Context Gathering in Meetings: Business Processes Meet the Agents and the Semantic Web

Ian Oliver[1], Esko Nuutila[2] and Seppo Törmä[2]

[1] Nokia Research, Itämerenkatu 11-13, 00180 Helsinki, Finland

[2] Helsinki University of Technology, PL 5400, FI-02015 TKK, Finland

Abstract. Semantic Web, Space Based Computing and agent technologies provide for a more sophisticated and personalised context gathering and information sharing experience. Environments which benefit from these properties are often found in direct social situations such as meetings. In the business world such meetings are often artificially structured according to a process which does not necessarily fit the physical (and virtual) environment in which a meeting takes place. By providing simple ontologies to structure information, agents and space-based information sharing environment with reasoning capabilities we can provide for a much richer social interaction environment.

1 Introduction

In practice, the complex cognitive tasks faced in organizations – including coordination, problem solving, design, planning, ideation, and decision making – need to be tackled in *meetings* where different perspectives and disciplines can be combined and shared understanding can be built [1, 2]. In these kind of content-centered meetings, the formal structure - or predefined process - is much less important than the wealth of information contained in a meeting. The content and development of the information has a significant impact on the course of the meeting, while the official agenda often provides only superficial names to the phases of the conversation.

A typical meeting both *uses and produces* information. The participants need to share existing information with each other, often in an ad hoc manner and interact with the meeting in varying ways as situations arise. This includes varying amounts and configurations of typical computing and display equipment such as mobile phones, displays, personal computers etc.

Increasingly the 'conversation' of the meeting is also supplemented with slide presentations, documents, diagrams, and other visual material as well as references to material not explicitly included in the meeting. During the meeting new information is usually produced, even if it not always explicitly written down. It should be noted that a large part of the information is related to the context of a meeting itself, its participants, as well as the materials presented. Much of this information could nowadays be automatically gathered for later use.

There is a lot of diversity in the actual arrangements of meetings. At one extreme are well-organized meetings kept in special meeting rooms. At the other extreme are

Oliver I., Nuutila E. and Törmä S. (2009).
Context Gathering in Meetings: Business Processes Meet the Agents and the Semantic Web.
In *Proceedings of the Joint Workshop on Advanced Technologies and Techniques for Enterprise Information Systems*, pages 5-16
DOI: 10.5220/0002195400050016

ad hoc gatherings to tackle urgent problem that can take place in office rooms or even cafeterias. Regardless of the type, it is nowadays typical to have lots of available displays. There can be shared displays such as video projectors or large screens as well as personal displays such as laptop computers or PDAs.

In this paper we will focus specifically on the problem of sharing (media-based) information in a easy and flexible manner during a meeting. We are especially interested in increasing the interaction among participants by making the granularity of presentations smaller and by allowing simultaneous presentation of information. We will present a case study of this functionality.

Our solution builds on the Semantic Web and context gathering and reasoning techniques [3–5] that enable rich information to be stored, linked and reasoned about from a variety of sources. In this paper we show how the interaction between a few simple ontologies and users of those ontologies frees a meeting scenario from the confines of a strict process enabling ad hoc linking and reasoning about the information being gathered in that context.

2 Space-based Computing Framework

The Sedvice architecture [6, 7] supports distributed agent based interactions over a declarative space-based infrastructure [8]. A diagrammatic representation of the architecture is shown in figure 1.

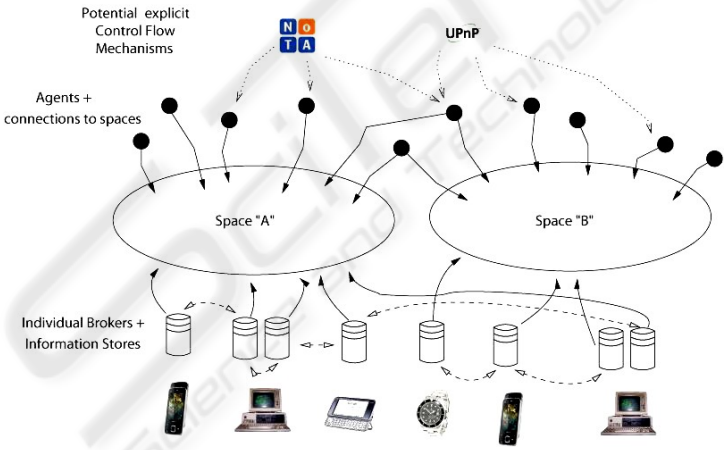

Fig. 1. An Example Implementation Configuration.

Spaces are virtual entities which contain the information; these are represented by a distributed, totally routable set of information brokers (known as SIBs). A Space at its simplest contains: listeners for one or more transport/connectivity protocols (eg: HTTP, TCP/IP, UMTS, SOAP etc), the logic for processing messages, policy/security and handling subscriptions, a deductive closure mechanism and an information store (an RDF store in our case).

Computation is placed primarily in agents [9] which interact with the space (physically via individual SIBs). The responsibilities of agents range from user-interaction to reasoning and performing tasks such as truth maintenance, belief revision, information consistency management etc. Agents are typically anonymous and autonomous from each other. Control flow is either provided through first-class mechanisms inside the spaces themselves or via agents communicating externally through specific control-based environments such as the Network on Terminal Architecture[3] or Universal Plug and Play[4]. Agents are necessarily anonymous and independent of each other unless the agent specifically shares information about itself through the space.

Information is stored as a graph using the Resource Description Framework (RDF) [10]. The semantics of the information is implied by the writer by way of what typing information is provided, the amount of information and what ontologies are used. The reader of any piece of information interprets the information according to its own criteria which is influenced by the typing, tagging and ontologies. Information is semi-structured deliberately allowing inconsistency and ad hoc expansion of the information schemata.

What the user perceives as being an application [11] is constructed from a number of independently operating agents which — if they have user-interfaces — may be grouped together visually (or by some other means, eg: audio) so that their combined results may be perceived as a single whole.

3 Simple Contextual Model of a Meeting

We present a simple contextual model of a meeting, which considers two major aspects [12]. The first aspect captures the notion of meeting as an assembly of group of people — the participants — gathering together at a specific time and place to handle a specific topic using a set of documents. The second aspect captures the dynamic notion of a meeting as set of acts that are done in a meeting using equipment. This model can easily be extended to capture other aspects.

For an implementation of this contextual model we create a number of individual agents that interact with small subsets of the ontologies defined below.

Within a meeting there exist an agenda, notes on the meeting, documents or presentation to be made in that meeting and the participants of that meeting etc. Outside of this we have notions of physical devices that might exist in the location of a meeting (though we don't enforce physical presence). We also show how these aspects, that of the meeting and its presentations and that of physical devices can be combined.

3.1 Ontologies

The basic information defining a meeting itself can be fairly simple: a set of participants, documents and an agenda. Within a meeting however the actual information structure can be extremely complex involving individual comments, ad hoc changes to

[3] www.notaworld.org
[4] upnp.org

the agenda, participants etc and then further links not only within the meeting information but outside of that to provide additional context and content to the meeting. For example, the participants of a meeting in many tools are just names; using the Semantic Web we can extend this out to business contacts (eg: LinkedIn, Facebook etc[5]) and the possibilities that arise from including that information [13].

Our basic design consists of two separate ontologies; we do not espouse the 'one true ontology' view and it is highly likely that differing formats for various aspects of the whole system will be used and thus need to interoperate [14].

In Figures 2 and 3 we present the classes that are used in these ontologies. We differentiate between the ontologies when necessary in this text by prefixing with shortened namespace declarations: `Nokia` and `TKK` respectively.

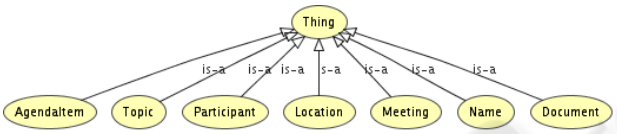

Fig. 2. Meeting Ontology Class Hierarchy.

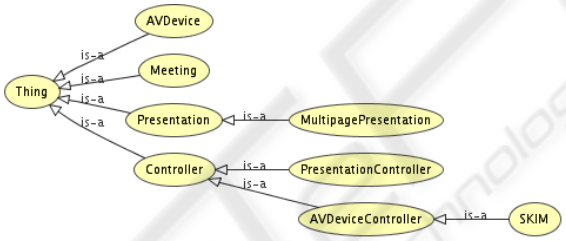

Fig. 3. Devices and Presentation Ontology Class Hierarchy.

We also define a set of properties between these classes, for example in the `Nokia` meeting ontology we have the following properties

`location`:	`Meeting` → `Location`	functional property
`meetingHost`:	`Meeting` → `Participant`	subrole of `participants`, cardinality 1
`participants`:	`Meeting` → `Participant`	cardinality: 0..∞
`topic`:	`Meeting` → `Topic`	cardinality: 1..∞
`belongsTo`:	`Presentation` → `Meeting`	functional property
`agenda`:	`Meeting` → `AgendaItem`	cardinality: 1..∞
`responsibleOf`:	`AgendaItem` → `Participant`	functional property
`presentations`:	`Meeting` → `Presentation`	cardinality: 0..∞
`participatesIn`:	`Controller` → `Meeting`	functional property
`controllers`:	`Meeting` → `Controller`	cardinality: 0..∞
`shows`:	`AVDeviceController` → `Presentation`	cardinality: 0..∞
`controls`:	`PresentationController` → `Presentation`	cardinality: 0..∞
`document`:	`Presentation` → `Document`	functional property

Data properties include the start and end times of the meetings, agenda item descriptions and the name of the participants: these are all of type 'date/time' and 'string' accordingly. Similar properties for the presentation and device ontology can be see in the example RDF graph in figure 4.

[5] http://www.linkedin.com/ http://www.facebook.com

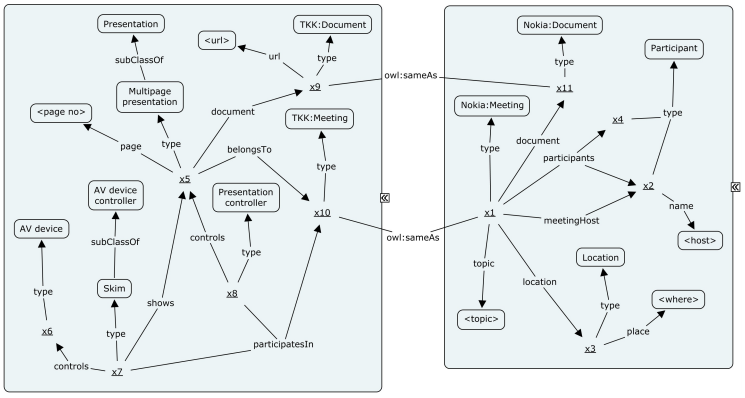

Fig. 4. RDF graph of a meeting.

The expressiveness required of the description logic for these ontologies is relatively simple, in the meeting ontology case it is $\mathcal{ALCHQ}_{(\mathcal{D})}$ and in the device/presentation case it is $\mathcal{ALCF}_{(\mathcal{D})}$. This means that any reasoning that is requires to be made over these ontologies can be reasonably simple and does not require the power of OWL[6] reasoner. Furthermore, most current business information models have been specified using entity-relationship models which are easily mapped to simple description logics of much less complexity than OWL [15].

Note that the construction of the ontologies are very much a compromise between the parties creating those ontologies. There are properties and even classes which do not necessarily accurately reflect 'real-life'. Ontologies will change over time and the aforementioned semi-structured/ontology agnostic nature of RDF assists in representing this.

Some classes represent rather technical matters. Class `AVDevice` represents a thing that is able to present audiovisual content, e.g., a video projector, a display of a laptop, a loudspeaker, a window on a computer desktop, etc. Class `AVDeviceController` represents a software component that can render output to an `AVDevice`. Class `SKIM` is an example subclass of `AVDeviceController`. It represents a software component that uses the SKIM viewer [16] to show a PDF document on a display. Class `Controller` is just an abstraction of the controller components. The class `MultiPagePresentation` is an important subclass of `Presentation` and it represents a presentation consisting of several pages, e.g., a presentation of a PDF document.

In Figure 4 we present an RDF graph of an instance of these ontologies. The underlined x's are URI's that represent individual objects: participants, presentations, controllers, etc. For each URI we present its type, i.e., the class the instance of which the URI is. In addition we present several relations between objects, e.g., the relation `shows` between and the instance x_7 of class `SKIM` and instance x_5 of class `MultiPage-Presentation`, which tells that a presentation is shown by a software component on a specific audiovisual device.

The concepts of a 'meeting' and a 'document' occurs in both ontologies which suggests some kind of semantic similarity. As we shall see we can take advantage of these

[6] http://www.w3.org/TR/owl-features/

kinds of semantic similarity to enable interoperability between the various ontologies [17, 18].

The information models are linked together by the classes `Meeting` and `Document`. We can see the separation of the ontologies: the left-hand side presents the devices-and-presentations aspect and the right-hand side the participants-topic-location-and-time aspect of a meeting.

3.2 Agents and Interaction

To effect a meeting we require actions over the defined ontologies. These actions are implemented in our system through simple agents [9] with varying degrees of responsibility (from simple atomic actions to implementing complete functionality) which can be distributed across many devices.

Consider the case where a meeting is being set up, this involves the creation of an instance of class `Meeting` and to ensure consistency, one participant which initially must be the meeting host, the topic and at least one agenda item. We defined the data properties and other object properties to be optional, though one should also specify the location, start and end times etc. Whether these properties should be optional (and others not) is a matter for the ontology designer and what particular aspects they are trying to capture.
To implement this, we need a software agent, say `MeetingBuilderAgent`, that is able to create the meeting in the Sedvice RDF store, add agenda items and participants of the meeting to the RDF store etc. This software agent communicates with Sedvice and has a user interface, e.g., a set of web pages, some widget-style application etc.

As the information store contains only the current context there is no necessity for presentations, participants etc to be bound to the meeting when it is created. In fact as we are only manipulating context we have a divorce between the temporal aspect of when entities have been created and linked to each other. This allows us to be free from any given specific structure and process associated with the setting up of a meeting.

As a meeting progresses the interactions between the agents with the body of shared information or the context of the meeting more accurately reflects how this particular interaction between users progresses. Information ranging from simple RDF triples to larger bodies of information can be linked to the entities defined in our ontologies as necessary. Additionally as we are 'free' from specific, rigid structures we can in effect link *any* information. For example, one could link some IRC conversation with an agenda point and use that information later when reasoning and processing about the overall context of a meeting such as effected by automatic meeting minutes generation and/or summarisation.

The agents controlling the presentation of the material work thus: we need at least two software agents, say `PresentationControllerAgent` and `SKIMAgent`. The former is used by the participants of the meeting for controlling presentations: starting presentations, moving forward and backward in the documents that are presented, etc. It communicates with the Sedvice RDF store adding there RDF triples representing the presentations, their corresponding documents, the current pages in a presentation etc. It also has a user interface, e.g. a set of web pages to be used by a meeting participant for controlling the presentation. Several `PresentationControllerAgent`

instances may exist simultaneously, if several participants control presentations at the same time.

The `SKIMAgent` component listens to modifications in the RDF store. Whenever a new presentation is to be shown by the SKIM agent, it awakes the local SKIM application and tells it to show the corresponding document[7]. Similarly, if a change of the current page appears in the RDF store, the SKIM agent tells the local SKIM application to change the page. If several displays are available, we should have several SKIM agents running.

3.3 Reasoning and Information Integration

Given a corpii of information such as that in figure 4 we can construct agents that can analyse and reason about this information. In particular we require agents that can:

1. relate two elements together
2. relate two (or more) information structures together
3. search for and generate additional information based on current context

The first case is relatively trivial and can be realised using agents that look for instances of certain types and link these together. The simplest version of this is an agent that simply notices an instance of one type and asserts that it also takes symmetrically a second type:

$$\frac{\Gamma, \quad x \texttt{ type } \mathit{Nokia} :: \mathit{Meeting}}{\Gamma, \quad x \texttt{ type } \mathit{TKK} :: \mathit{Meeting}} \text{ type\_inference}_1 \tag{1}$$

$$\frac{\Gamma, \quad x \texttt{ type } \mathit{TKK} :: \mathit{Meeting}}{\Gamma, \quad x \texttt{ type } \mathit{Nokia} :: \mathit{Meeting}} \text{ type\_inference}_2 \tag{2}$$

A more sophisticated variation on this is to utilise the `owl:sameAs` predicate [19] but this relies upon reasoning over a larger set of information instead of indiscriminate linking of instances (nodes in the RDF graph) together. For example, one could implement an agent with the following rule for some properties p_1 and p_2:

$$\frac{\Gamma, \quad x \texttt{ type } \mathit{Nokia} :: \mathit{Meeting} \wedge y \texttt{ type } \mathit{TKK} :: \mathit{Meeting} \wedge x.p_1 = y.p_2}{\Gamma, \quad x \texttt{ owl:sameAs } y} \tag{3}$$

The above are of course monotonic rules though in our experience rule sets within agents and typical information corpii tend to be non-monotonic in nature. This then leave various problems regarding rule conflict and aspects of negation and missing information - open world vs closed world assumptions. Investigation is progressing in these ares but at this time is not mature enough to meaningfully discuss in this context.

This then leads to analysing larger information structures and deciding, potentially at run-time what constitutes the required structure: [20] defines the notion of *RDF-molecule* which denotes a larger structure based upon the necessary set of properties

[7] We chose SKIM as the PDF viewer, since it can easily be commanded outside using AppleScript

aggregated together to make a more meaningful element or object [21].

How larger structures are mapped together or made equivalent beyond the above syntactic methods is open for development [22]. Techniques based upon semantic similarity in the linguistic sense probably provide the most reasonable method for performing this matching [23, 24].

In simple examples such as the case study in this paper the three explicitly stated rules above implemented over a number of agents suffices to link the information structured by the given ontologies together. In many cases simple rule set suffice although as we have noted there is the need for non-monotonic reasoning in some cases.

4 Content-driven Meetings Scenario

In this section we show how meetings work using our framework. In content-centered, informal meetings the activity is not driven by an agenda or predefined process; instead, the activity emerges in a dynamic and opportunistic manner as reaction to the development of the information content.

The crucial aspect in our approach is the flexible and easy interleaving of meeting conversation with visual presentations. The conversation itself is flexible and can lead into vivid and dynamic interaction between participants. However, the difficulties to make visual presentations often lead to static arrangements, over-prepared presentation materials, and as a result, in non-interactive presentation events. Such meetings give little room for successful building of shared knowledge.

Below we go through meeting scenarios to show how flexible presentation interoperability tools have a potential to provide more freedom to ordinary meetings. It should be stressed that the presentation framework is based on open and free participation: each participants must decide whether to follow the presentation by subscribing to it. In addition, these scenarios should not be confused with ”smart meeting room” scenarios; we are focusing on ubiquitous, easy, and flexible meetings.

Scenario (Setting Up and Working in the Meeting). Three researchers, Alice, Bob and Eve, have to prepare a conference paper. Alice has written a draft for sections 1 and 2, Bob and Eve drafts for sections 3 and 4. They should meet to discuss the paper.

Using `MeetingBuilderAgent`, Alice creates in a ‘smart space’ (or just space) a meeting with agenda items *Go through Alice’s draft*, *Go through Eve’s and Bob’s draft*, and *Producing the final version*. Later, Alice creates a presentation for document `s12.pdf` and binds it to the meeting in the space. Here one might see a structure similar to the right-hand-side of the diagram in figure 4.

Alice, Bob and Eve agree on a meeting at university cafeteria. Like Alice before, Bob creates a presentation for document `s34.pdf`. This can be made in the same space or in a different space with the two spaces being merged or one projected into the other - for simplicity we assume the same space.

Alice, Bob and Eve meet at university cafeteria. They have their laptops with them. To show his presentation, Alice launches a `PresentationControllerAgent`, which creates a the meeting and presentation to be controller (but of different types to the earlier meeting and presentation) in the space and starts to control the presentation.

One of the reasoner agents recognises this and infers that these can be unified in the semantic sense - for example using the rules 1,2 and 3 described earlier and possibly enhanced with the more sophisticated semantic pattern matching also described. Now the various agents that work across the different ontologies do not see any difference in the separate information structures.

To see the presentation at their laptops, Alice, Eve and Bob launch `AVDeviceControllerAgent`s. Using `PresentationControllerAgent`, Alice starts the presentation on the first page. The `AVDeviceControllerAgent`s detect the change of page and show the corresponding page on a window in each laptop. Using `PresentationControllerAgent`, Alice moves forward in the presentation. The `AVDeviceControllerAgent`s detect the page changes and show the corresponding pages.

At some point, Eve notes that there is an inconsistency between two document drafts. He launches another `PresentationControllerAgent`, makes it join the meeting and control the presentation of Bob's and Eve's draft. The participants set their `AVDeviceControllerAgent`s to show this presentation too. Using his `PresentationControllerAgent` Eve moves to page three of Bob's and Eve's draft. `AVDeviceControllerAgent`s show the corresponding page on the laptops. Alice adds into the space an annotation about the inconsistency between the drafts.

At this point, the participants decide that it is better to simultaneously go through the both presentations to check the inconsistencies. Alice and Eve control the presentations, and the participants add annotations about detected inconsistencies in the presentations. After working like this for a while, the participants decide that they should write new versions of their drafts. They decide to meet again later to finalize the paper using the new versions of the drafts.

Note how the situation developed. In the original agenda, the idea was to go through the drafts one at a time and then decide how to produce the final version. However, when an inconsistency between the drafts was detected, the meeting took a different path. In traditional, rigid meeting situations, this may be considered as a fault in the process. However, when we have a content-driven approach, this is very natural and causes no problems.

The required software agents can be rather simple. `MeetingBuilderAgent` and `PresentationControllerAgent` can be implemented as web pages that access the RDF store. `AVDeviceControllerAgent` can be implemented as a couple of scripts that monitor changes in the space and command a PDF viewer. The SKIM controller mentioned above was implemented this way.

Scenario Continued (Note Taking). Eve makes copious notes during the meeting using an unrelated agent. As Eve makes these notes they are linked both to the meeting context and to the current presentation. Furthermore as the note contain valuable information regarding what is currently being discussed Eve starts a further search agent that can analyse the text and make searching for other interesting presentation via some suitable search technology (Citeseer, Google etc).

Eve notices that some of these suggested presentations are highly relevant and se-

lects them from her web-browser which then writes the details of the selected presentations into the space. Eve has also set up her web browser agent to link these presentations directly into the current meeting so that they can become both part of the current meeting context and because of the reasoning agents described earlier (cf: eqns.1,2,3) are also available for show during the meeting.

This kind of interoperability and specifically the ability to link in information into the meeting in a very ad hoc manner allows not only for the overall meeting context to be expanded but also that items such as searches for material made 'outside' the meeting scope not to be lost.

There are interesting privacy issues here such as what happens if much of the above is fully automatic and one of the meeting participants is not fully concentrating on the meeting and starts searching for 'other' material - an all too common occurrence in many meetings. However we espouse the use of much more direct interaction with the user thus empowering the user with more control over how his or her information is being used and the context gathered.

Scenario Continued (Minutes). Later at a meeting room in TKK, Bob and Eve have another meeting to edit their draft. Eve creates a new meeting and adds the presentations of the previous meeting to this meeting. He also adds a new presentation for the new draft that they are going to edit. In addition to their laptops, they have a computer that is connected to a video projector. To see what happened in the last meeting, Bob creates another presentation "last meeting minutes". This is a special kind of presentation that can be controlled by `MeetingMintesPresentationAgent`, a software agent that is capable of going through the events of the previous meeting. Such an agent might even be working continuously such that the minutes are being continuously generated.

For showing the minutes, Bob also launches an `AVDeviceControllerAgent` that shows a time-line of the last meeting on the video projector. Using these agents, Bob and Eve go through the annotations of in the previous meeting. For each annotation, the meeting minutes agents show the pages of the presentations that were shown in the previous meeting when the annotation was made.

Here we see how the context of the previous meeting is seamlessly merged to the context of the new meeting. There was no need to separately write down the minutes of the meeting; they were created during the meeting. The information that the agents record in the RDF store is a much richer and more accurate presentation of what really happened. What is needed in addition to the classes that we described in Section 3 is a way to time stamp events. This can easily be added to the agents.

The software for `MeetingMintesPresentationAgent` and the corresponding `AVDeviceControllerAgent` can be simple. The former can be implemented as web pages that access the space to detect the course of action in the previous meeting and provides a way to go through this information. In addition, the agent acts as a con-

troller for the presentations that were shown in the previous meeting to show how the presentations proceeded in the previous meeting.

5 Discussion, Conclusions and Future Work

We have described the ontologies, the typical reasoning over ontologies and the actions might be performed by participants of a meeting. Our approach does not explicitly imply a strict process nor ordering of actions but rather enables the actions (implemented as agents) to be made under the control of the users at the appropriate times according to the context of the whole meeting.

The following points are specifically raised and are supported by the distributed, information sharing nature of our system:

- Individual actions implemented as agents
- User use the useful set of agents at any given point in time
- The context of the meeting is independent from the actual meeting itself
- The physical and temporal location of the meeting is independent from the content and thus context of the meeting

This benefits the user in more realistic and complex situations where people make annotations, make multiple presentations, link to external content, are physically mobile (temporally and physically) and so on.

One of the particular features that supports a much freer environment for the users is that there is no flow of control explicitly enforced between agents - the environment is declarative in nature. This of course has certain problems such as agents getting access to and potentially modifying data before a particular information structure is ready. These are however handled by suitable guards in the agent design and even by external signalling mechanisms as described earlier.

Issues relating to security, privacy and trust are not addressed by our framework at this time. However the nature of a space is such that the architecture support policy based agent joining to a space: that is for an agent to have access to a space then it must supply the correct set of credentials before being admitted.

Acknowledgements

This work has been partially funded as part of the TEKES ICT SHOK DIEM (www.diem.fi) project.

References

1. Conklin, J.: Dialogue Mapping: Building Shared Understanding of Wicked Problems. Wiley (2005)
2. Marjanovic, O., Seethamraju, R.: Understanding knowledge-intensive, practice-oriented business processes. In: Proceedings of the 41st Hawaii International Conference on System Sciences. (2008)

3. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American 284 (2001) 34–43
4. Khushraj, D., Lassila, O., Finin, T.: stuples: semantic tuple spaces. Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on (22-26 Aug. 2004) 268–277
5. Athanasopoulos, D., Zarras, A., Issarny, V., Pitoura, E., Vassiliadis, P.: Cowsami: Interface-aware context gathering in ambient intelligence environments. Pervasive and Mobile Computing 4 (2008) 360–389
6. Oliver, I., Honkola, J.: Personal semantic web through a space based computing environment. In: Middleware for Semantic Web 08 at ICSC'08, Santa Clara, CA, USA. (2008)
7. Oliver, I., Honkola, J., Ziegler, J.: Dynamic, localised space based semantic webs. In: WWW/Internet Conference, Freiburg, Germany. (2008)
8. Nixon, L.J.B., Simperl, E., Krummenacher, R., Martin-Recuerda, F.: Tuplespace-based computing for the semantic web: a survey of the state-of-the-art. The Knowledge Engineering Review 23 (2008) 181–212
9. Haag, S., Cummings, M., McCubbrey, D.J.: Management Information Systems for the Information Age. McGraw-Hill Companies (2003) 4th edition. 978-0072819472.
10. Lassila, O.: Taking the rdf model theory out for a spin. In: ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web, London, UK, Springer-Verlag (2002) 307–317
11. Rodriguez, M.A., Bollen, J.: Modeling computations in a semantic network. CoRR abs/0706.0022 (2007)
12. Parnas, D.L.: On the criteria to be used in decomposing systems into modules. Communications of the ACM 15 (1972) 1053–1058
13. Turnitsa, C.D.: Extending the levels of conceptual interoperability model. In: Proceedings IEEE Summer Computer Simulation Conference, IEEE CS Press. (2005)
14. Hu, B., Dasmahapatra, S., Lewis, P.H., Shadbolt, N.: On capturing semantics in ontology mapping. In: AAAI. (2007) 311–316
15. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation and Applications. 2nd edition edn. Cambridge University Press (2007) 978-0-521-87625-4.
16. : (Skim, pdf reader and note-taker for os x.) http://skim-app.sourceforge.net/.
17. Budanitsky, A., Hirst, G.: Evaluating wordnet-based measures of lexical semantic relatedness. Computational Linguistics 32 (2006) 13–47
18. Lindsey, R., Veksler, V.D., Grintsvayg, A., Gray, W.D.: Be wary of what your computer reads: The effects of corpus selection on measuring semantic relatedness. In: Proceedings of the 8th International Conference on Cognitive Modeling, Ann Arbor, MI (2007)
19. Passant, A.: :me owl:sameas flickr:33669349@n00. In: Linked Data on the Web (LDOW 2008), Beijing, China. (2008)
20. Ding, L., Finin, T., Peng, Y., Silva, P.P.D., L, D.: Tracking rdf graph provenance using rdf molecules. Technical report, 2005, Proceedings of the Fourth International Semantic Web Conference (2005)
21. Smith, B.C.: On the Origin of Objects. The MIT Press (1996) 0-262-69209-0.
22. Shafiq, O., Scharffe, F., Wutke, D., del Valle, G.T.: Resolving data heterogeneity issues in open distributed communication middleware. In: Proceedings of 3rd International Conference on Internet and Web Applications and Services (ICIW 2008), Athens, Greece. (2008)
23. Ruotsalo, T., Hyvönen, E.: A method for determining ontology-based semantic relevance. In: Proceedings of the International Conference on Database and Expert Systems Applications DEXA 2007, Regensburg, Germany, Springer (2007)
24. Dumais, S.: Data-driven approaches to information access. Cognitive Science 27 (2003) 491–524