILSClient and its *awareness support* is based on two different protocols. XMPP is used to transport all online states. Yet, the basic online states (i.e., available, unavailable, away, invisible) are handled with XMPP standard online states, whereas the e-learning–specific online states (i.e., readyToHelp, requiringHelp) are processed as status messages. In the user interface all basic and e-learning–specific online states are handled equally, so users do not see any difference. XML-RPC [38] is used to transport all conversation states. The `PCSensors` capture data on the initiation and termination of text, audio, and video chats, and sends these data to the `PSSensBase` subsystem of the PILSServer via XML-RPC, where the conversation status for each user is inferred (cf PILSServer below). The `PCSsensors` also captures other meta-information on the users' communication behaviour. Via a publish-subscribe mechanism the `PCSensors` is then informed over the inferred communication state of every user in the rosters.

Open Implementation in the PILSClient

The *open implementation* of the PILSClient allows for remote control of the PILSClient in three different ways. First, by utilising a parameterised Java Network Launching Protocol (JNLP) file in the Web browser using the Java Web start technology, the e-learning platform can provide users who are already logged into the e-learning platform with a user interface element that, when pressed, automatically downloads, starts, and logs the users into PILS. The PILSClient than receives a list of all courses the users are subscribed to and automatically logs them in. Second, the remote changing of online states is done directly on the PILSServer. The PILSClient automatically retrieves new remotely set online states for all buddies on the rooster directly via XMPP. Changes to the users own online states are propagated to the PILSClient via publish-subscribe from the PILSServer. Thirdly, multi-user text chats that are initiated remotely are also sent via XMPP. In this case the remote commands are sent from the PILSServer to the PILSClient in the properties of the message that the user wants to send. These are then processed by the PILSClient and lead to an automatic opening of a multi-user text chat window.

B. *PILSServer*

In this section we give a general overview, and discuss the core concept of open implementation. The PILSServer offers a single point of access for PILSClients as well as for e-learning platforms that want to take advantage of the PILS functionality via remote control. It is programmed in Java and leverages a series of services that are seamlessly integrated.

Core Concepts for Instant Messaging

The core concepts for instant messaging in the PILSServer are straight-forward. Dyadic and multi-user text chat is handled by the `PSXMPPDaemon`, an integrated Openfire server [23]. The initiation of audio and video chat is handled by the `PSSIPDaemon`, an integrated Brekeke SIP proxy and registrar [9]. Online states are also handled by the `PSXMPPDaemon`. The `PSSensBase`—a sensor-based infrastructure based on the Sens-ation platform [15]—receives the information from the PILSClient. This information is routed to the PSInferencing subsystem, which, for instance, calculates the current conversation states for each user and passes this information back to PSSensBase, where it is published to the PILSClients of all currently subscribed users.

Open Implementation in the PILSServer

The open implementation for the remote control of the *course and user administration services* is primarily based on the integration and smooth interplay of the PSCore with PSSensBase, PSXMPPDaemon, and PSSIPDaemon for course-specific awareness and communication.

The PILSServer administrates multiple courses for each user—that is, for each user in each course we need a specific online state, conversation state, and particularly one roster of course buddies. Since most instant messaging platforms support only one online state as well as one roster for each user, the PILSServer needs to extend these capabilities of existing platforms. The PILSServer creates several accounts for each user: one account per user and per course for the PSXMPPDaemon and the PSSIP-Daemon. For each course a shared group is created on the PSXMPPDaemon. Shared groups, like courses, are symmetrical—that is, a shared group automatically adds all its users to the rooster of any other user in this group. So, the server can automatically manage the roosters for users by adding new users when they register for a course and removing them when they unregister from a course.

The administration of multiple courses for each user works as follows: The PILSServer offers a single point of access via XML-RPC in PSGateway for the administration of users and their courses, and handles all necessary steps in the background. The PSGateway works as a proxy to the PSCore that has access to the domain model of all users and their subscribed courses, centrally backed up in PSSensBase. For this purpose the component PSSensBaseAdapter provides an entry point to the event-based data structure of PSSensBase for the PSCore by providing methods like registerUser or addUserToCourse that are then transformed to PSSensBase conform XML calls. The PSCore delegates calls for creating accounts, etc. to the PSXMPPDaemon and PSSIPDaemon. The XMPPDaemonAdapter and SIPDaemon-Adapter are thereby responsible for passing calls to their respective daemon by translating them to http calls to the administration interfaces of the particular service.

Since this interplay is complex, we provide the user registration and addition of a user to a course via XML-RPC in an example sequence diagram (cf. Figure 2).

The call flow in this sequence diagram is the following for new users. When users register in the e-learning platform and subscribe for their courses, the e-learning platform calls the registerUser method of the

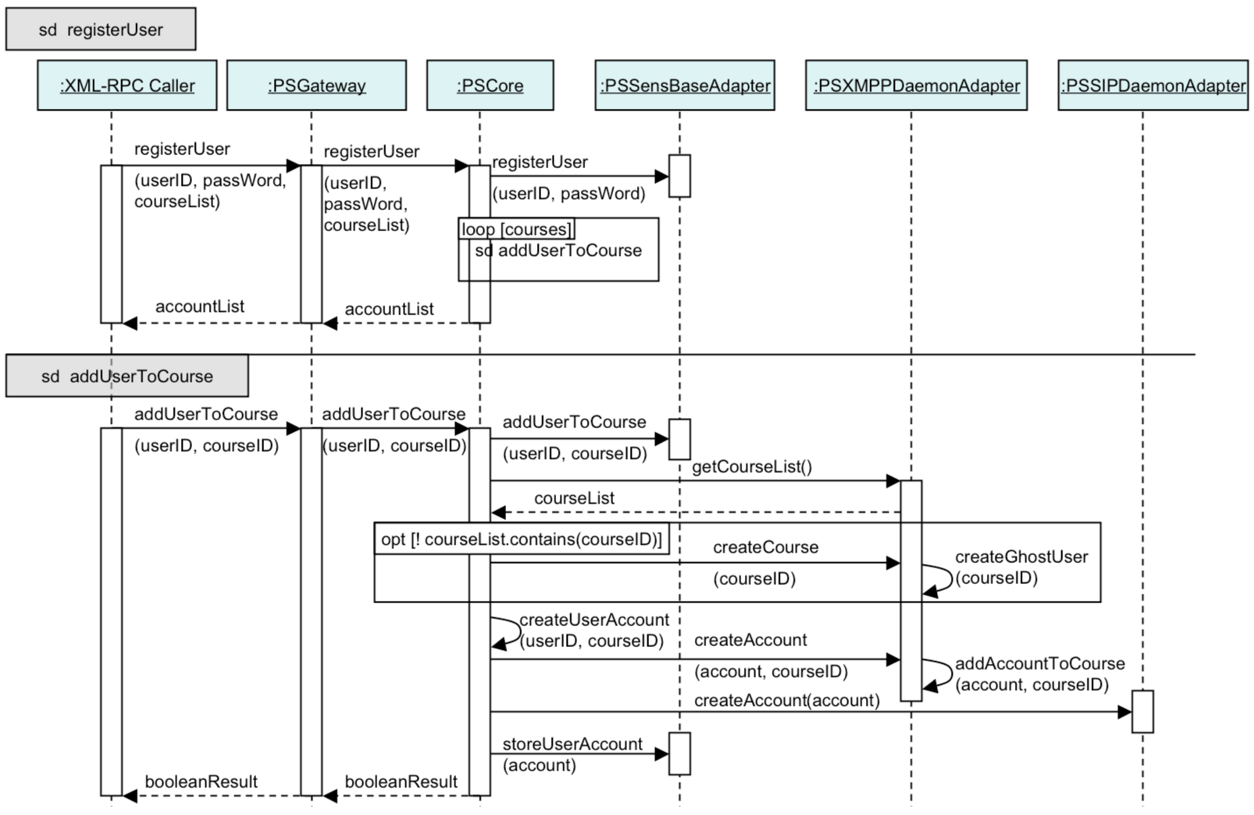

sd registerUser
:XML-RPC Caller
:PSGateway
:PSCore
:PSSensBaseAdapter
:PSXMPPDaemonAdapter
:PSSIPDaemonAdapter
registerUser (userID, passWord, courseList)
registerUser (userID, passWord, courseList)
registerUser (userID, passWord)
loop [courses]
sd addUserToCourse
accountList
accountList
sd addUserToCourse
addUserToCourse (userID, courseID)
addUserToCourse (userID, courseID)
addUserToCourse (userID, courseID)
getCourseList()
courseList
opt [! courseList.contains(courseID)]
createCourse (courseID)
createGhostUser (courseID)
createUserAccount (userID, courseID)
createAccount (account, courseID)
addAccountToCourse (account, courseID)
createAccount(account)
storeUserAccount (account)
booleanResult
booleanResult

Figure 2. Sequence diagram for user registration and the addition of a user to a course.

`PSGateway` of the PILSServer via XML-RPC. The method is forwarded to the `PSCore`. The `PSCore` passes the user ID, the password, and the list of courses of the user via the `PSSensBaseAdapter` in the `PSSensBase`, where the credentials are stored.

Afterwards, in a loop, the users are added to every course they have registered for by calling the `PSCore`'s `addUserToCourse` method, which than runs through the following steps for each course. At first the `PSCore` stores the information about the user subscriptions to the course in `PSSensBase` via the `PSSensBaseAdapter`. The `PSCore` then retrieves a list of all available courses from the `PSXMPPDaemon` via the `PSXMPPDaemonAdapter`. If a user is the first one to register to a course, a new course is created in a two-step process: a new shared group is created on the `PSXMPPDaemon`, and a ghost user account is created for this course and added to the shared group. After the course is created—or if the course already existed—a course-specific account of the form `userID$courseID@-pils` is generated for the users. With this account information the `XMPPDaemonAdapter` registers the users and adds their accounts to the shared group with the `add-AccountToCourse` method, both at the `PSXMPPDaemon`. In the next step the `PSCore` initiates the creation of a SIP account of the same form `userID$courseID@pils` via the `PSSIPDaemonAdapter`. After both accounts are generated the account information is stored together with other user information via `PSSensBase` and a `true` is delivered back by the `addUserToCourse` method to the caller.

After all courses have been processed in the loop, the `PSCore` hands back an `accountList` based on all these results via XML-RPC to the caller.

If a user is already registered, this user can be added to courses by directly calling the `addUserToCourse` method.

After the users are registered and added to their courses, the `PILSClient` logs into the `PILSServer` by providing userID and password, the `PILSClient` receives a list of the accounts the user is registered for and autonomously logs into the `PSXMPPDaemon` and `PSSIPDaemon`.

The call flow to unregister users or remove them from a course has a similar complexity.

The open implementation for the remote control of the *presence and communication control services* is divided into setting the online states of users, and starting a multi-user text chat. For online states, the `PSGateway` passes the call from the remote caller containing the user, the course, and the new online status to the `PSCore`. The `PSCore` fetches the user's account information from `PSSensBase` and logs into the XMPP account via the `PSXMPPDaemonAdapter`, changes the online state, and logs out again. The new status is then transmitted to all buddies via XMPP. The `PSSensBase-Adapter` notifies the users client via XML-RPC. For opening of a multi-user text chat, a similar call flow is used. But in this case the `PSXMPPDaemonAdapter` uses the ghost user account instead, created in every course and not visible to clients. With this account the PILSServer logs in and sends a message with special properties to all users of a course.

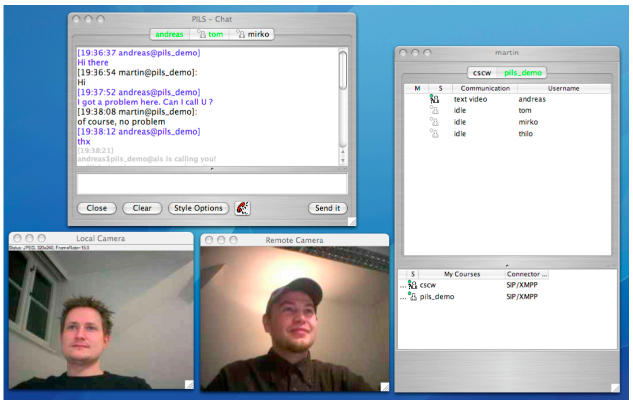
IV. USER INTERACTION WITH PILS

In this scenario we demonstrate how users can interact with PILS in order to have sophisticated online presence and smooth online communication. The screenshot (cf. Figure 3) shows the student Martin's PILS client main window on the right, a text chat window on the top left, and a video chat window on the bottom left.

In the main window he can see the two tabs of his courses cscw and pils_demo, and the currently opened tab for the pils_demo course with a list of his fellow students. Andreas is currently online and indicates that he is requiring help (cf. Andreas' icon in buddy list). Tom, Mirko and Thilo are currently offline. At the bottom of the main window Martin can see that he has set his status to requiringHelp for cscw and available for the pils_demo course (cf. icons in list on bottom of the main window).

Martin is currently having a text, and audio and video chat with Andreas. As no other users are currently online, Andreas took the initiative and started a conversation with Martin to ask for help (cf. chat window in upper left). Probably Andreas had noticed by means of Martin's communication awareness that Martin just has ended two other conversations—as can be seen in the chat window, based on the opened tabs in the background, Martin indeed had two text chats with Tom and Mirko, but both users have left PILS already and are now offline. So, Andreas approached Martin, since Martin is currently the only online user in this course. In a text chat Andreas asked Martin if he is willing to help him with a problem in the pils_demo course. As Martin agreed, Andreas started a video chat (cf. lower left). On start of the video chat, the PILS client updated the communication awareness of Andreas and Martin, showing that they currently have an ongoing text as well as an audio and video chat (cf. Andreas' icon in buddy list).

This scenario shows the explicit interaction of Martin and Andreas with the PILS client. It would be easy to extend the scenario to situations in which the remote control is used from other applications within the adaptive learning space to, for instance, add or remove users from courses, or to automatically start a text chat between Martin and Andreas.



[19:36:37 andreas@pils_demo]
Hi there
[19:36:54 martin@pils_demo]:
Hi
[19:37:52 andreas@pils_demo]
I got a problem here. Can I call U ?
[19:38:08 martin@pils_demo]:
of course, no problem
[19:38:12 andreas@pils_demo]
thx
[19:38:21]
andreas$pils_demo@als is calling you!

M	S	Communication	Username
		text video	andreas
		idle	tom
		idle	mirko
		idle	thilo

S	My Courses	Connector ...
	cscw	SIP/XMPP
	pils_demo	SIP/XMPP

Figure 3. PILS client with main window, text chat window, and audio and video chat window.

V. RELATED WORK

There are three categories of concepts and systems with relation to the specific contributions of PILS: instant messaging in general, instant messaging support for e-learning, and open application programming interfaces.

Instant messaging systems in general provide basic functionality for online presence and communication. Examples of wide-spread systems are single-protocol instant messaging systems supporting specific communication protocols such as ICQ [22] and Skype [35]; as well as multi-protocol instant messaging systems supporting a number of protocols in parallel such as Miranda [27] for Windows and Adium [1] for Mac OS X. Some systems provide mechanisms for organising contacts in groups such as iChat [4]; and some prototypes provide concepts for multiple online identities such as PRIMIFaces [16]. Only few prototypes aim to show communication awareness to provide an impression of users' availability; examples are the OpenMessenger [8] that supports blurred glances at other user's computer screens to see if they are busy, and the ChatCircles that provide a visualisation of chatting users through the proximity of their circular representation [11].

Overall, these systems and prototypes provide interesting base functionality for instant messaging, with some extensions. However, their concepts are not adapted to e-learning and their implementation is often closed.

Instant messaging support for e-learning are underrepresented, since course management systems for online learning often only provide rather basic communication functionality. For instance, the two most wide-spread open source learning management systems Sakai [33] and Moodle [28] mainly offer course-specific Web-browser-based group text chat along with asynchronous communication functionality such as in forums. Sakai offers dyadic text and video chat with an add-on named Agora [26], but does not provide course-specific awareness in this tool. Another third-party development for Sakai is the context-aware activity notification system CANS; it informs users of activity in the e-learning platform, also providing information on ongoing communication [3]. However, it does not offer direct means to initiate conversations with other users.

Compared to the mentioned systems PILS has greater flexibility, since it supports detailed awareness of online states and conversation states within each course, but also global notifications (e.g., for incoming chat request) on a global level outside the individual courses. Furthermore, it supports both bilateral conversations among two users as well as multilateral conversations among all online course members. To our knowledge, there are no e-learning platforms that support e-learning–specific online states.

Open application programming interfaces are not common for instant messaging, but are wide-spread in other areas. The base technology of XML-RPC [38] and especially of its successor SOAP [18] provides great means for open interfaces, where remote procedure calls are encoded into XML messages that are sent via HTTP/HTTPS. Yet, they are hardly used in instant messaging. Some examples of instant messaging systems with open interfaces are commercial instant messaging systems such as IBM Lotus Sametime [21], and Trillian [10]. Sametime is based on IBM portlets and it can be easily integrated into these portlet applications; and Trillian provides its own application programming interface, primarily in the C programming language.

In comparison, the PILS infrastructure has a very sophisticated, yet easy to access, open implementation based on Sens-ation and its SensBase implementation [15]. It provides a light-weight interface via XML-RPC towards the outside, and easy mechanisms for adding new query and inferencing engines inside.

VI. CONCLUSIONS

We introduced the PILS infrastructure providing advanced instant messaging concepts for e-learning based on an open implementation. We have presented the concept and implementation, and illustrated its use in a scenario. We have discussed related work.

There are two areas of ongoing and future activities: From the open implementation perspective PILS needs more sophisticated inferencing engines—that is, it needs more sensors capturing information and more algorithms (incl. machine learning) for inferring on the captured data to get an even better impression on the users' current activity and availability. From the user interaction concepts' perspective we have made some initial user studies, but systematic feedback on the current functionality and recommendations for revisions of existing and additional new functionality still need to be done. The initial studies showed that users liked the system, and did not have any concerns about privacy—this was important for us, since in PILS information about users is captured and presented.

In the next steps the PILS system needs to be integrated with the Sakai e-learning environment and more systematic user evaluations need to be made to show the real strengths from a users' perspective. This integration will particularly benefit from the open implementation and will make the user interaction more convenient (e.g., a user only needs to log in into Sakai, and Sakai can then log this user into PILS via XML-RPC calls).

ACKNOWLEDGMENTS

Thanks to Thilo Paul-Stueve, Martin Klusmann, Andreas Thenn, and all members of the Cooperative Media Lab; Part of the work in this paper has been funded by the EU Socrates Minerva project Adaptive Learning Spaces (ALS) (229714-CP-1-2006-1-NL-Minerva). Thanks to the anonymous reviewers for valuable feedback.

REFERENCES

[1] Adium Team. *Adium - Download.* http://www.adiumx.com/, 2008. (Accessed 3/8/2008).

[2] ALS Project. *Adaptive Learning Spaces Projetcts.* http://www.als-project.org/, 2008. (Accessed 3/8/2008).

[3] Amelung, C. Using Social Context and E-Learner Identity as a Framework for an E-Learning Notification System. *International Journal on E-Learning 6*, 4 (2007). pp. 501-517.

[4] Apple Computer Inc. *Apple - Mac OS X Leopard - iChat*. http://www.apple.com/macosx/features/ichat/, 2008. (Accessed 20/10/2008).

[5] Apple Computer Inc. *Quicktime for Java*. http://developer.apple.com/quicktime/qtjava/, 2008. (Accessed 23/07/2008).

[6] Bates, A.W.T. *Technology, E-Learning and Distance Education*. Routledge, London, Great Britain, 2005.

[7] Begole, J.B., Matsakis, N.E. and Tang, J.C. Lilsys: Sensing Unavailability. In *Proceedings of the 2004 ACM Conference on Computer-Supported Cooperative Work - CSCW 2004* (Nov. 6-10, Chicago, Illinois, USA). ACM Press, 2004. pp. 511 - 514.

[8] Birnholtz, J.P., Gutwin, C., Ramos, G. and Watson, M. OpenMessenger: Gradual Initiation of Interaction for Distributed Workgroups. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2008* (Apr. 5-10, Florence, Italy). ACM, N.Y., 2008. pp. 1661-1664.

[9] Brekeke Software Inc. *Brekeke SIP Server - SIP Proxy, Registrar Server*. http://www.brekeke.com/sip/, 2008. (Accessed 23/07/2008).

[10] Cerulean Studios. *Main Page - TrillWiki*. http://developer.ceruleanstudios.com/index.php/Main_Page, 2008. (Accessed 3/8/2008).

[11] Donath, J. and Viegas, F. The Chat Circles Series: Explorations in Designing Abstract Graphical Communication Interfaces. In *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques - DIS 2002* (June 25-28, London, UK). ACM, N.Y., 2002. pp. 359-369.

[12] Dourish, P. and Bellotti, V. Awareness and Coordination in Shared Workspaces. In *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work - CSCW 1992* (Oct. 31 - Nov. 4, Toronto, Ontario, Canada). ACM Press, 1992. pp. 107-114.

[13] Dourish, P. and Bly, S. Portholes: Supporting Awareness in a Distributed Work Group. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 1992* (May 3-7, Monterey, California, USA). ACM Press, 1992. pp. 541-547.

[14] Gonzalez, V. and Mark, G. Managing Currents of Work: Multi-Tasking Among Multiple Collaborations. In *Proceedings of the Nineth European Conference on Computer-Supported Cooperative Work - ECSCW 2005* (Sept. 18-22, Paris, France). Springer-Verlag, Heidelberg, 2005. pp. 143-162.

[15] Gross, T., Egla, T. and Marquardt, N. Sens-ation: A Service-Oriented Platform for Developing Sensor-Based Infrastructures. *International Journal of Internet Protocol Technology (IJIPT) 1*, 3 (2006). pp. 159-167.

[16] Gross, T. and Oemig, C. From PRIMI to PRIMIFaces: Technical Concepts for Selective Information Disclosure. In *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications - SEAA 2006* (Aug. 29-Sept. 1, Cavtat, Dubrovnik, Croatia). IEEE Computer Society Press, Los Alamitos, CA, 2006. pp. 480-487.

[17] Gross, T., Stary, C. and Totter, A. User-Centered Awareness in Computer- Supported Cooperative Work-Systems: Structured Embedding of Findings from Social Sciences. *International Journal of Human-Computer Interaction 18*, 3 (June 2005). pp. 323-360.

[18] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., Nielsen, H.F., Karmarkar, A. and Lafon, Y. *Simple Object Access Protocol (SOAP)*. W3C, http://www.w3.org/TR/SOAP/, 2008. (Accessed 3/8/2008).

[19] Gutwin, C., Stark, G. and Greenberg, S. Support for Workspace Awareness in Educational Groupware. In *Proceedings of the Conference on Computer Supported Collaborative Learning - CSCL 1995* (Oct. 17-20, Bloomington, IN). Lawrence Erlbaum Associates, 1995. pp. 147-156.

[20] Hudson, S.E. and Smith, I. Techniques for Addressing Fundamental Privacy and Disruption Tradeoffs in Awareness Support Systems. In *Proceedings of the ACM 1996 Conference on Computer-Supported Cooperative Work - CSCW'96* (Nov. 16-20, Boston, MA). ACM, N.Y., 1996. pp. 248-257.

[21] IBM. *IBM Software - IBM Lotus Sametime*. http://www-306.ibm.com/software/lotus/sametime/, 2008. (Accessed 3/8/2008).

[22] ICQ Inc. *ICQ.com - Community, People Search, and Messaging Service!* http://www.icq.com/, 2008. (Accessed 3/8/2008).

[23] Ignite Realtime. *Openfire Server*. Jive Software Community, http://www.igniterealtime.org/projects/openfire, 2008. (Accessed 23/07/2008).

[24] Ignite Realtime. *Smack API*. Jive Software Community, http://www.igniterealtime.org/projects/smack/, 2008. (Accessed 23/07/2008).

[25] Iqbal, S. and Bailey, B.P. Leveraging Characteristics of Task Structure to Predict the Cost of Interruption. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2006* (Apr. 22-27, Montreal, Canada). ACM, N.Y., 2006. pp. 741-750.

[26] Lancaster University. *Agora. The Online Meeting Tool*. http://agora.lancs.ac.uk/, 2008. (Accessed 29/07/2008).

[27] Miranda IM. *Miranda IM - Home of the Miranda IM Client. Smaller, Faster, Easier*. http://www.miranda-im.org/, 2008. (Accessed 3/8/2008).

[28] Moodle. *Moodle - A Free, Open Source Course Management System for Online Learning*. http://moodle.org/, 2008. (Accessed 29/07/2008).

[29] Nardi, B.A., Whittaker, S. and Bradner, E. Interaction and Outeraction: Instant Messaging in Action. In *Proceedings of the 2000 ACM Conference on Computer-Supported Cooperative Work - CSCW 2000* (Dec. 6-10, Philadelphia, USA). ACM Press, 2000. pp. 79 - 88.

[30] Ranganathan, M. and O'Doherty, P. *JAIN-SIP: Java API for SIP Signaling*. https://jain-sip.dev.java.net/, 2007. (Accessed 23/07/2008).

[31] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and Schooler, E. *SIP: Session Initiation Protocol (RFC 3261)*. http://www.ietf.org/rfc/rfc3261.txt, 2002. (Accessed 23/7/2008).

[32] Saint-Andre, P. *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence (RFC 3921)*. http://www.ietf.org/rfc/rfc3921.txt, 2004. (Accessed 23/7/2008).

[33] Sakai Foundation. *Sakai: Collaboration and Learning Environment for Education*. http://sakaiproject.org/, 2008. (Accessed 29/07/2008).

[34] Schulzrinne, H., Casner, S.L., Frederick, R. and Jacobsen, V. *RTP: A Transport Protocol for Real-Time Applications (RFC 3550)*. The Internet Society, http://www.ietf.org/rfc/rfc3550.txt, 2003. (Accessed 23/07/2008).

[35] Skype. *Make the most of Skype - Free Internet Calls and Cheap Calls*. http://www.skype.com/, 2008. (Accessed 20/10/2008).

[36] Stahl, G. *Group Cognition: Computer Support for Collaborative Knowledge Building*. MIT Press, Cambridge, MA, 2006.

[37] SUN Microsystems Inc. *Java Media Framework API (JMF)*. http://java.sun.com/javase/technologies/desktop/media/jmf/, 2008. (Accessed 23/07/2008).

[38] Winer, D. *XML-RPC Specification*. UserLand Software, http://www.xmlrpc.com/spec, 1999. (Accessed 10/07/2007).