Macrothink Institute™

Network Protocols and Algorithms
ISSN 1943-3581
2012, Vol. 4, No. 4

Context capturing in smart space applications

Andrey Vasilev, Ilya Paramonov

Department of Computer Sciences

Yaroslavl State University, Yaroslavl, Russia

E-mail: vamonster@gmail.com, ilya.paramonov@fruct.org,

Sergey Balandin

FRUCT Oy

Helsinki, Finland

E-mail: sergey.balandin@fruct.org,

Ekaterina Dashkova

Center for Wireless Communications

University of Oulu, Oulu, Finland

E-mail: edashkov@ee.oulu.fi,

Yevgeni Koucheryavy

Department of Communications Engineering

Tampere University of Technology, Tampere, Finland

E-mail: yk@cs.tut.fi

Received: July 30, 2012 Accepted: November 11, 2012 Published: December 16, 2012

DOI: 10.5296/npa.v4i4.2169 URL: http://dx.doi.org/10.5296/ npa.v4i4.2169

Abstract

This paper describes a study of the role and methods for context capturing in smart spaces. The collected context information is captured for further use by pro-active services provided in user-centric service spaces. Another aspect addressed by the paper is representation of the

collected context data in human-readable format. For that purpose we use collaborative mind map editor HiveMind. The developed solution for context management is illustrated by the case study. We apply the proposed method for capturing context information to the smart space application Smart Conference System and use HiveMind as user-friendly context monitoring tool.

Keywords: Context management, HiveMind, Mind Map, Smart Spaces, Smart-M3, Smart Conference.

1. Introduction

Smart space is a paradigm for developing applications in accordance with ubiquitous computing approach. The paradigm incorporates knowledge and spatial approaches [1] for implementing the next generation of user-centric applications. Smart spaces are usually attached to a specific location and allow to exchange information between applications and devices in the area, so they could cooperate for better user living or work operation. Future smart space applications shall be proactive and be able to act on behalf of a user to minimize involvement of human intelligence in all stages of service provision chain. The key enabler for that is efficient capturing and use of the context information.

The main goal of context-aware applications is to adapt to changes of an environment according to the current user activities and needs. For that purpose frameworks [2] are developed that help to collect context information, pre-process and classify it based on the actual needs of services and perform reasoning about current and future user contexts. Frameworks provide this information to applications build on top of them, so that applications can adopt its operation accordingly. As a result, applications created following such a principle provide higher level of comfort for the users in comparison with the conventional solutions.

A number of prior art studies have been done in the field by various research teams. The most notable class of applications for such technologies is the smart houses, which help elderly people, and special work environments, which provide intellectual support for daily operations. One example is development around Smart-M3 open source platform done within scope of SOFIA project [3] and by FRUCT Association [4].

Other examples of smart space services are multilingual generator of smart space access libraries [5] and multi-blogging service [6]. With the use of multilingual generator and the predefined ontology of the system, application developers may create specialized library for information manipulation in required language, therefore reducing the functionality delivery time. The multi-blogging system allows user to have a personal blog space, which efficiently operates with external conventional services and allows sharing information between such blog spaces.

The context capturing techniques applied in context-aware applications are mostly used to evaluate the higher-level state of the system and to train the reasoning engines, thought the context information is useful for the end user. Availability of the context information provides

users with more data to better understand a case and refresh memory of the past events. The form of context representation may vary from one task to another. Use of mind map for hierarchical nodes representation provides simple and user-friendly way to visualize large sets of interconnected data.

The paper presents an approach for context capturing in smart space environment and the visualization tool to assist user working with the context data. For this purpose we developed semi-formal language for describing data transformations. The proposed approach was implemented as an extension of the context visualization for Smart Conference System [7], the Smart Space application that eases conference management routines. The visualization is developed on top of HiveMind cross-platform mind-mapping application [8].

The paper is structured as following. The second chapter is focused on providing general background of the study by giving overview of the relevant prior art and related works in the field. The third chapter discusses role of context for smart space applications. In the fourth chapter we introduce the mind map formation algorithm. The fifth chapter presents a case study for context visualization in the selected application - Smart Conference System. In the end of paper we derive the main conclusions of the study and place acknowledgments and list of reference literature.

2. Related work

Information exchange in the smart space is formed according to the ontology, a high-level description of the domain knowledge and relationship between them. When operating on ontology, it is often useful to visualize the concepts and their relations. There are a lot of tools that allow representing and processing knowledge provided in different forms. The overview of existing visualization tools can be found in [9]. The general approach is to display all given information using a kind of user-friendly graphical representation. The user can select any part of the available data to study it in more detail, apply filtration to all information and so on. Such tools are created mainly for domain experts or analysts [10], who are capable of formulating custom queries to information databases and select appropriate visualization methods. The aim of VizBoard [11] application is to ease the process as much as possible for a non-expert user via exploring the user context. Though such approach eases the task of data extraction and visualization, it cannot be used for illustrating dynamic event-driven context.

The value of context capturing and playback for the users is shown by the work flow and collaboration support tools. The paper of Hunter at el. [12] demonstrates a prototype of interactive tabletop system MemTable, which provides special services for the group meetings such as activity capturing in audio and video formats, document sharing, search and review of the previous recorded events. The study shows that collaborators, who used capturing functionality, were able to recall more information from the meeting opposed to collaborators, who were using a plain paper for note taking, so the proposed approach allows recording more context data. Though being useful, this tool required to manually enable recording tools and some features were unused due to complexity of operations.

The advantages of the activity awareness to resume the state of shared activities are demonstrated in [13]. The described tool allows managing shared projects and individual tasks. It tracks the interaction of users with the system and creates several specialized views, where current collaboration spheres and activity trends can be seen. Another tool for context information gathering is presented in [14]. By the context authors presume the set of documents associated with the user activity. The key feature of developed tool to track user interactions with the document, such as opening, editing, sending different commands, so when one selects a task, the system could display corresponding set of documents. Case studies show that having context information on the activity helps knowledge workers to easily switch between tasks and increase their productivity. The disadvantage of the latter two projects is in narrowing the context gathering field to the interaction with the system, so some relevant information concerning external activities and the work environment is missing.

One of the base information sources for context data is sensor readings. There are special tools for recording and playback of all gathered readings from the sensors. PyViz project [15] is devoted to capturing and playing back the sensor readings done in a smart house-like environment. The users need to create the structure of the house inside the application, place sensor visualizations on corresponding places and link them with data sources. Cheun at el. [16] propose a special framework for capturing mobile context of the user. The mobile context is generated using sensor inputs available for the mobile phone, e. g. accelerometer, absolute position of the person, light sensors and so on. The visualization application, implemented as a web service, allows browsing the sensing history of the single person or a group, or some extrapolation for a period of time. The approach presented in these works should help to capture useful information about end users for the future context-aware services. Though providing a lot of context information, such approach requires user to manually associate raw data with substantial events.

3. Role of context for smart space applications architecture

The below discussed application is based on Smart-M3 platform. Smart-M3 is an open-source information sharing platform, which provides infrastructure for creation of smart space applications. It is based on a whiteboard architecture model, where information is shared between smart agents. Platform provides knowledge storage named semantic information broker (SIB), which stores the shared data and provides API for the data modification. Distributed agents that interact with the SIB are called knowledge processors (KP). They may run on any device, providing different kind of services, ranging from sensor data collection to assisting the end-users. In order to achieve better coordination with each other, KPs usually place data according to the defined ontology thought Smart-M3 does not restrict data manipulation in any way.

The SIB store data in the form of Resource Description Framework (RDF) triples. The Web Ontology Language (OWL) is usually used to describe the structure and semantics of the data graph. The use of such instrumentation allows KP not only to consume predefined data, but also to extract required information from data described by divergent ontologies.

For example, KP may need information on the number of heat sources. Such information can be directly acquired from the heat sensor data readings, or roughly computed as the sum of the number of people in the room and the number of equipment, that operates right now and consumes more than 500 watt.

The context for applications, which were built on top of the platform, is defined explicitly as the information stored in the space. The KP may adapt behavior to better suite current user needs. The process of information inference based on available data is called context reasoning. Common approach for context reasoning is to provide some sort of mapping from information space to the smaller number of supported KP behaviors. Usually it is achieved using artificial intelligence techniques like Bayes networks, neural networks and so on. KP then only needs to read the evaluated information and modify the operations accordingly.

Mind map provides good abstraction for graphical representation of the complex data sets. For example, mind maps can help to hierarchically structure and represent collected context data in human-readable way. The mind map nodes are organized to the groups of logically connected data elements, so this format is easy to understand and interpret even for complex and large sets of data. There are a number of approaches to form mind maps, so the most suitable approach can be selected for a given case. For any selected node in the mind map we can easily see a set of neighboring and connected nodes, which provide the node context. So even if an application or user cannot understand or recall information related to some information node, the context information can be used to refine the case.

The mind map is an efficient tool for capturing information about single events as well as about series of events distributed in time or space. Nodes describing repeated events could be aligned according to their appearance from the most aged to youngest ones or aggregated in some way.

The events captured in the form of the mind map could be applied for KP behavior modification. One possible way is to acquire information related to utilized resources or related to clients of the KP. Using this information KP could disable some of provided services, as they are not or less demanded, or apply some scheduling scheme for them. Another way is to gather information about all services of some kind, which have been registered in the network, so this way KP could optimize its operation according to the usage context.

4. Mind map formation algorithm

In the current chapter we describe a method on how to capture the context information from smart space applications in the form of the mind map.

4.1. General idea

4.1.1 Define structure of the resulted mind map based on the ontology and required context relations.

The ontology of a Smart Space application in a general case cannot be directly represented as a mind map. Firstly, the ontology is created to ease the processing by computers, so it structure could not correlate with the user needs. Secondly, it may contain some technical information, for example about the network location of application components or intercommunication agreements, which are not interesting for the end-user. In the end, the ontology describes an arbitrary graph, which could contain closed loops, so it cannot be presented as a tree-like structure.

During the design of the mind map structure, the main focus should be held on the data, which is interesting to the user, e.g. user-recognizable context. The descriptive data should be attached to the nodes, containing required information. This way user will easily find required information on the mind map. One more thing to consider during the design process is the nesting of the nodes, describing meaningful events. Some events last long enough to include other events, so information about nested events should be attached to the nodes, which describe long-lived events. Such composition should help to correlate the events and their context information.

4.1.2. Describe mapping of the smart space contents to the mind map according to the rules of the description language.

There are two different approaches to store and represent information coming from the smart space. According to the first one, the mind map should contain only the last value of the data in the smart space, this way user is informed with the current information, but could not detect the dynamics of data change. Another approach is to store each value of data, coming from smart space as a separate node, so the history of data change is visible to the user, but mind map fills with lots of nodes.

The semi-formal descriptive language discussed in Sec. 4.2 allows formulating rules for converting information stored in the smart space in the form of RDF triples into mind map nodes. Such description allows making a preview of generated mind map and estimate the number of queries needed to fill the structure. In future, this structure could be used as an input for a mind mapping visualization software.

4.1.3. Define events in the smart space to be used for mind map updates.

The ontology used by the smart space application only describes the data used by modules to communicate with each other, but not the way they modify the data, e.g. there is no description of concrete communication protocol, that build upon data change sequences. One way to handle the uncertainty is to make subscriptions for every triple required to build mind map, but the subscription operation is expensive, so such approach can-not be used. We propose to study the interaction of the application with the smart space to depict required RDF triples, which modification can be used as event to modify the sub-tree of the mind map. The sub-tree should be defined with the use of descriptive language. When selected event occurs, corresponding part of the mind map is updated with the current data from the smart space, or new nodes are added to the mind map.

4.2. Semi-formal language for mind map representation

We use a semi-formal description to determine the structure of the resulted mind map. It is used for initial mind map generation as well as for mind map updates during the smart space applications execution.

The description consists of a set of lines, each of which describes a node or a group of nodes generated as a result of a single or multiple triples query. In the rest of this section we consider the main syntax constructions of the descriptive language.

Static nodes with no dependency on information from the smart space are presented as lines with fixed contents in single quotes. Indentation is used to show parent-child relationship between the nodes. If an edge between two nodes contains a label, the text of this label enclosed with square brackets precedes the text of the descendant node. For example the pseudo-code

```
'Book'
   'Chapter 1'
   ['interesting'] 'Chapter 2'
```

denotes the mind map with the root node ‘Book’ and two child nodes of the root having the text ‘Chapter 1’ and ‘Chapter 2’ respectively.

For nodes generated dynamically from the information retrieved from the smart space we use the following syntax:

```
selector : node_contents
```

Here *selector* is a construction, defining triples to be retrieved from the smart space, and *node_contents* is the contents of the node to be generated for each of these triples.

The simplest construction for a selector is the following:

```
(subject, predicate, object)
```

It stands for a selector of triples, with the corresponding *subject*, *predicate* and *object*. Each of the triples matching the selector produces a node. All these nodes become the children of the nearest parent node in the hierarchy. *Subject*, *predicate*, and *object* can be a literal in single quotes, an expression, or a new identifier not defined along the path from the current node to the root. In the latter case, the corresponding part of the query is set to a placeholder, which matches any value. Moreover, assigning the identifier equals to the value of the corresponding component retrieved as a result of the query everywhere in the subtree of the node it originally appears. To make this clear, consider the example

```
'Library'
   (book, '#type', 'book') : book
      (book, 'contains', section) : section
```

and suppose, the smart space contains the following triples: (‘book1’, ‘#type’, ‘book’), (‘book2’, ‘#type’, ‘book’), (‘book1’, ‘contains’, ‘introduction’).

According to our description, after root node creation we inquire for all triples having a predicate '#type' and an object 'book'. As a result, we create two descendants of the root node 'Library' with the text 'book1' and 'book2' respectively. Then we start processing the subtree with the root node 'book1' and assign a value 'book1' to book identifier. Then we execute a query for all triples having 'book1' as a subject and 'contains' as a predicate and receive the triple ('book1', 'contains', 'introduction') as a result. According to our description, we create a node with text 'introduction' being a descendant of the node 'book1'. Then we start processing the 'book2' subtree, and assign a value 'book2' to book identifier. In this case the query returns nothing, because there is no triples with subject 'book2' as a subject and 'contains' as a predicate in the smart space. From this example it is clear how the proposed description allows narrowing the filtering condition according to the context of the current subtree.

More complex form of the selector may include several simple selectors combined with the '&&' operator. It leads to a multiple query execution and can be used to reflect sophisticated relations between the data stored in the smart space. For example, consider the following description:

```
'Library'

   (book, '#type', 'book') && (book, 'title', title) : title
```

The first part of the selector is the same as in the previous example, but the second one inquires one more set of triples for each of the retrieved books. These triples are supposed to contain the title of the corresponding book. As a result, we fill the mind map not with ids of the books, but with their titles.

For the sake of clarity we use the following simplified syntax for the queries similar the last one:

```
'Library'

   (book, '#type', 'book') : book.title
```

The other valid constructions include context markers and arbitrary function invocations. Context markers are important to reflect information, dynamically changed during the functioning of the smart spaces application. They are considered in Sec. 4.3.

Function invocations are denoted by the function name embraced with curly brackets (e.g., {current_time}). We have no implications on what these function can do, except that they can have read-only access to the mind map being generated. In fact, these functions can be specially defined for each concrete situation, being a simple mechanism for extension of the proposed mind map generation approach.

4.3. Steps of Mind Map Formation Process

In the previous section we defined the description language, which is used for definition of mind map formation rules. In this section we point out how this description is utilized to construct the mind map, reflecting the information, which is stored in the smart space and

processed by applications.

The corresponding procedure contains two steps. The first is executed once, and the second is executed several times, once for each of the events registered for mind map updates. The steps are following.

1. The description is read line-by-line. For each of the selectors the corresponding query to the smart space is executed. The triples returned by the query are used to create new nodes and subtrees in the mind map. As a result of this step, we retrieve the information already contained in the smart space. When we are interested only in presentation of static data in the form of mind map this step can be sufficient.

2. The description is read line-by-line. For each of the selectors we check if the added/modified triples received by subscription match the selector. If so, we create new nodes for all added triples, update contents of the nodes corresponding to the modified triples, and continue processing the subtree of the current node using the same rules. Otherwise, we do not process the descendant nodes and move on to the nearest node of the same or higher level in the hierarchy.

The proposed approach allows acquiring any kind of context information in the smart space in either static or dynamic mode and rearranging it according to the predefined mind map structure.

5. Mind map as a context visualization for Smart Conference System

This section describes the use of proposed approach for context visualization of Smart Conference System.

5.1. Target user models

As the first step of developing the context capturing for Smart Conference system the models of target users have been formalized. There were depicted two target user groups: the participants of the conference, who want to take notes during presentations, and the presenters, who are interested in how much time they spend on the each slide during the talk. The main target of the first group is to create a meaningful record of the presented information and events that took place throughout the presentation, for short-time revision at question section and for long-time revision later-on. The second group is interested in having materials for an analysis of their presentation.

5.2. Smart Conference System

The Smart Conference System is created using Smart-M3 platform, therefore it consists out of several KPs, that interact with each other by the means of shared information space. The KPs are: whiteboard KP, projector KP, user KP. The first one is responsible for maintaining conference schedule, tracking the availability and absence of speakers and so on. The projector KP displays the slides of the speaker and provides screen shoots of the slides to other elements. The user KP allows accessing information about presenters and speakers,

view current presentation slides and control the flow of the one's presentations.

The ontology of the Smart Conference System [7] consists of such concepts: user profile, presentation profile, time slot, whiteboard, projector, current slide. Presentation, user profile and time slot concepts are most valuable for the end user, because they describe the visible parts of the conference process. The relationship between concepts is shown on Fig. 1.

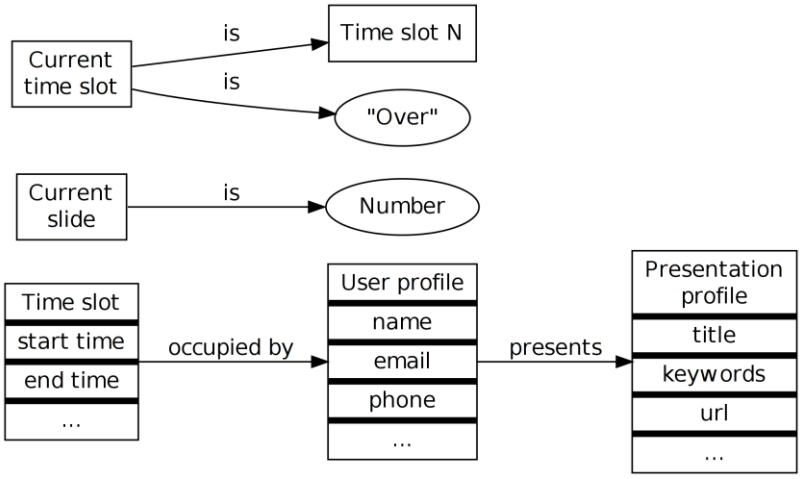

Figure 1. Part of the Smart Conference ontology describing user sensed events.

As you can see there is a quite lot of information describing the big events, such as the start of the talk, presenter entrance and departure, but the only information published into smart space about slide being presented is the slide number. Such information cannot be used to efficiently visualize context, so content extractor KP was introduced [8]. The main aim of the KP is to extract additional information about slides from presentation files. The current implementation of the content extractor KP publishes only slide names. The ontology of KP extends the ontology of the Smart Conference System. The additional elements to the base ontology are shown on Fig. 2.

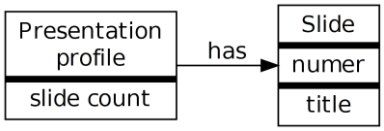

Figure 2. Content extractor KP extension to Smart Conference ontology.

5.3. Algorithm appliance

This subsection describes the appliance of context capturing algorithm to the task of context capturing of Smart Conference System.

5.3.1. Mind map structure

According to the depicted target user roles, the mind map structure for recording context information was formed. The mind map is formed as a set of base elements, which structure is presented on Fig. 3.

The key event, which is visible by the conference visitor, is the talk, so node containing the name of the presentation is the root of the element. The descriptive data for the presentation, such as speaker name and his contact info, keywords, duration, is put in the child nodes of the root element. The time of presentation start and end is attached to the node "duration", when corresponding event occurs. The next noticeable event is the change of the presentation slide. When slide is shown first time, the new node containing the name of the slide is added to the mind map. The time of slide appearance and disappearance is recorded in child nodes of the slide node. In order to simplify understanding of the mind map the links between nodes may have descriptive labels, for example the link between root node and speaker name may have the 'speaker' label.

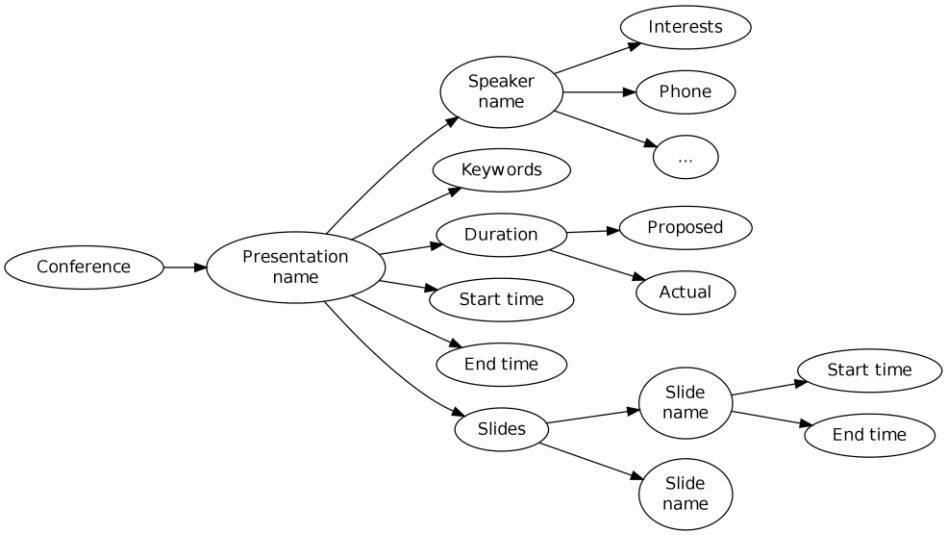

Figure 3. Mind map structure for Smart Conference System.

5.3.2. Mapping rules

The ontology of the Smart Conference System cannot be directly mapped into the proposed mind map structure. The most notable differences are: the direction of the connection between user profile and presentation profile is different; the mind map does not contain nodes, which store unique identifiers of the entities.

The mapping rules are presented on Fig. 4. The detailed line-by-line explanation of this listing is presented below.

1. The root node contains the text 'Conference'.
2. Each of the root node descendants corresponds to a single presentation. To retrieve necessary information from the smart space we inquire for all triplets having 'Title' as a predicate. The contents of the object field of these triples become the text of the corresponding node.
3. For each talk we inquire for a triplet having the talk name as an object and 'presents' as a predicate (there is the only one such a triplet according to the conventions of Smart Conference application). As a result, we retrieve the speaker id from the subject field of the triplet, and inquire for another triplet, containing this id as subject, and a

'name' literal as a predicate. The object field of the resulting triplet becomes the text of the node.

4. We inquire for all the attributes regarding the speaker in question. To do this the query matching the speaker id as a subject and arbitrary predicate and object values is executed. After that we create subnodes of the speaker node, and fill them with the attributes retrieved. Attribute names become the titles of the corresponding edges.

5. This step is similar to the previous one. We retrieve all the attributes related to the talk and fill in the talk subtree with values of these attributes.

6. The selector defined in this line matches the non-empty set of triples, when the current timeslot is occupied by the speaker of the current talk. It is activated by the presentation starting event, so we create a node labeled 'started', fill it in with the current time, and set the $current_talk_context to acquire the current subtree. The latter is required to accumulate the information emitted to the smart space during the current talk (e.g. information about shown slides) directly to the current subtree.

7. When the triple ('TS', 'is', 'over') appears in the smart space, we create a descendant node of the talk node with the 'ended' label, and fill it in with the current time. The $current_talk_context marker guarantees that only the current presentation subtree is affected. After that, we unset the $current_talk_context marker, as the corresponding talk has finished.

8. This line defines a parent node for subtree containing information about the talk duration.

9. The proposed talk duration is taken directly from the smart space. In this line "+" is used as a concatenation operator.

10. Similarly to the 'ended' node creation in the line 7, we create a node containing overall presentation duration (it is calculated with the use of calculate_overall_time function).

11. In the subtree corresponding to the current talk the 'slides' node is created.

12. When the current slide is changed, we extract its title and create a node in the 'slides' subtree, which contains the title.

13. Right after the previous step we create a descendant node labeled 'started', containing slide change time, i.e. current time at the moment of the node creation.

14. The selector in this line matches the triple corresponding to the slide change, similarly to the one of the line 12. But it is done only in when $current_slide_context is set for the current subtree. It means that the node defined in this line is added when the current slide is substituted with the next slide, and this is different from the case processed in the line 12. In this case we create a node labeled 'ended' and fill it in with the current time. After that we unset $current_slide_context denoting the end of the current slide displaying.

```
1 'Conference'
2    (talk, 'Title', talkName) : talkName
3       (speaker, 'presents', talk) : speaker.name
4          (speaker, speakerAttr, speakerAttrValue) :
           [speakerAttr] speakerAttrValue
5       (talk, talkAttr, talkAttrValue) : talkAttrValue
6       (speaker, 'presents', talk) && (timeslot, 'is_occupied_by', speaker)
        && ('TS', 'is', timeslot) ['started'] :
        {current_time} $set_current_talk_context
7       $current_talk_context && ('TS', 'is', 'over') :
        ['ended'] {current_time} $unset_current_talk_context
8       'duration'

9          ['proposed'] talk.duration + " min."
10         $current_talk_context && ('TS', 'is', 'over') :
           ['actual'] {calculate_overall_time}
11      $current_talk_context 'slides'
12         ('current_slide', 'is', current_slide) && (slideInfo, 'number',
           current_slide): slideInfo.title $set_current_slide_context

13            'started' {current_time}
14               $current_slide_context && ('current_slide', 'is',
                 next_current_slide) : 'ended' {current_time}
                 $unset_current_slide_context
```

Figure 4. The mapping rules for Smart Conference System ontology.

5.3.3. Mind map modification events

We define three types of events managing mind map updates.

The first one relates to a situation, when information about the presenter or the presentation is updated in smart space either by the whiteboard KP or by the user KP, the triple (talk, ‘Title’, talkName) is updated, so one subscription for the triple modification is sufficient to track the addition and modification operations for the corresponding nodes.

The other two events are defined in the form of subscription to the triples, having either ‘TS’ or ‘current_slide’ as a subject, and ‘is’ as a predicate. These events allow accumulations of the information regarding talks and slides changes during Smart Conference System operation.

5.4. Algorithm application example

The proposed approach has been successfully applied for development of the context capturing and visualization in the Smart Conference System with use of HiveMind mind mapping application. The system was successfully demonstrated a number of times, e.g., at the smart spaces sections on 10th and 11th FRUCT conferences, where it was actively used by the participants and proved to be useful for the target user groups.

During the conference any participant could start the HiveMind application and get information about presentations he or she has seen. One could easily supply generated mind map with specific nodes avoiding the routine of manual addition of structural nodes. Another instance of the application was run by the organizers. The mind map generated by that application contained information about all talks given during the smart spaces section. This map have been used by the presenters and organizing committee for speech timing assessment.

The example of the generated mind map is given on Fig. 5. The example is based on information gathered during the smart spaces section on 11th FRUCT conference. The mind map provides information about the talk (title, start time, proposed duration), presenter (interest list, phone number, e-mail, etc.) and each slide shown so far (title, duration metrics). The map contains the description of John Doe talk being presented at the moment, so there is no information about the end of the presentation or the last slide.

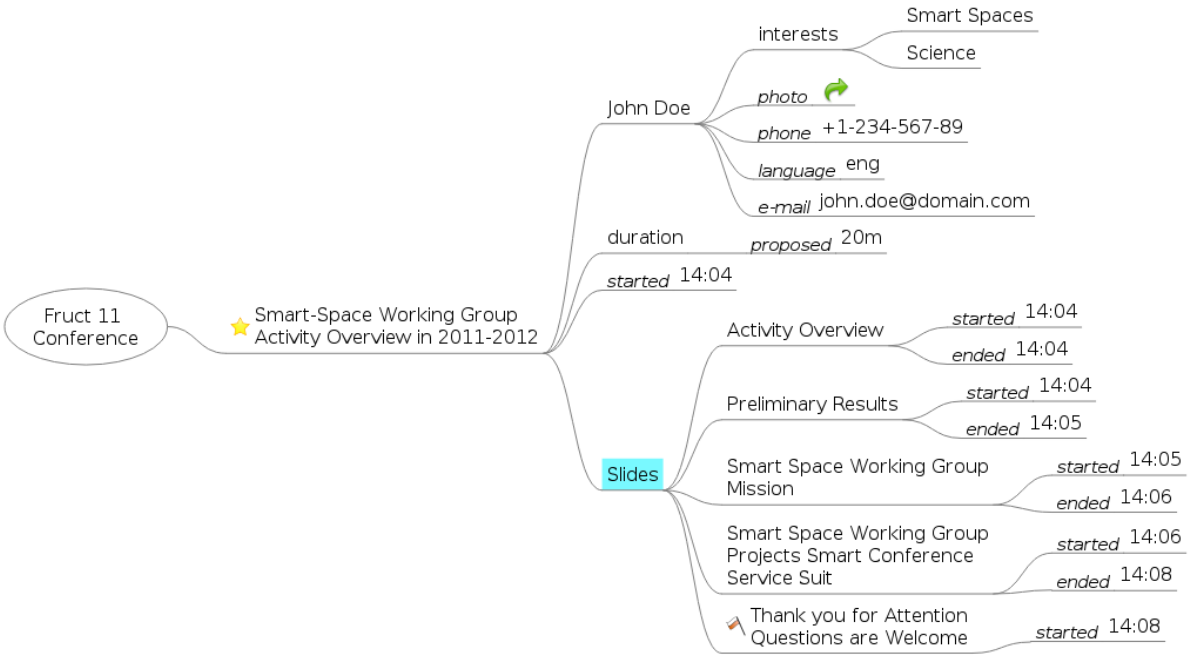

Figure 5. The example of generated mind map describing the talk of John Doe.

6. Conclusion and Future Proposals

The paper proposes semi-formal language for translating smart space information into the mind map format. The language is rather expressive and allows acquiring static as well as

dynamic context information, based on the event capturing in the smart space. The developed language can be used not only for mind map formation, but it also can be adopted for use with different visualization and context reasoning techniques. The ability to reason about current situation and user intentions give a possibility to design new types of applications that will be able to proactively adopt its operation according to current and expected user needs.

The developed language was presented as a semi-formal, as the task of context information capturing and visualization cannot be fully automated due to high complexity and probabilistic nature of the user intentions. Also applicability of the proposed approach is restricted by a subset of context-aware applications.

The developed methodology allows collecting all relevant context information and generates mind maps of top of it. Availability for such mind map is beneficial in many cases, e.g., when a user is interested to review contextual events and find possible correlations or when one is interested in getting all possible information about a particular event.

Future proposals include two main directions. The first one constitutes in elaboration of the field of applicability and definition of the more specific use cases for the proposed approach (e.g., end user context capturing, KP context capturing with the subsequent reasoning etc.). Imposing additional restrictions, correlated to these use cases, would allow increasing the formalization level of the algorithm and potentially applying it automatically in the corresponding cases.

The second direction relates to removal of the obsolete context data from the mind map. For now, the proposed semi-formal language does not allow to remove the nodes of the mind map, which have become irrelevant at the certain moment. That is why adding the node removal constructions to the proposed semi-formal language and corresponding modifications to the algorithm would be beneficial.

Acknowledgment

The study was done within scope of FRUCT Smart Spaces working group activities and project targeted in development of distributed modules for smart room environment at Petrozavodsk State University and authors would like to thank Open Innovation Association FRUCT, ENPI Karelia KA-179 project and the project “Cross-platform future services - technologies of smart space and internet of things” funded by grant of Federal Purpose-Oriented Program “Scientific and scientific-pedagogical personnel of innovative Russia in 2009-2013” for the provided support and R&D infrastructure.

References

[1] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A. and Riboni, D., “A survey of context modelling and reasoning techniques”. Pervasive and Mobile Computing, Vol. 6, Issue 2. pp 161-180. 2009. http://dx.doi.org/10.1016/j.pmcj.2009.06.002

[2] Keling, DA, Dalmau, M. and Roose, P., "A Survey of Adaptation Systems". International Journal on Internet and Distributed Computing Systems, Vol. 2. Issue 1, pp. 123-140. 2012.

[3] Official web site of SOFIA project, Available at: www.sofia-project.eu. 2011.

[4] Balandin, S., Oliver, I. and Boldyrev S., "Distributed Smart Spaces M3 Platform for Building Professional Social Networks with Seamless PCs and Mobile Access", International Journal on Advances in Intelligent Systems, Vol. 3, No. 1&2. pp. 141-149. 2010.

[5] D. Korzun, A. Lomov, P. Vanag, S. Balandin, J. Honkola "Multilingual Ontology Library Generator for Smart-M3 Information Sharing Platform", International Journal on Advances in Intelligent Systems, Vol. 4, No. 3&4, 2011, pp. 68-81. 2011.

[6] D.G. Korzun, I.V. Galov, S.I. Balandin, "Proactive Personalized Mobile Multi-Blogging Service on Smart-M3", Journal of Computing and Information Technology. Vol. 20, No. 3, pp.175-182. 2012. http://dx.doi.org/10.2498/iti.2012.0405

[7] Korzun, D., Galov, I., Kashevnik, A., Shilov, N., Krinkin, K. and Korolev, Y., "Integration of Smart-M3 Applications: Blogging in Smart Conference". Smart Spaces and Next Generation Wired/Wireless Networking, LNCS 6869, Springer. pp. 51-62. 2011.

[8] Vasilev, A., Paramonov, I., Dashkova, E., Koucheryavy, Y. and Balandin, S., "Interaction of HiveMind Application with Smart Conference System", IEEE International Conference on Communications (ICC WS - SCPA), Ottawa, Canada, 2012.

[9] Lanzenberger, M. and Sampson, J. and Rester, M., "Ontology Visualization: Tools and Techniques for Visual Representation of Semi-Structured Meta-Data", Journal of Universal Computer Science, Vol. 16, Issue 7. pp. 1036-1054. 2001.

[10] Catenazzi, N. and Sommaruga, L. and Mazza, R., "User-friendly ontology editing and visualization tools: the OWLeasyViz approach". Information Visualisation, 13th International Conference. pp. 283-288. 2009.

[11] Voigt, M., Pietschmann, S. and Meißner, K., "Towards a semantics-based, end-user-centered information visualization process". Semantic Models for Adaptive Interactive Systems. Lisbon, Portugal, February 14-17, 2012.

[12] Hunter, S., Maes, P., Scott, S. and Kaufman, H., "MemTable: an integrated system for capture and recall of shared histories in group workspaces". Proceedings of the annual conference on Human factors in computing systems, pp. 3305-3314. 2011. http://dx.doi.org/10.1145/1978942.1979432

[13] Ardissono, L., Bosio, G. and Segnan, M., "A Visualization Model Supporting an Efficient Context Resumption in Collaboration Environments". Advances in User Modeling, Vol. 7138. pp. 5-17. 2012. http://dx.doi.org/10.1007/978-3-642-28509-7_2

[14] Kersten, M. and Murphy, G.C., "Task Context for Knowledge Workers", Tasktop Technologies. Available at: http://tasktop.com/pdfs/docs/publications/2012-05-aaai-Task%20Context%20for%20Knowledge%20Workers.pdf. 2012.

[15] Thomas, B.L. and Crandall, A.S., "A demonstration of PyViz, a flexible smart home visualization tool". Pervasive Computing and Communications Workshops (PERCOM Workshops). pp 304-306. 2011. http://dx.doi.org/10.1109/PERCOMW.2011.5766889

[16] Cheun, D.W., Kim, M.K., La, H.J., Bae, H.J., Keum, C.S. and Kim, S.D., "A Practical Framework for Comprehensive Mobile Context Visualization". E-Business Engineering (ICEBE), IEEE 8th International Conference. pp. 201-206. 2011. http://dx.doi.org/10.1109/ICEBE.2011.24

Copyright Disclaimer