Advanced Publish and Subscribe for Distributed Sensor-Based Infrastructures: The CoLocScribe Cooperative Media Space

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

***Abstract*—Sensor-based infrastructures are important base technology for cooperative media spaces that support the natural interaction of users with their physical environment and with other users beyond the traditional keyboard and mouse. Sensor-based infrastructures basically capture data from sensors, store and process these data, and provide the data to clients. Several infrastructures have been developed; they all have their specific strengths in supporting either publish or subscribe, push or pull. In this paper we present a generic and advanced concept and implementation of a publish and subscribe mechanism for distributed sensor-based infrastructures that is sophisticated yet easy to configure and that is resource-saving through load balancing and provides push and pull. We exemplify this publish and subscribe mechanism in the CoLocScribe use case where we designed and developed advanced publish and subscribe for a cooperative media space.**

***Keywords*—Distributed System; Sensor-Based Infrastructures; Cooperative Media Space.**

I. INTRODUCTION

Sensor-based infrastructures [9] are an important base technology for ubiquitous environments that support the natural interaction of users with their physical environment beyond the traditional keyboard and mouse [1, 26]. Distributed sensor-based infrastructures for ubiquitous environments in cooperative media spaces support the social interaction among co-located groups of users as well as between different co-located groups [12]. All these sensor-based infrastructures are basically composed of three building blocks: sensors that capture information from the real world and from the electronic world, processing units that store and infer on the captured data, and actuators that push changes and adaptations to the environment based on the inferencing results [8].

Several sensor-based infrastructures have been developed (e.g., Sens-ation [9], Khronika [17], Elvin [6], Siena [4]). Their functionality can be characterised on two dimensions: publish (i.e., provision of data to interested parties) and subscribe (i.e., registration of interests in data); as well as pull (i.e., manual queries of clients) vs. push (i.e., automatic delivery of sensor data to clients). None of them support both dimensions completely.

In this paper we present the concept and implementation of the Pub/Sub advanced publish and subscribe concept for distributed sensor-based infrastructures that is sophisticated yet easy to configure and that is resource-saving through load-balancing and provides push and pull. In the next section we introduce the generic concepts of Pub/Sub for publish and subscribe and their evolving adaptations. We then report on the CoLocScribe use case, where we developed a cooperative media space based on the Pub/Sub concepts and did an empirical study to evaluate both the generic concepts and the specific system. We provide details on the implementation of Pub/Sub and CoLocScribe. We sketch related work. And, finally, we draw conclusions and glance at future work.

II. PUB/SUB CONCEPT

The Pub/Sub concept provides an advanced publish and subscribe mechanism to provide scalable adaptive publishing of information to interested subscribers. Publish is the process of making sensor data that is captured in a specific context (e.g., the temperature measured in an office, the online state in an instant messaging client) public and delivering these sensor data. Subscribe means the process of expressing interests in certain sensor data and getting notified according to these interests (e.g., a warning when the temperature in a remote room goes above 35 degrees Celsius). Figure 1 depicts the basic concept of technological support for publish and subscribe; the details are explained in the sub-sections below.

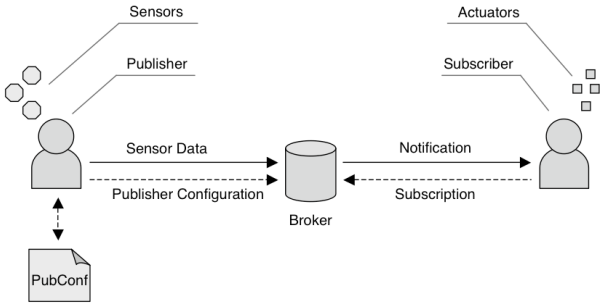

Figure 1. The publish and subscribe concept (dashed arrows are control flow; solid arrows are actual information flow).

A. Publish of Pub/Sub

The publish concept provides mechanisms for selective sharing of sensor data. It supports users with powerful, yet easy to use support to add new sensors, to specify preferences for disclosing captured sensor data, and to actually share these data. Sensors can be hardware that is typically small sensing devices capable of measuring a series of values in the real world (e.g., temperature, movement), or software that is mainly capturing data from local or remote computers (e.g., online calendar information, program-switching behaviour). Besides the base technology for sensors and networking, the configuration of publications is a *two-step process* registering new sensors, and activating registered sensors.

Registering new sensors in the publisher configuration is done by inserting a short sensor description into a publish configuration file—the PubConf file—in eXtensible Markup Language (XML) format. Sensor descriptions include general data about the sensor, and an arbitrary number of collecting variants of the sensor's data capturing behaviour representing different levels of granularity of sensor data to be shared. The following lines we show an example of a PubConf file:

```
<?xml version="1.0" encoding="UTF-8"?>
<AvailableSensors>
  <Sensor name="Apple Mail" type="Other"
subtype="applescript" fields="none" >
    <Variant privacyType="public"
path="./scpt/AppleMail/newMailsCountXML.scpt">
Unread email count
    </Variant>
    <Variant privacyType="private"
path="./scpt/AppleMail/newMailsDetailXML.scpt"
>New email with name of sender and subject
    </Variant>
  </Sensor>
  <Sensor name="iCal" type="Other"
subtype="applescript" fields="none" >
    <Variant privacyType="public"
path="./scpt/iCal/getEventsCountXML.scpt">Cale
ndar events count
    </Variant>
    <Variant privacyType="private"
path="./scpt/iCal/getEventsDetailXML.scpt">Cal
endar events in detail
    </Variant>
  </Sensor>
</AvailableSensors>
```

Activating registered sensors is done by selecting the sensor and the level of granularity from a list of available sensors and levels: the system parses the PubConf file and knows all publisher configurations with sensors and their collecting variants, and allows to select sensors and one specific collecting variant for each sensor. The selected sensors and collecting variants are then sent to the broker where they are activated, and the capturing and transmission of the respective sensor data starts to run in a background process.

B. Subscribe of Pub/Sub

The subscribe concept provides mechanisms for specifying interests in and receiving sensor data from the infrastructure. It aims to support subscribers expressing a specific interest in sensor data and specifying the preferences for the notification, as well as to provide mechanisms for the actual parsing and matching of data and notifications about matches found. Subsequently we describe the specification, the matching, and the notification.

Specifying interests allows a subscriber to place subscriptions containing simple or advanced expressions that represent the selection of sensor data they want to be notified about: Simple expressions represent exact values in the format of fields and values <SensorDataFieldName>: <SensorDataFieldValue> (e.g., SensorName: "iCal" and SensorValue: "Project CoLocScribe Meeting" matches all calendar entries related to meetings of the CoLocScribe project). Advanced expressions represent ranges of sensor data and have the following general format <SensorDataFieldName>: {<LowerBound>|<Asterisk>}{<UpperBound>|<Asterisk>} (e.g., SensorName: "B11.Temperature" and SensorValue: "50;*" matches all room temperatures greater then 50 degrees Celsius, which could be used for a fire alarm notification). In Table 1 we illustrate the four possible permutations of upper and lower bounds with some examples.

Description	Syntax	Example
Inside: a closed interval of interest	<LowerBound>; <UpperBound>	"23:26.5" for temperatures greater than or equal to 23ºC, but less than or equal to 26.5ºC
Outside: interest in all values, except for a specific closed interval	<UpperBound>; <LowerBound>	"400;240" for movement levels lower than 240 and higher than 400
Less than: an upper limit of interest	<Asterisk>; <UpperBound>	"*;35" for temperatures less than or equal to 35ºC
Greater than: a lower limit of interest	<LowerBound>; <Asterisk>	"12;*" for temperatures greater than or equal to 12ºC

Table 1. Subscription expressions.

The subscription also contains further information: The *subscription period* is the start and end dates between which the parsing and matching of data takes place. The *notification type* specifies how the information is presented, which can either be of type thin notification (only the fact that a match was made is communicated) or

of type fat notification (the matching sensor data is communicated). The *notification interval* is the rhythm between notifications that can be 0 meaning that the notification is sent immediately after the match, or any value expressed in seconds, minutes, hours, days, or weeks (e.g., 1h means that the notification is always presented at the exactly same time of the hour).

Matching subscriptions to publications is comparing the names and the values of all individual fields of each sensor data that is captured with the names and the values of all individual fields of the specifications of interest in the subscriptions. It is important to note that in the Pub/Sub concept on the publishers' side only the sensor data are captured, stored, and free for comparison that are part of the publications; and on the subscribers' side only the sensor data that matches is considered and the unmatched data are ignored.

All sensor data have a standardised schema including mandatory fields such as Sensor Type, Sensor Value, Occurrence Date, Occurrence Time, and Location; optional fields such as User List, Urgency, Sampling, Frequency, Granularity, Ingredients, and Relationship; as well as custom fields that can be any key-value pairs that might be needed. Therefore, the comparison between the names and the values of the fields of the published data and the subscribed data is straightforward.

This matching concept allows load balancing between clients and servers, since the clients only receive filtered and pre-processed events. This is particularly valuable for thin clients in complex ubiquitous environments, where typically a broad range of sensors captures vast amounts of data that can hardly be handle by the thin clients.

Notifying subscribers happens when the comparison described above is successful and finds a match. Then a notification is sent to the subscriber according to its preferred notification type. It is also checked if the notification interval and sends notification with the preferred timing. The subscriber can then program any actuator to react on the notification such as starting a presentation of data, or triggering actions in a database.

C. Adaptable Publish and Subscribe

The above-described concepts for publish and subscribe in Pub/Sub provide both publishers and subscribers with powerful mechanisms to share and receive mutual information. They also facilitate the easy configuration and reconfiguration of publication and subscription preferences over time. This is particularly important in settings where the needs and requirements for publishing and subscribing change frequently (e.g., media spaces and ubiquitous environments).

The concept allows the fast and straightforward configuration of sensors as information sources. Besides the creation of new sensors, the publisher is capable of activating and deactivating information source immediately. Subscribers can subscribe to information they are interested in, such as certain data of a sensor without knowing the infrastructure deeply in form of sensors identification numbers.

III. THE COLOCSCRIBE USE CASE

In this CoLocScribe use case we exemplify how these flexible and adaptable concepts are applied to a media space and ubiquitous environment. In CoLocScribe we used the Pub/Sub concept, and developed a distributed sensor-based infrastructure on top of it that can then be deployed with real users to learn about its usefulness and usability and to get inspiration for further developments.

The CoLocScribe cooperative media space supports the *Co*-presence of users at the same or remote *Loc*ations that is based on a publish and sub*Scribe* concept in order to provide users with mutual awareness information. This mutual awareness information provides an important basis for users to have a shared frame of orientation in the team and to therefore work more effectively and efficiently together, independent of their location [13, 14]. The awareness information is captured with sensors, mediated by a distributed sensor-based infrastructure enhanced with Pub/Sub concepts, and presented on large public displays.

The support for awareness information faces several conceptual challenges—one is specifically the dual trade-off between on the one hand the need for information, and on the other hand the wish for privacy as well as for little disruption [15]. Concepts for selective information disclosure, where users can specify the information they want to share and the recipients of the information try to address this dual trade-off (e.g., [11, 16, 20]). With the CoLocScribe use case we wanted to get input for new concepts of selective information disclosure based on publish and subscribe.

A. CoLocScribe Setup

The CoLocScribe cooperative media space was deployed at our laboratory in two remote sites, where sensors captured the information from either room, and where the mutual information was presented on large public displays in either room [18]. A permanent video link supported additional mutual visual awareness.

The CoLocScribe cooperative media space consists of specific *hardware at each site*. On each site a 37" widescreen monitor is mount on a free wall, so it can be viewed from each student's desk. A camera is mounted close to the public display in order to cover a wide area of each site. Both, the monitor and the camera, are connected to a Mac mini computer.

The CoLocScribe cooperative media space provides *two software applications at each site*. The *PubClient* of the CoLocScribe is the software application that runs on all workstations of the users. It enables users to disclose information. As a publishing default setting for the user study, users can disclose four types of sensors at three different levels of detail. Table 2 summarises the sensors and levels of granularity used in the CoLocScribe use case.

The *SubClient* of the CoLocScribe is the software application shown on the widescreen monitor. It presents a video stream for looking into the distant site and a smaller one as control view of the appearance of the local

site. It also shows the actually gathered awareness information and a floor plan of the remote site. The windows can be freely arranged.

The overall configuration of the CoLocScribe cooperative media space supports reciprocity of awareness information, where on either site the same information is presented on the widescreen monitor. Furthermore, the information that is presented is coupled to present persons—that is, if a person leaves a site, her information is not displayed any longer.

Sensor	Low	Medium	High
BSCW: updates in the shared workspace	No information about workspace	Count of recent changes in workspace	Detailed information about changes in directories and files (incl. owner, modification time)
Calendar: calendar appointments	No calendar appointments	Count of appointments for current and next day	Concrete appointments (incl. start and end dates, location)
Email: unread email	No information about unread email	Count of unread email in inbox	Detailed information (incl. sender name, subject, date received)
Music: music playing information	No information about currently listened music	Playing state of music player ('playing', 'paused', 'stopped')	Playing state (incl. artist name, song title)

Table 2. CoLocScribe sensors and granularity.

B. CoLocScribe User Study

The user study we conducted with the CoLocScribe media space was a pre-study to obtain information of how adaptation to co-present others is done and which factors are important for adapting the publication and subscriptions of awareness information in a co-present media space.

For this pre-study nine participant used the CoLocScribe media space over one week to cooperate on their student projects. Their age was between 23 and 31 years and they were studying for at least three years. There were eight students working in four student projects with two participants each, and one additional student writing on a bachelor's thesis. The rooms in which they worked were located in two buildings in the same city. While the participants worked in their rooms and with the media space, three different work situations could be identified: project work means they are working on artefacts (e.g., source code or documents); presentation preparation means they were preparing for the weekly project meeting (where the presentations were stored in a shared workspace system), and project meeting means they presented their work to supervisors.

After the use period we conducted a comprehensive qualitative interview study as described in [21]. We used an interview guide containing 51 questions in five sections as a flexible methodical technique for making the participants talk.

C. CoLocScribe Study Results

All interviews were recorded, transcribed, and analysed. All participants expressed that they used the CoLocScribe cooperative media space intensely for communication and awareness, and felt less distance during the use.

The publishing—that is, the information disclosure—of the users followed patterns. Typically, disclosure was configured at the beginning and included the relation of trust towards all present group members. Later, the users only seldom reacted to changing co-presence among the group members (e.g., activated or deactivated different levels of granularity). So, the co-presence of other student group members had low influence. However, the participants reported that the presence of superiors had medium influence on their information disclosure configuration (e.g., they adapted their configuration before the weekly project meeting when the superiors visited the site through lowering the level of granularity of disclosed information).

Besides the reaction to the changed presence in the media space there was also another factor of influence: Two users reported that they adapted their configuration in reaction to the publication behaviour of other peers. For instance, some users received additional calendar information (i.e., the other user changed the collecting variant of the calendar appointment sensor from medium to high) and immediately reacted by also publishing more information (i.e., they also adapted the collecting variant of the calendar appointment sensor from medium to high).

Despite the fact that overall the publishing was changed infrequently, the PubClient was perceived as fast and easy. This includes the manual adaptation of the levels of granularity such as in the case another person entered.

So, these results provides three important lessons for the CoLocScribe cooperative media space specifically, but also for the Pub/Sub concept in general: The *CoLocScribe* cooperative media space was well received and users were satisfied with it—despite some initial sceptic thoughts on privacy. So, publish and subscribe concepts as in Pub/Sub are needed. The *adaptation* of the publishing took place, although not as frequently as expected. So, publishing concepts need to be easy and flexible in their configuration and reconfiguration over time. The adaptations can have *subtle reasons*, which are not always immediately obvious. So, concepts for automatic adaptations of publish and subscribe need to be developed with care.

IV. IMPLEMENTATION

In this section we provide details on the software architectures of the Pub/Sub infrastructure and the CoLocScribe cooperative media space.

A. Pub/Sub Software Architecture

The publish and subscribe mechanism of Pub/Sub is implemented as the PubClient for publishing sensor data from sensors, the PubSubServer for making the matches between the publications and the subscriptions and the corresponding notifications, and the SubClient for subscribing to sensor data and receiving notification. Figure 2 shows the architectural overview, which is described in the subsequent sections. All software applications are implemented in Java (Java 2 Platform, Standard Edition 1.5.0_13) [23].

PubClient

The *PubClient* application configures sensors, gathers sensor data, and transmits these data to PubSubServer via XML Remote Procedure Call (XML-RPC). It consists of three subsystems. The *PCGUI* subsystem provides a GUI with a main window showing the active sensors, and offering buttons for activating and deactivating sensors, and for editing their collecting variant. The *PCSensors* subsystem implements two sensor handlers. The PCAppleScriptHandler invokes external software sensors implemented in AppleScript that capture data from applications on a Macintosh computer, and it retrieves the captured data from the external software sensors. The PCJavaHandler invokes external software sensors implemented in Java that capture data from applications via Java calls, and it retrieves the captured data from the external software sensors. Each sensor handler provides tasks for querying sensors at a predefined frequency. The tasks are running in different non-influencing threads and can be started, stopped, and cancelled independently. The *PCPubManager* gets the captured sensor data from the sensor handlers of the PCSensors and forwards these data to the PubSubServer via XML-RPC. Due to the flexible push-paradigm it only transmits sensor data with changes. No sensor data is generated and sent to PubSubServer, if the sensor's state remains the same.

PubSubServer

The *PubSubServer* is responsible for receiving sensor data, matching them to subscriptions, and delivering notifications. It consists of several subsystems.

The *PubSubManager* is the most important subsystem of the PubSubServer—it stores subscriptions including notification preferences, matches incoming sensor data to subscriptions, and provides notifications of matching sensor data according to the notification preferences. The PubSubManger communicates with the SubClient via Java Remote Method Invocation (RMI) [24]. RMI allows easy pushing of notifications from the PubSubManager to the SubClient. The PubSubManager has three subsystems.

The *SubscriptionHandler* holds a `HashSet` of the subscriber stubs and provides the `attach()` and `detach()` methods that allow subscribers to connect or disconnect to the PubSubServer. Additionally, it holds a database of the subscriber stubs in order to store them persistently beyond

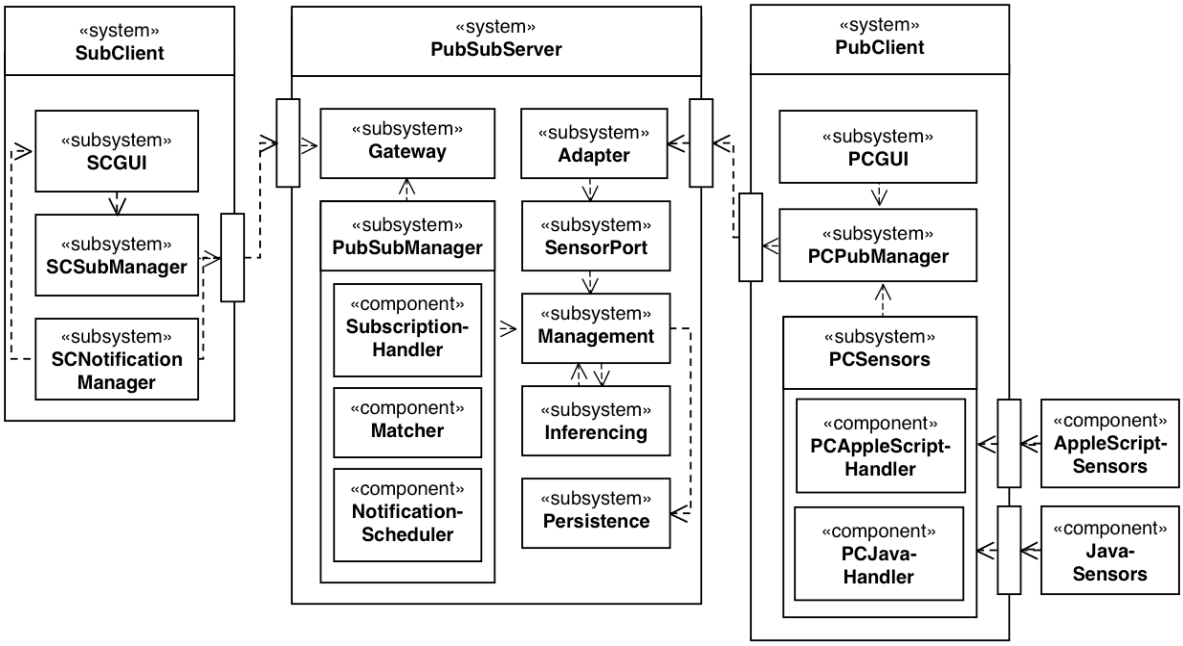

Figure 2. The architecture of the Pub/Sub infrastructure.

the shutdown of the PubSubServer. Furthermore, it holds a `HashTable` with information on subscribers with delayed notification preferences preference (i.e., a notification interval greater than 0).

When subscribers attach to the PubSubServer, the SubscriptionHandler saves the complete subscriber stub to the database; it adds the subscriber stub to the `HashSet`; and it adds a subscriber task for subscribers with delayed notifications to the `HashTable`.

The detach process of subscribers follows a similar pattern. First, the `HashTable` is checked for subscriber tasks of the detaching subscriber. The tasks found are cancelled and deleted from the `HashTable` by the SubscriptionHandler. Second, the subscriber stub is removed from the database. And, third, the subscriber stub is removed from its internal `HashSet`. When a subscriber cannot be contacted any longer, it gets detached automatically.

The *Matcher* compares incoming sensor data to the subscriptions of all subscribers. It takes the subscribers' stub from the `HashSet` of the SubscriptionHandler and compares all its subscriptions with the sensor data in a series of string comparisons. When matches are found, the Matcher checks the notification preferences. Real-time notifications are sent immediately to the SubClients via the Gateway; delayed notifications are sent to the NotificationScheduler.

The *NotifcationScheduler* receives delayed notifications, and puts them into a queue, implemented as a private fixed rate `Timer` object. This object stores the time of execution of each subscriber task. When the time of execution has come the `Timer` triggers the corresponding tasks. The task autonomously checks if they have sensor data, and notify the subscriber if sensor data are available.

The remaining subsystems and components of the PubSubServer are based on implementations of the Sensation platform and have already been published [cf. 9]. Here we just mention them. The Adapter subsystem and the SensorPort subsystem provide interfaces to publishers based on multifarious sensors and protocols (e.g., SOAP, XML-RPC, sockets). The Gateway subsystem provides interfaces to subscribers and actuator Clients via multifarious protocols (e.g., SOAP, XML-RPC, sockets). The Persistence subsystem stores and caches sensor data. The Inferencing subsystem provides algorithms for processing and comparing the stored sensor data. Finally, the Management subsystem administrates the overall PubSubServer and supports the communication among its subsystems and components.

SubClient

The *SubClient* application is responsible for managing subscriptions and dealing with notifications of matching subscriptions. The *SCGUI* component provides a user interface for specifying subscriptions, and presenting notifications. The *SCSubManager* is responsible for transmitting the description of subscriptions from the client to the PubSubServer. It implements interfaces for the `attach()` and `detach()` methods for subscribing and unsubscribing submissions on the PubSubServer. The *SCNotificationManager* is responsible for receiving notifications from the PubSubServer and for triggering further action in the actuators. It implements interfaces for the `update()` of the two methods for receiving thin and fat notifications respectively. As described above thin notification only provide feedback about a match, and fat notifications provide feedback about the matching sensor data.

B. CoLocScribe Software Architecture

The CoLocScribe media space—as described above—uses the Pub/Sub concept for awareness information disclosure and presentation facilitating smooth social interaction in and between two student sites of the CML. Accordingly, the hardware and software is distributed between the two sites. All software applications are implemented in Java (Java 2 Platform, Standard Edition 1.5.0_13) [23].

The *Workstations with the CLSPubClients* are the computers used by the students as work machines and that are extended for publishing and sensing. In the CoLocScribe setting we equipped six computers with standard hardware such as keyboard, mouse, and monitor on either site (CML-B11, and CML-HS7) with the CLSPubClients. The users configure their information disclosure by using the *CLSPubClient* client application for publishing captured information. Four different sensors capture remote and local information about the users. The BSCWSensor is implemented in Java and captures remote information from shared workspaces on the central BSCW server via XML-RPC. The CalendarSensor, the EmailSensor, and the MusicSensor are implemented in AppleScript and capture local information about calendar entries, unread email, and music played via the scripting interface of Mac OS X.

The *CLSPubSubServer* stores and manages incoming sensor data, stores and manages subscriptions, and delivers notifications based on matching subscriptions (CML-110). The central subsystems and components work as explained in the implementation of the PubSubServer described above.

The *CLSPresenters with the CLSSubClients* are the nodes for capturing and presenting the video images, and the subscription and presentation of the awareness information on each site. Both are equally equipped with the same standard hardware such as keyboard and mouse, a widescreen monitor, and a camera. Each CLSSubClient provides a GUI in the *CLSGUI*, In the GUI all windows are displayed in a heads-up display (HUD) as known from Mac OS X—that is, a semi-transparent lightweight black window without a menu. These HUDs are advanced nested interface components, which provide the advantage of elegant updates upon arrival of new information. The *CLSManagers* handle the connection to the cameras, and deal with subscriptions. The subscriptions are handled as already described in the implementation of the SubClient above.

V. RELATED WORK

The work presented here touches different research areas: sensor-based platforms, media spaces, and investigations on information disclosure.

A. Sensor-Based Platforms

Sensor-based platforms rely on events that carry data measured by sensors. They support the flexible and rapid implementation of infrastructures that assemble sensors within an environment.

Sens-ation [9] offers inferences engines to aggregate sensor input and interfere complex abstractions on sensor data. It filters and aggregates relevant sensor data. Inference engines need to be implemented and integrated into the platform for later use. The CollaborationBus editor on top of Sens-ation provides concepts for content filtering of sensor data [10]. It allows users to specify configurations of filters at run-time. However, its event handling produces huge amounts of data between the server and the clients, because all data are transferred to the client.

Khronica [17] provides both a pull mechanism for retrieving and browsing events, and a simple notification service that is able to push notifications to interested subscribers. Its publish and subscribe mechanism provides information channels that allow users to define subscription daemons for string matching. Compared to the Pub/Sub concept, Khronica provides simple filters.

Elvin [6] is an event notification service that provides quenching, which validates incoming event notifications and routes the notification to subscribed clients. Elvin is based on a thorough event model and sophisticated filters. However, it only provides push notifications from the server. *NESSIE* [22] is similar and also provides pull with an onus on the client side, but it is based on a single server instance whereas Sens-ation is distributed.

Siena [4] is also an event notification service, and is implemented on the network layer. It provides an advertise-language for the specification of interests. Due to its architecture, Siena is limited to pull events directly.

The service-oriented paradigm of Sens-ation can be found in other technologies such as the *Services Gateway Intitiative (OSGi)* framework [25], and the *Jini Network Technology* [19]. The OSGi framework provides publish and subscribe mechanisms between its services called bundles; it supports the easy integration and removal of bundles. The Jini Network Technology also provides light-weight integration of publish and subscribe services. Sens-ation provides similar software concepts; however, its main focus is on elegant and easy processing of user-generated event data.

B. Media Spaces and Information Disclosure

We show media spaces that directly address awareness and information disclosure. And we reflect on general studies of information disclosure.

Bellotti and Sellen [2] present a framework on control and feedback of information gathered in a media space environment called Ravensoft Audio/Visual Environment *RAVE*. Control means that people can decide what information is disclosed about them. Feedback is a concept to inform people when and what information is gathered and whom the information is made accessible to.

Notification Collage [7] is a media space that supports controlling confidentiality, which is the control over information that is outgoing towards others [3]. It provides a tool to configure the refresh rate of captured images from the desktop camera in the range from near live to once-per minute.

Studies in the field of information disclosure tried to elucidate questions on conditions for voluntary sharing of detailed information among people like in which situation people are willing to share or whom they want to share which information with.

A theoretical study as described in [20] on information disclosure, which was conduced in two steps revealed 40 types of information and 19 types of people to which information is disclosed differently. A second study aimed at identifying how comfortable users feel to share the previously gathered 40 types of information towards the 19 types of people. The clustered results show that the information disclosure of the subjects had limited variants with respect to the recipients of the information.

In a study [16], faces are suggested for distinct information disclosure settings for specific situations and audiences. They show that for the information disclosure behaviour the respective recipients have a considerably higher influence than that respective situation.

VI. CONCLUSIONS

In this paper we have motivated the need for powerful publish and subscribe mechanisms supporting push and pull as a basis for distributed sensor-based infrastructures. We have presented the concept and implementation of Pub/Sub as well as the CoLocScribe use case. And we have introduced related work.

The current version of the Pub/Sub concept and implementation allows users to specify preferences based on captured sensor data. It does currently not provide generic support for the specification of the format of the results. A more generic solution on the publisher's side would reduce the overall effort of developers, since the concept and implementation for the selection and presentation of the matching sensor data only have to be implemented once on the publisher's side, rather than on each individual subscriber's side.

The Pub/Sub concept as implemented in the CoLocScribe use case is perfectly scalable for the CLSPubSubServer, the CLSPubClients, and the CLSPresenters. We are currently evaluating the general scalability of the Pub/Sub concept in Sens-ation for huge amounts of events and developing novel concepts for better scalability through publish and subscribe in Sens-ation's distributed peer-to-peer environment.

In the use case the CoLocScribe SubClient provides simple representations of the data that match the subscription request and that are presented on the public screen. While this was sufficient for the use case, it would

be desirable to have more sophisticated and user-oriented forms of data presentation. Existing work on informative art can be used as a source of inspiration [5].

Finally, the adaptations in Pub/Sub and in CoLocScribe are done manually. It would be interesting to use machine learning techniques to train the system social settings, and then have the system do automatic adaptations by capturing the room situation and the present users, by inferring and comparing to the data, and by adapting the room in case a match could be found.

ACKNOWLEDGMENTS

We thank Thilo Paul-Stueve for support with the Sens-ation platform, and Maximilian Schirmer for his vital contributions to the concept and implementation of the generic publish and subscribe mechanisms. Thanks to the anonymous reviewers for valuable feedback.

REFERENCES

[1] Abowd, G.D. and Mynatt, E. Charting Past, Present, and Future Research in Ubiquitous Computing. ACM Transactions on Computer-Human Interaction 7, 1 (Sept. 2000). pp. 29-58.

[2] Bellotti, V. and Sellen, A. Design for Privacy in Ubiquitous Computing Environments. In Proceedings of the Third European Conference on Computer-Supported Cooperative Work - ECSCW'93 (Sept. 13-17, Milan, Italy). Kluwer Academic Publishers, Dortrecht, NL, 1993. pp. 77-92.

[3] Boyle, M. and Greenberg, S. The Language of Privacy: Learning from Video Media Space Analysis and Design. ACM Transactions on Computer-Human Interaction 12, 2 (June 2005). pp. 328-370.

[4] Carzaniga, A., Rosenblum, D.S. and Wolf, A.L. Design and Evaluation of a Wide-Area Event Notification Service. ACM Transactions on Computer Systems 19, 3 (Aug. 2001). pp. 332-383.

[5] Ferscha, A. A Matter of Taste. In Proceedings of the European Conference on Ambient Intelligence - AmI 2007 (Nov. 7-10, Darmstadt, Germany). Springer-Verlag, Heidelberg, 2007. pp. 287-304.

[6] Fitzpatrick, G., Mansfield, T., Kaplan, S., Arnold, D., Phelps, T. and Segall, B. Augmenting the Workaday World with Elvin. In Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work - ECSCW'99 (Sept. 12-16, Copenhagen, Denmark). Kluwer Academic Publishers, Dortrecht, NL, 1999. pp. 431-450.

[7] Greenberg, S. and Rounding, M. The Notification Collage: Posting Information to Public and Personal Displays. In Proceedings of the Conference on Human Factors in Computing Systems - CHI 2001 (Mar. 31-Apr. 6, Seattle, WA). ACM, N.Y., 2001. pp. 514-521.

[8] Gross, T. Cooperative Ambient Intelligence: Towards Autonomous and Adaptive Cooperative Ubiquitous Environments. International Journal of Autonomous and Adaptive Communications Systems (IJAACS) 1, 2 (2008). pp. 270-278.

[9] 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.

[10] Gross, T. and Marquardt, N. CollaborationBus: An Editor for the Easy Configuration of Ubiquitous Computing Environments. In Proceedings of the Fifteenth Euromicro Conference on Parallel, Distributed, and Network-Based Processing - PDP 2007 (Feb. 7-9, Naples, Italy). IEEE Computer Society Press, Los Alamitos, CA, 2007. pp. 307-314.

[11] 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.

[12] Gross, T., Paul-Stueve, T. and Palakarska, T. SensBution: A Rule-Based Peer-to-Peer Approach for Sensor-Based Infrastructures. In Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications - SEAA 2007 (Aug. 27-31, Luebeck, Germany). IEEE Computer Society Press, Los Alamitos, CA, 2007. pp. 333-340.

[13] Gross, T. and Prinz, W. Modelling Shared Contexts in Cooperative Environments: Concept, Implementation, and Evaluation. Computer Supported Cooperative Work: The Journal of Collaborative Computing 13, 3-4 (Aug. 2004). pp. 283-303.

[14] 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.

[15] 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.

[16] Lederer, S., Mankoff, J. and Dey, A.K. Short Talk: Who Wants to Know What When? Privacy Preference Determinants in Ubiquitous Computing. In Extended Abstracts of the Conference on Human Factors in Computing Systems - CHI 2003 (Apr. 5-10, Fort Lauderdale, Florida). ACM, N.Y., 2003. pp. 724-725.

[17] Loevstrand, L. Being Selectively Aware with the Khronika System. In Proceedings of the Second European Conference on Computer-Supported Cooperative Work - ECSCW'91 (Sept. 24-27, Amsterdam, NL). Kluwer Academic Publishers, Dortrecht, NL, 1991. pp. 265-278.

[18] Lynch, K.J., Snyder, J.M., Vogel, D.M. and McHenry, W.K. The Arizona Analyst Information System: Supporting Collaborative Research on International Technological Trends. In Gibbs, S. and Verrijn-Stuart, A.A., eds. Multi-User Interfaces and Applications. Elsevier, Amsterdam, NL, 1990. pp. 159-174.

[19] Microsystems Inc., S. Jini Network Technology. http://www.sun.com/software/jini, 2008. (Accessed 24/10/2008).

[20] Olson, J.S., Grudin, J. and Horvitz, E. Late Breaking Result: A Study of Preferences for Sharing and Privacy. In Extended Abstracts of the Conference on Human Factors in Computing Systems - CHI 2005 (Apr. 2-7, Portland, OR). ACM, N.Y., 2005. pp. 1958-1988.

[21] Patten, M.Q. Qualitative Research and Evaluation Methods. Sage Publications Inc., Thousand Oaks, CA, 2002.

[22] Prinz, W. NESSIE: An Awareness Environment for Cooperative Settings. In Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work - ECSCW'99 (Sept. 12-16, Copenhagen, Denmark). Kluwer Academic Publishers, Dortrecht, NL, 1999. pp. 391-410.

[23] Sun Microsystems, I. J2SE 5.0. http://java.sun.com/j2se/1.5.0/, 2007. (Accessed 31/7/2008).

[24] Sun Microsystems, I. Java Remote Method Invocation. http://java.sun.com/javase/technologies/core/basic/rmi/whitepaper/, 2008. (Accessed 11/7/2008).

[25] Tavares, A.L.C. and Valtente, M.T. A Gentle Introduction to OSGi. ACM SIGSOFT Software Engineering Notes 33, 5 (Sept. 2008). pp. 1-5.

[26] Weiser, M. The Computer of the 21st Century. Scientific American 265, 9 (Sept. 1991). pp. 94-104.