Interaction of HiveMind Application with Smart Conference System

Andrey Vasilev, Oleg Kandaurov, Eldar Mamedov, Ilya Paramonov
Yaroslavl State University
Yaroslavl, Russia
{vamonster, kandaurovoleg, ivparamonov}@gmail.com

Abstract

This paper describes the integration of the collaborative mind map editor HiveMind with the smart space application Smart Conference System. Using this reference use case we demonstrate two approaches for adapting the application to the smart space paradigm: passive and active. At the first step, HiveMind is supplied with data consumption module which was adapted to use the ontology of the Smart Conference System. The second step consists of the development of the custom ontology and the implementation of a special service application providing information according to the ontology.

Index Terms: Mind Map, Smart Spaces, Smart-M3

I. Introduction

Mind map is a tree-like diagram which consist of elements called nodes. Nodes may contain any pieces of information. HiveMind is a cross-platform mind map editor. Its most important feature is collaborative mind mapping. The initial idea of the project was to create an application for Maemo 5 platform, allowing different people to edit mind maps together with an ability to start collaboration at any moment without necessity of specially configured server or on-line service [1].

Smart Conference System [2] is a system, which aim is to intelligently assist the complicated conference process by automating the work of conference organizers and providing useful services to conference participants. It is designed and implemented according to the smart space paradigm [3], being a distributed application organized as a suit of services.

The core technology of Smart Conference System is Smart-M3 platform [4]. It is an open source solution, which provides infrastructure for development of smart space applications. Smart-M3 application consists of a knowledge base and distributed services called knowledge processors (KP), running on a variety of devices. Knowledge kept in a RDF triple storage is accessible via a semantic information broker (SIB). Knowledge processors can inquire, add, modify information, and subscribe to changes in the knowledge base.

The idea of facilitating mind mapping process by automating content generation provided us new use cases for HiveMind application. One possible use case for mind mapping is note taking during the conference. According to it, a person adds nodes, containing his/her comments or question to the node describing a topic of the talk. In order to make this notes meaningful in foreseeable future he/she needs to have additional information, such as the name of the talk he/she is listening to, or a statement describing the current slide. Most of this information can be found in Smart Conference System. By providing a reasonable way of collecting such data and displaying it in the mind map, HiveMind can assist the note taking process.

This paper describes the integration of HiveMind application and Smart Conference System. The rest of the paper is organized as following. In section II two use cases for mind mapping during the conference are described. In section III we describe what changes were made to the HiveMind to support proposed scenarios. In section IV the issue of mind map generation based on RDF triple storage is discussed and overview of proposed node structure is given. In section V we describe new support KP for extraction of the slide titles and its ontology.

II. Use Cases for Mind Mapping During Smart Conference

We developed two particular use cases for mind mapping during the conference. According to the first case, a participant uses HiveMind during the conference for taking personal notes. The application connects to the SIB of Smart Conference System and generates the initial mind map containing information about presentations and speakers. User can modify the mind map in any way one wants.

When information in the smart space is changed, the corresponding changes are made to the nodes which were automatically added earlier. For example, if the chairman changes the order of presentations, the nodes representing the talks are rearranged appropriately. When the user leaves the section, HiveMind disconnects from the smart space and no modifications are applied to the mind map afterwards. The result document contains detailed information only about the talks, the user have listened to.

The second use case could be interesting for remote participants of the conference who are unable to attend conference. In this use case, HiveMind instance in connected to the smart space throughout the conference, and it collects information about all events available. Additionally, the generated mind map can be shared, so everyone can see this information by joining the collaboration mode. To keep the information intact, the document is shared in read-only mode. Though, it can be optionally opened to remote modification forming arena for question and note sharing.

III. Integration Architecture

Smart Conference System consists of a several KPs running on devices. User KP allows presenters to change their slides directly from a mobile device during the talk, to start video playback, view other participants presentations and so on. Projector KP shows the current presentation on a screen. Whiteboard KP maintains up-to-date session schedule, which is usually differs from printed version, and makes it available to all participants. Though all

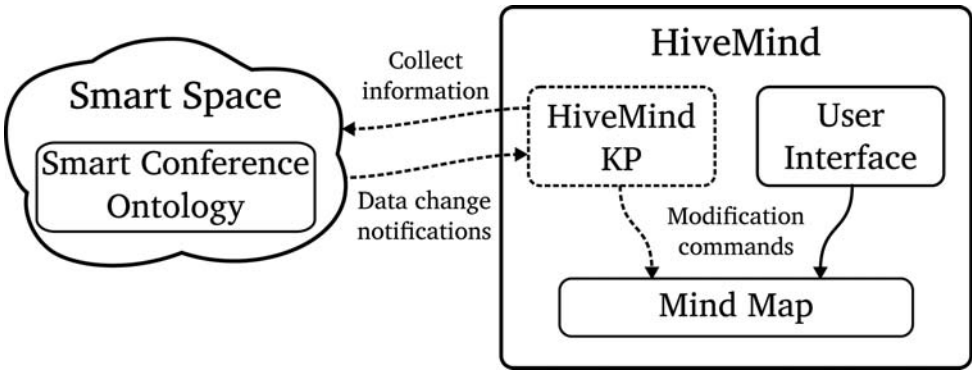

Fig. 1. Architecture of HiveMind and Smart Conference System interaction according to the first use case.

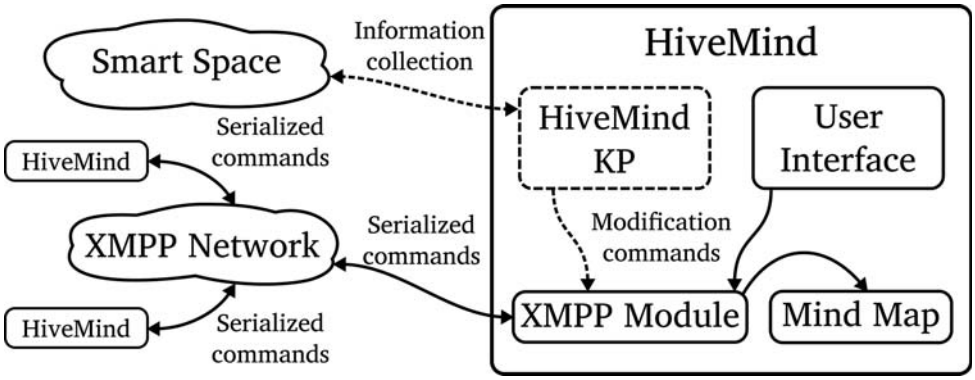

Fig. 2. Architecture of the HiveMind and Smart Conference System interaction according to the second use case.

needed control functions are present, the chairman rarely uses them due to high level of automation. They all interact by the means of the single smart space.

On a component level, integration can be described as following. We introduce a new component of HiveMind application: HiveMind KP. It connects to the smart space of Smart Conference System, collects the information from there, and subscribes to the data change. HiveMind KP creates mind map modification commands and applies them to the document. They are no different from commands formed by a user via UI. The components supporting integration and their interaction is depicted on figure 1. Dotted lines represent new components which were added to support the use case.

When information changes in the smart space, new modification commands are created and applied to the mind map. HiveMind KP keeps track of created nodes in order to apply modification commands correctly and does not force the user to follow the proposed node structure. Tracking is based on the identification field of the node, which is unique for every node on the map. The both integration scenarios assume unidirectional connection between HiveMind KP and smart space, e.g. HiveMind KP only reads data from there.

HiveMind uses the XMPP protocol and its publish-subscribe extension to enable collaboration functions [5]. When the user shares the mind map a XMPP service is created, other participants use its jabber ID (JID) to enter the teamwork. XMPP messages transmit modification commands between the service and clients. According to the second integration scenario the generated mind map is published on an XMPP network. The integration architecture is shown in figure 2. It can be formed by a local XMPP server or a public network can be used. In the latter case, remote users can connect from an arbitrary location.

IV. Mind Map Generation and Information Ranking

We decided to provide a uniform node structure acceptable for both use cases. In order to develop a suitable node hierarchy we investigated the information available in the smart space of Smart Conference System. Classes of data which knowledge processors operate with and their relationships are described in the corresponding ontology [6]. Though description of the data is usually given in terms of classes, the information in Smart-M3 platform is stored in the form of RDF triples.

It is possible to build a mind map directly out of RDF triples stored in smart space, if the ontology forms a tree. In this case, nodes are created for all subjects and objects, and all relations are represented as links between the corresponding nodes. The predicate type is displayed as a text of the link label. Unfortunately, in practice, the mind map created in

this way will probably not be a suitable structure for efficient mind mapping, because of information redundancy and structure shortcomings. For example, a user might be primarily interested in the data put in the leaf nodes, but the mind map will contain many unnecessary information. Here we face the problem of determination of relevant context data needed for the person at the moment. As the first step, we decided to rank and filter information from the smart space and therefore propose a node hierarchy.

The ontology of Smart Conference consists of several classes, including Participant, Presentation, Time slot and Projector. Participant class includes such information as name, photo, contacts, interests, and preferred language. Presentation-related information includes: name of the presentation, keywords, URL to the presentation file, and video URL. Time slot class contains time interval for presentation. Class entities form the schedule of the conference.

In order to decouple information about different talks, we place the nodes, containing talk names, as the children of the root node, therefore forming independent node trees. These nodes are sorted according to the schedule. Information about presenter, proposed duration, URLs is added to the corresponding talk node as child nodes. These nodes are created at the moment of connection to the smart space and updated when information changes. As the presentation class contains URL to the photography of the presenter, we added support for showing images in the nodes, so presenter avatar is shown on top of the corresponding presenter node.

When the presentation is shown, smart space is updated with information about current slide is being shown. Projector KP indicates the number of the slide and provides a picture of it. When the speaker changes the slide, a new node containing its number is attached to the current talk node. The nodes, corresponding to the current talk and to the current slide, are labeled with icons, so the user can easily identify them. Additionally, statistical information on how long each slide and all the talk lasted is added to the mind map. A part of the generated mind map related to the talk being presented is shown in fig. 3.

V. Content Extraction KP

The mind map generated according to the rules, stated in the previous section, is bloated with non-informative nodes, containing only the slide numbers. The user still needs to provide additional description to the slide nodes. There is no additional information provided by Smart Conference System ontology to describe the context of the slide in a more informative way. Ontology on the other hand describes the URLs to conventional files which contain the information we need. Therefore we propose to extract slide titles out of the presentation files and place them into the corresponding nodes. Though slides may be structured in a variety of a ways, almost all of them include a brief description in their titles.

Presentation file is provided by user KPs and is accessible by all other components. There are two ways for extracting slide titles out of presentation files: HiveMind can download every presentation file, extract titles and add them to the mind map, or this activity can be delegated to a separate KP, which extracts information and publishes it in the smart space, making it available to all KPs.

Main disadvantage of the first approach is the need to make energy inefficient operation of data extraction on every device running HiveMind application. Transmission of large presentation files also consumes energy and may flood the wireless network. The second approach helps to overcome this issue as only one presentation file is downloaded and data extraction is performed only once. So, we decided to implement the additional KP, Content Extractor KP, and provide corresponding ontology.

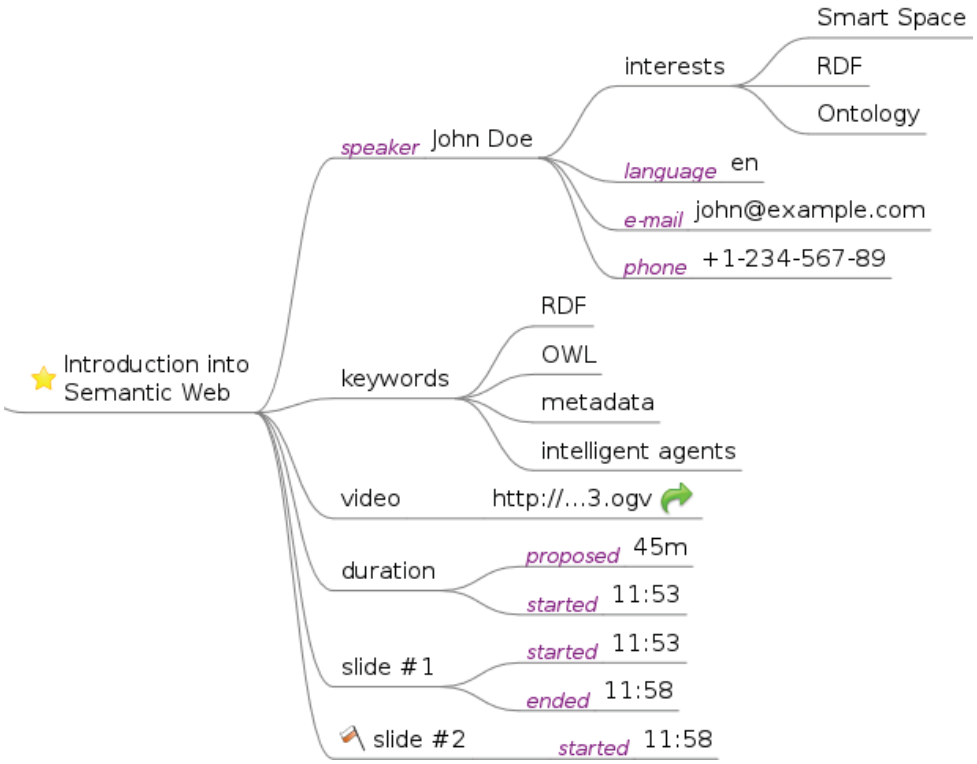

Fig. 3. Automatically generated mind map

We developed an ontology for storing the slide titles in the smart space. The ontology consists of two classes: presentation and slide. Presentation class has following properties:

- slideCount - integer - indicates the number of slides in the presentation
- hasSlide - instance of class slide - indicates that this slide belongs to the presentation

Slide class contains following fields:

- number - integer - number of the slide in the presentation
- title - string - title of the slide

As this ontology is meant to be used with Smart Conference System, presentation class properties are added to the existing entities of presentation class described in SCS ontology.

CE When KP connects to the smart space it searches for links to presentation files, downloads and processes them. If a user changes the presentation file, CE KP removes all previous slide entities from the smart space and inserts the new one. It removes all inserted records out of the smart space on shutdown, because the information found there would become invalid on presentation file change.

HiveMind KP was modified to support the developed ontology. If smart space contains the required information nodes indicating slides are filled with their names. Otherwise previous naming scheme is used.

VI. Conclusion

In this paper we described the integration of the mind map editor HiveMind with the smart space application Smart Conference System. During the integration process application was supplied with the knowledge processor module which allowed to transform information

found in the smart space into the mind map nodes. When the lack of context information was discovered we proposed and implemented a separate service, Content Extractor KP, which allows to extract slide titles from presentation files and put them into the smart space according to the developed ontology.

ACKNOWLEDGMENT

Authors would like to thank Open Innovation Framework Program FRUCT for the support and R&D infrastructure. We would also like to thank Sergey Balandin, Alexey Kashevnik, and Ekaterina Dashkova for providing feedback and guidance.

REFERENCES

[1] A. Vasilev, A. Golovchenko, A. Kulikov, I. Paramonov "HiveMind: Cross-platform Application for Collaborative Mind Mapping," *Proceedings of the 8th Conference of Open Innovation Framework Program FRUCT*, pp. 219–224, 2010.

[2] A. Kashevnik, Y. Valchenko, M. Sitaev, and N. Shilov, "Smart conference system", *SPIIRAS Proceedings*, vol. 14, pp. 228–245, 2010, in Russian.

[3] I. Oliver, "Information spaces as a basis for personalising semantic web", *Proc. 11th Int'l Conf. Enterprise Information Systems (ICEIS 2009)*, vol. SAIC, pp. 179–184, May 2009.

[4] J. Honkola, H. Laine, R. Brown, and O. Tyrkkö, "Smart-M3 information sharing platform", *The 1st Int'l Workshop on Semantic Ineroperability for Smart Spaces (SISS 2010) in conjunction with IEEE ISCC 2010*, Jun. 2010.

[5] A. Vasilev, O. Kandaurov, A. Kulikov, I. Paramonov "HiveMind Collaborative Mind Map Editor: Architecture and Implementation of Network Subsystem," *Proceedings of the 9th Conference of Open Innovation Community FRUCT and 1st Regional MeeGo Summit Russia—Finland*, pp. 217–222, 2011.

[6] Dmitry G. Korzun, Ivan V. Galov, Alexey M. Kashevnik, Nikolay G. Shilov, Kirill Krinkin, Yory Korolev "Blogging in the Smart Conference System", *Proceeding of the 9th Conference of Open Innovation Community FRUCT and 1st Regional MeeGo Summit Russia—Finland*, pp. 63–73, 2011.