Serving both Worlds
Infolayer Status Report

—

LS8-Report 26

Stefan Haustein
University of Dortmund, Computer Science VIII,
D-44221 Dortmund, Germany
haustein@ls8.cs.uni-dortmund.de

February 12, 2001

Abstract

The CoMRIS Information Layer (Infolayer) is an information system that is able to serve structured information in formats suitable for both, human users and software agents. The system used an UML based ontology as structure definition and is able to build an generic HTML based user interface for complex structures without additional effort. It is built to support existing communication and query standards like HTTP, XHTML, XML, FIPA and OQL.

1 Introduction

The COMRIS Information Layer (Infolayer) is an ontology-based information system that is designed for two main tasks:

1. It is able to provide a human readable representation of general structured content. Any common web browser like Netscape Navigator or Microsoft Internet Explorer can be used to browse and edit the content of the information layer without needing any proprietary extensions or plug-ins. The HTML generation is driven by the ontology and allows to build a generic HTML interface for complex structures without additional effort. The HTML code generated can be custumized using XML based template techniques.

2. It provides easy-to-use information services for software agents, enabling developers to concentrate on the main purpose of their agents. This means especially that agents do not need to care much about data storage and persistence and are able to access stored information at a high level of abstraction. The Infolayer can also be used as a kind of persisten blackboard for indirect communication and reducing communication complexity [5, 6].

   The agent interface is not limited to access by software agents, it is useful for general access to the information in a machine readable format.

This report shows the motivation behind building the Infolayer and explains its main building blocks including their current state of development and future plans.

2 Motivation

The Motivation for the Infolayer was the *Co-Inhabited Mixed Reality Information Spaces* (COMRIS) project. The goal of COMRIS was to implement a conference support system as an example application for a mixed reality scenario, where software agents help their users by performing a "virtual conference" in parallel to the "physical conference" [8].

For the conference scenario, a lot of information needed to be modelled in a structured way in order to be accessible by both, software agents and human users. The amount of concepts to be modelled like participants, speakers, talks, sessions, rooms, agents, boths, schedules etc., became quite large. Moreover, all concepts had a lot of complex relations to other concepts.

Using relational tables for this purpose seemed unappropiate because of the complicated mapping that is required to transform the ontology to a high number of tables. Also, the table solution seemed inflexible because ontological changes would cause a lot of changes in database tables and additional "agentification" and HTML wrappers.

Description Logic systems provide additional features like automated classification that are computational expensive but not required in the system. All reasoning was intended to be performed at the specialized agents. Like for the relational tables, additional wrappers to HTML and an Agent Communication Language (ACL) would be required.

So we decided to build a new kind of information system that

- provides ACL and HTML access in the first place,
- is agent based itself,
- and is built on an ontology that is not hard-wired to the system.

3 System Architecture

The Infolayer is an agent based system itself, where the kernel provides only a memory cache representation for structured information.

In order to do something more "senseful", additional modules or agents are provided, where the agents perform specialized tasks like:

- Building a generic HTML presentation from the ontology and the actual Infolayer content
- Handling communication with other agents
- Synchronization with the underlying persistent data storage.

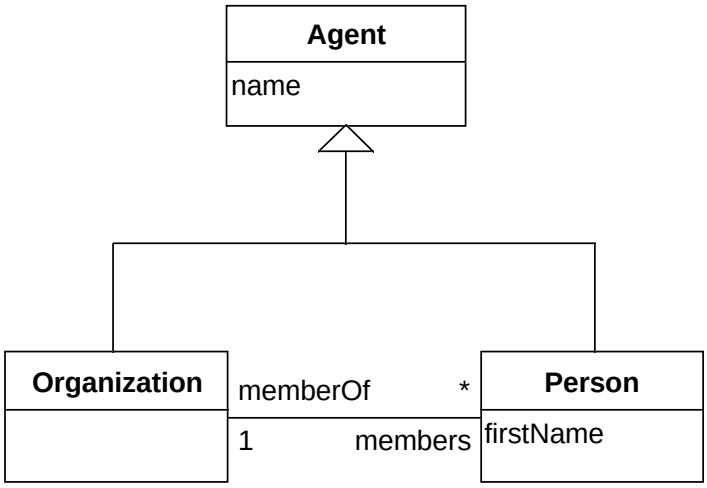

Figure 1: Sample UML diagram

```
<Concept name="Agent" show="name">
  <attr name="name" />
</Concept>

<Concept name="Person" show="name + ', ' + givenName">
  <attr name="givenName"/>
</Concept>

<Concept name="Organization" extends="Agent">
  <relation name="members" type="Person"
       inverse="memberof" imax="1"/>
</Concept>
```

Figure 2: Ontology encoding example

```
<Organization id="555777">
   <name>IBM</name>
</Organization>

<Person id="888543">
   <name>Katschenko</name>
   <firstName>Tanja</firstName>
   <memberOf idref="555777"/>
</Person>

<Person id="878653">
   <name>Gomez</name>
   <firstName>Carlos</firstName>
   <memberOf idref="555777"/>
</Person>
```

Figure 3: Content language example

4 Ontology and Data Model

The information layer uses an object-orientated model for data representation. Objects consist of atomic attributes and relations to other objects. The consistency of relations in both directions is ensured automatically, avoiding inconsistencies inside the system. The concepts and relations are defined application-dependent in an external ontology definition file. All files used by the information layer are stored as XML documents.

The ontology used in the COMRIS Information Layer is defined using an UML model [4] encoded in a simple XML file. Figure 1 shows an examplSe of an UML model and figure 2 the corresponding XML definition. Concepts (Classes) are defined by the element <`Concept`>. Inside the <`Concepts`> element, attributes are defined by <`attr`>, and relationd are defined by <`relation`>.

Inside the definition of `Organization.members`, also the inverse relation `Person.memberof` is created. Both relations are synchronised automatically. There is no difference whether a relation is created directly or as an inverse relation at the corresponding target class. The only condition for defining a relation is that both classes are already declared in the structure definition file. The purpose of the `show` attribute is to generate a human readable name for all instances.

5 Communication and Content Languages

The content language for software agents and system components is based on XML, too. The actual content language format is derived from the ontology automatically. Figure 5 shows the content language encoding of Tanja Katschenko and Carlos Gomez working at IBM corresponding to the ontology example in the previous section.

Instead of using the `idref` attribute to describe relations, instances can also be embedded in the relation element. Figure 4 shows a corresponding nested structure.

```
<Organization id="555777">
  <name>IBM</name>
  <members>
    <Person id="888543">
      <name>Katschenko</name>
      <firstName>Tanja</firstName>
    </Person>
    <Person id="878653">
      <name>Gomez</name>
      <firstName>Carlos</firstName>
    </Person>
  </members>
</Organization>
```

Figure 4: Encoding example for nested structures

All modules of the information layer use the flat encoding model for internal information exchange.

The encoding used for sending instances to software agents or other entities can be controlled by the corresponding entity to fit its particular needs best.

6 Query Interface

The Infolayer supports a subset of OQL [3] as query language for agents. Additional languages may be plugged in by adding corresponding agents. By subscribing to the Infolayer, it is possible to keep an agent up to date without polling [7].

7 HTML Generation

The InfoLayer contains a module that provides built-in web-server functionality. Since XML is not yet well supported by web browsers, the server is able to generate HTML dynamically: For any object, the attributes are simply displayed, and the relations are converted to sets of hyperlinks to the related objects (figure 5).

In additon to generic HTML generation, also HTML templates can be used. These templates look like standard HTML files except that they may contain special elements ("tags") for server side includes (SSI). When the page is parsed, the SSI elements are interpreted with respect to the current contents of the information system and replaced. The modified page – now fulfilling the XHTML standard without extensions – is then sent to the web browser.

In the COMRIS project, we have been using the template mechanism to convert the data structure of the information layer to an input structure suitable for the an template based text generation system (TG/2, [2]) in order to generate natural language output for the wearable device.

Figure 5: Access to the Infolayer using a Web Browser.

8 Conclusion and Outlook

The implemented system provides a new possibility of publishing structured and massively linked data to the web. Additionally it is able to generate XML data that is also readable by software agents. The system may not only be useful for modeling some aspects of a conference but also for other applications with many sets of small and massively linked objects.

Currently, the comris information layer is used for two internal projects and as the teaching server of MLnet[1].

The most important future development is to make the Infolayer compliant to SOAP serialization [1] in order to use a standardized XML format for the message content language as well as ontology definition.

References

[1] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., and Winer, D. Simple Object Access Protocol (soap) 1.1. Note, World Wide Web Consortium, 2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508.

[2] Busemann, S. A shallow formalism for defining personalized text. In *Workshop Professionelle Erstellung von Papier- und Online-Dokumenten at the 22nd Annual German Conference on Artificial Intelligence (KI-98)* (Bremen, September 1998).

[1] http://www.mlnet.org, please follow the teaching link in the menu bar to the left

[3] Cattell, R. G. G., Ed. *The Object Database Standard: ODMG 2.0.* Morgan Kaufmann, 1997.

[4] Cranefield, S., and Purvis, M. Uml as an ontology modelling language. In *Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99* (1999).

[5] Haustein, S. Information environments for software agents. In *KI-99: Advances in Artificial Intelligence* (Bonn, Germany, September 1999), W. Burgard, T. Christaller, and A. B. Cremers, Eds., vol. 1701 of *LNAI*, Springer Verlag, pp. 295 – 298.

[6] Haustein, S., and Lüdecke, S. Combination of Agent- and Blackboard-Technologies for Buisiness Applications. In *Workshop "Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien"* (Ilmenau, 1999), S. Kirn and M. Petsch, Eds., no. 16 in Arbeitsbericht, TU-Ilmenau, Wirtschaftsinformatik 2, pp. 231 – 237.

[7] Haustein, S., and Lüdecke, S. Towards Information Agent Interoperability. In *Cooperative Information Agents IV – The Future of Information Agents in Cyberspace* (Boston, USA, July 2000), M. Klusch and L. Kerschberg, Eds., vol. 1860 of *LNCS*, Springer, pp. 208 – 219.

[8] Plaza, E., Arcos, J. L., Noriega, P., and Sierra, C. Competing agents in agent-mediated institutions. *Personal Technologies Journal 2*, 3 (1998), 1–9.