ConTeXtualized local ontology specification via CTXML*

Paolo Bouquet[1] **Antonia Donà**[2] **Luciano Serafini**[2] and **Stefano Zanobini**[1]

[1]Dept. of Computer Information and Communication Technologies, University of Trento, Italy
[2]ITC-Irst, Via Sommarive – Località Povo, 38050 Trento, Italy
bouquet@dit.unitn.it, antodona@itc.it,
serafini@itc.it, stefano.zanobini@libero.it

Abstract

In many application areas, such as the semantic web, knowledge management, distributed databases, it's been recognized that we need an explicit way to represent meanings. A major issue in all these efforts is the problem of semantic interoperability, namely the problem of communication between agents using languages with different semantic. Following (Bonifacio, Bouquet, & Traverso 2002), we claim that a technological infrastructure for semantic interoperability between "semantically autonomous" communities must be based on the capability of representing local ontologies and mappings between them, rather than on the attempt of creating a global, supposedly shared, conceptualization. The goal of this paper is to define a theoretical framework and a concrete language for the specification of local ontologies and mappings between them.

Introduction

In many application areas, such as the semantic web, knowledge management, distributed databases, it's been recognized that we need an explicit way to represent meanings. This is needed to enhance (or enable) services like search, knowledge and service integration, discovery. This produced efforts like ontology development, semantic-based markup languages, agent communication languages.

A major issue in all these efforts is the problem of semantic interoperability. Even in restricted domains, like medicine, tourism, banking, automotive, it is very difficult to find an agreement on common vocabularies and shared conceptualizations. This leads to a situation in which different agents use the same word to mean different things, use different words to mean the same thing, use different granularity to describe the same domain, describe a domain from a different perspective, and so on. All together, this is what researchers call *semantic heterogeneity*, namely a situation in which agents do not understand each others as they use languages with heterogeneous semantic.

*This work has been partially supported by the project EDAMOK Enabling Distributed and Autonomous Management of Knowledge, funded by the Provincia Autonoma di Trento with deliberation number 1060 on date 4/5/2001.

A common approach to overcome semantic heterogeneity is to introduce shared representations of meanings. Indeed, it is obvious that designing any semantic-based application would be much easier if people agreed to organize and exchange information in a common language with a common semantic. However, in general this is not the case. Despite the effort for defining a standard semantic for various domains, people seem to resist to such an attempt of homogenization. Partly, this is due to practical problems (it can be very costly to change the overall organization of a database, or the classification of large collections of documents. But we believe that there are also theoretical reasons why this homogeneity is not accepted, and in the end is not even desirable. In fact, lots of cognitive and organizational studies show that there is a close relationship between knowledge and identity. Knowledge is not simply a matter of accumulating "true sentences" about the world, but is also a matter of interpretation schemas (e.g. paradigms (Kuhn 1979), contexts (Benerecetti, Bouquet, & Ghidini 2000), mental models (Johnson-Laird 1983), perspectives (Boland & R.V.Tenkasi 1995), ...), which allow people to make sense of what they know. Therefore, any attempt of imposing external interpretation schemas (and a definition of meaning always presupposes some interpretation schema, at least implicitly) is perceived as an attack to an individual's or a community's identity. Moreover, interpretation schemas are an essential part of what people know, as each of them provides an alternative lens through which reality can be read. Thus, imposing a single schema is always a loss of global knowledge, as we throw away possibly innovative perspectives.

If we accept that interpretation schemas are important, then we need to approach the problem of semantic interoperability from a different perspective. Instead of pushing towards a greater uniformity, we need a theoretical framework in which:

- different conceptualizations (called "local ontologies") can be autonomously represented and managed (and, therefore, we call them contextualized);
- people can discover and represent relationships between local ontologies;
- the relationships between local ontologies can be used to provide semantic-based services without destroying the

"semantic identity" of the involved parties.

We see *meaning negotiation* as the process that dynamically enable agents to discover relationships between local ontologies. The goal of this paper is to create an "environment" in which the preconditions for meaning negotiation are satisfied. In particular, on the one hand, we define a theoretical framework in which local ontologies and mappings between them can be represented; on the other hand, we provide a language for describing what we call a *context space*, namely a collection of contexts and their mappings; this language is called *C*onTeXt Markup Language (CTXML) and is based on XML and XML-Schema. Local ontologies are represented as *contexts*, in the sense discussed in formal papers like (Ghidini & Giunchiglia 2001; Benerecetti, Bouquet, & Ghidini 2000). In what follows, we will use knowledge management as our main motivation for contextualized local ontologies; however, as we said at the beginning, we believe that similar motivations can be found in any semantically distributed application, e.g. the semantic web.

Contextualized local ontologies in knowledge management

Knowledge has been recognized as one of the most important assets of modern organizations. (Drucker 1994) claims that we are entering "the knowledge era", in which the basic economic resource is no longer capital, or natural resources, or labour, but knowledge.

As a managerial practice, Knowledge Management (KM) can be described as a collection of methodologies and tools that provide support in creating new knowledge within the organization and codifying such newly created knowledge into "storable objects" (e.g. documents, repositories, databases, procedures, forms).

(Bonifacio, Bouquet, & Traverso 2002) shows that in KM we can find the same dichotomy between centralized and distributed approaches to knowledge, and provides motivations to support a distributed approach to KM, namely an approach that starts from the recognition that there exist autonomous communities within an organization and that technology should supports knowledge exchange not by eliminating differences, but by designing systems that will enable *semantic interoperability* between autonomous communities.

We assume that autonomous communities organize their (local) knowledge according to a *local ontology*, i.e. a set of terms and relations between terms, which provides a conceptualization of the piece of world which is relevant for the objectives of that community. Examples of local ontologies are: a set of shared directories, a taxonomy used by a scientific community, and so on. Each community (team, group, and so on) within an organization has its own conceptualization of the world, which is partial (i.e., covers only a portion of the world), approximate (i.e., has a degree of granularity), and perspectival (i.e., reflects the community's viewpoint on the world). Each local conceptualization represents a community's perspective on the world (see for example the concept of perspective making in (Boland & R.V.Tenkasi 1995)), and contributes to define a community's identity. In general, different perspectives are not completely unrelated. For example, two communities may have different viewpoints on the same domain, or have overlapping interests. Possible existing relations between different perspectives can be seen as mappings between the relative different and autonomous conceptualizations. These mappings cannot be defined beforehand, as they presuppose a complete understanding of the two conceptualizations, which in general is not the case. This means that mappings can be discovered dynamically. e.g. through communication, experience, trial and error. In other words, through meaning negotiation. We are now going to describe the theoretical framework presented in the introduction with respect to such a KM scenario.

Contexts

The **abstract representation** of a context is a triple $\langle c, \mathcal{A}, \mathcal{K} \rangle$, where: c is a context identifier; $\mathcal{A}$ is a collection of explicit assumptions; and $\mathcal{K}$ is an explicit representation.

The *context identifier c* is a unique identifier associated with a context; *explicit assumptions* are attributes (parameter/value pairs) that provide meta-information about the context (e.g., the context owner, or its history); the *explicit representation* is the real content of a context, namely a conceptualization, and is represented as a labelled tree. Possible reference models for the content of contexts can be based on first order logics, propositional logics, description logics, general graph structures (graphs, acyclic graphs, lattices, etc.), concept hierarchies, and so on. In this paper, we chose to use concept hierarchies (in the sense defined in (Büchner *et al.* 1999)).

Concept hierarchies are built from a set L of labels. L is composed by two disjoint subsets:

1. L_C (concept labels);
2. L_R (relation labels).

L_R is split into hierarchical (L_H) and non-hierarchical – or general – labels (L_G).

Definition 1 (Concept hierarchy). A *concept hierarchy* is a graph $H = \langle C, E \rangle$, where C is a finite set of nodes, E a finite set of directed edges between nodes, and all the nodes and edges have a label from L, such that the edges labelled with hierarchical labels form a tree.

We don't put any restriction on the set L_C of concept labels, whereas we require that $L_R = \{$*is-a*, *part-of*, *inst-of*$\}$, where: *is-a* represents the subclass relation (for instance "Cat *is-a* Animal", "Man *is-a* Mortal"); *part-of* represents the relation of being part of (for instance, "Leg *part-of* Man", "Tenor *part-of* Choir"); *inst-of* represents the fact that a certain individual is an instance of a concept (for instance "Paolo *inst-of* Man", "Michele *inst-of* Tenor").

Contexts can be given a concrete representation using different languages: XML, XML-schema, KIF, CycL, Ontolingua, DAML-OIL, RDF, RDF-schema (see (Corcho & Pérez) for a survey on these languages). Among them, we decided to adopt XML and XML-Schema.

The *concrete representation* of a context is an XML document composed of two main parts: the *header* and the *content*. The header is an XML document containing the following components:

Owner The owner of the context. This is important in KM applications, as it can be used, for example, to grant permissions, and to allow users to assign a degree of confidence to a context based on their trust in the context owner.

Group The group in which this context has been developed. This is necessary as the owner can be a member of different groups.

Security Contains information about the access rights, passwords, encryption, etc.

History Contains information on *how* a context was generated. Examples are: "from scratch", "by combining (portions of) existing contexts", "by copying some existing context".

The content of a context is a concrete representation of a concept hierarchy in an XML-schema. Since XML-schema is an XML document, an XML-schema document can be included in the XML-document that describes a context. The elements and the edges of a concept hierarchy are represented via XML-schema constructs as follows:

1. Each node of a concept hierarchy is represented either as a `complexType` or as an `element`. For example, the concept "Man" and its instance "Paolo" would be represented as follows:

```
<complexType name="Man"/>
<element name="Paolo" type="Man"/>
```

2. The relation *is-a* is represented via the element `extension`. Thus, the relation "Man *is-a* Mortal" is represented as follows:

```
<complextype name="Man">
  <extension base="Mortal"/>
</complexType>
```

3. The relation *part-of* is represented through a combination of the primitives `element` and `extension`. For example, to declare that "Keyboard", "Monitor", and "Case" are elements of a "Personal Computer", one can use the following XML-schema:

```
<complextype name="PC">
  <extension base="anything">
    <element name="keyboard"
             type="PCKeyboard"/>
    <element name="monitor"
             type="PCMonitor"/>
    <element name="case"
             type="PCCase"/>
  </extension>
</complexType>
```

4. The relation *inst-of* is represented through the primitive `element`. The example shows the declaration of the fact that that "NokiaAB *inst-of* PCMonitor".

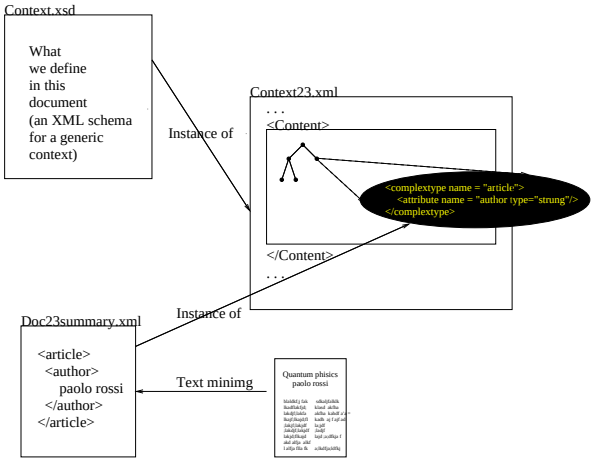

Figure 1: CTXML

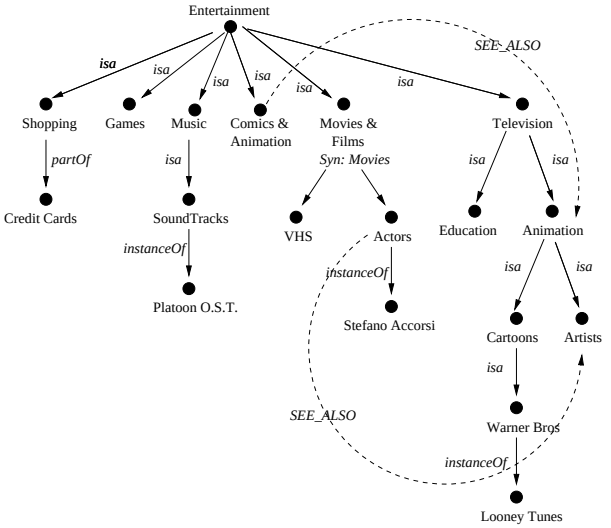

Figure 2: The context structure 'Entertainment'

```
<element name="NokiaAB"
         type="PCMonitor"/>
```

In order to validate a context, we can write down a DTD, or another XML-schema, that describes the structure of the XML-schema for Concept Hierarchies. For uniformity reasons, we chose to express the structure of a context in an XML-schema document. An overview of the intuition underlying the concrete representation of the content (i.e. the explicit representation) of a context is depicted in Figure 1. `Context.xsd` is an XML schema that describes the structure of a generic context; any context can be specified as an XML document (e.g., `context23.xml`) of the type described by `context.xsd`.

An example

In this section we present an example of two contexts that represent the domain of Entertainment; the conceptual struc-

tures are portions of the web directory structures of two well-known Internet search engines. The graphical representation of the two structures is shown in Figure 2 and Figure 4, respectively.

Hierarchical arcs of both context structures are not labelled by any relation, as this information is left implicit. These relations can be guessed on the basis of the semantics of the labels of the nodes. In the pictures above, we deliberately added relation labels as an example. Furthermore, both contexts contain a non-hierarchical relation, called "SEE_ALSO".

As we said, a context is represented as an XML document composed of two parts: header and content (i.e., the concept hierarchy). The context header of 'Entertainment' is described below, while the the concept hierarchy is represented in Figure 3. For the sake of simplicity, we chose to report in Figure 3 only a subset of concepts and relations of the original concept hierarchy.

```
<ctxHeader ctxId="cxt01"
           label="Entertainment"
           add-manager="antodona"
           status="draft">
<owner>
  <agentId>Agent02</agentId>
</owner>
<group>
  <groupId>Entertainment</groupId>
</group>
<security>
  <accessRights>read</accessRights>
  <encription>none</encription>
</security>
<history>
  <version>v1.0</version>
  <generatedFrom>
    <operationType>scratch</operationType>
    <elementUri/>
  </generatedFrom>
</history>
</ctxHeader>
```

In Figure 3, one can easily recognize a piece of XML-Schema with definitions of new `complexType` and `element` as components of a new schema. The last part of the document is the reference to existing mappings for the defined context:

```
<mappingForCtx mapId="map01"/>
```

The other selected context is the one of the 'Arts and Entertainment'. The concept hierarchy is depicted in Figure 4. For lack of space, we do not provide the XML code of this context.

Mappings

The explicit representations of two contexts can describe a common portion of the world. For instance, a context about "cars components" and a context about "radio and hi-fi" might overlap on "radio and hi-fi for cars". Despite this fact, it is not guaranteed that the common part about "radio and

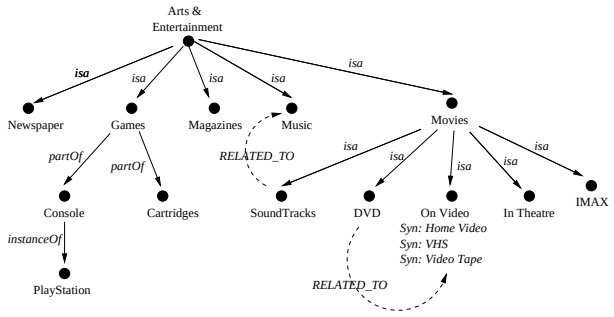

Figure 4: The context structure 'Arts and Entertainment'

hi-fi for cars" is conceptualized in the same way in the two contexts. Nonetheless, the concepts in the two contexts are in some relation. Of course, it is essential that we provide a way of defining such a relation between concepts of the two contexts (for example, to return relevant documents from a context as an answer to a query asked in the other context). This is achieved by introducing the idea of a *context mapping*. In this section we give a data structure for *context mapping*.

The general intuitions behind a context mapping are the following:

- a context mapping is *directional*. Indeed, we want to be able to represent the situation in which a context c_1 imports information from another context c_2 using a certain context mapping m, without forcing c_2 to use the same (or the inverse) mapping m to import the information for context c_1.
- a context mapping should not be limited to the representation of equivalence between concepts of two different contexts. It should allow for the representation of relations between concepts at different abstraction level. For instance, a context mapping should be able to represent the fact that the concept "printer" in a context is more general than the concept "laser printer" in some other context.

A mapping is a relation between a context (called *source*) and another context (called *target*), and it can be formally represented by a 4-tuple $\langle m, c_s, \mathcal{M}, c_t \rangle$, where m is a unique identifier for the mapping; c_s and c_t are two different context identifiers, for the source and the target context respectively; $\mathcal{M}$ is the real mapping, i.e. the actual relation between the explicit representations of c_s and c_t.

Let us define the **abstract structure** of mappings between two context hierarchies.

Definition 2. A context mapping is a 4-tuple $\langle m, c_s, \mathcal{M}, c_t \rangle$, where:

1. m a unique identifier associated with a mapping;
2. c_s and c_t are distinct context identifiers, called *source* and *target* context;
3. $\mathcal{M}$ is *concept mapping* from the content of the source context to the content of the target context. Where a context mapping from a concept hierarchies $H_s = \langle C_s, E_s \rangle$ to $H_t =$

```
<ctxContent language="en">
<CHContent>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://.../entertainmentYahoo">
  <complexType name="Entertainment">
    <annotation>
      <appinfo><label>Entertainment</label></appinfo>
   </annotation>
  </complexType>
  <complexType name="Entertainment_Shopping">
    <annotation>
      <appinfo><label>Shopping</label></appinfo>
      <documentation> This is the node Shopping which IS_A Entertainment </documentation>
    </annotation>
    <complexContent>
      <extension base="Entertainment">
        <element name="CreditCards" type="Entertainment_Shopping_CreditCards"/>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="Entertainment_Shopping_CreditCards">
    <annotation>
      <appinfo><label>CreditCards</label></appinfo>
      <documentation> This is the node CreditCards which is
      PART_OF the node Shopping </documentation>
    </annotation>
  </complexType>
  <complexType name="Entertainment_Music">
    <annotation>
      <appinfo><label>Music</label></appinfo>
      <documentation> This is the node Music which IS_A Entertainment </documentation>
    </annotation>
    <complexContent>
      <extension base="Entertainment"/>
    </complexContent>
  </complexType>
  <complexType name="Entertainment_Music_SoundTracks">
    <annotation>
      <appinfo><label>SoundTracks</label></appinfo>
      <documentation>
        This is the node SoundTracks which IS_A Entertainment_Music
      </documentation>
    </annotation>
    <complexContent>
      <extension base="Entertainment_Music"/>
    </complexContent>
  </complexType>
  <element name="PlatoonO.S.T." type="Entertainment_Music_SoundTracks">
    <annotation>
      <appinfo>
        <label>PlatoonO.S.T</label>
      </appinfo>
      <documentation>
        This is the node PlatoonO.S.T which is an
        INSTANCE_OF Entertainment_Music_SoundTracks
      </documentation>
    </annotation>
  </element>
</schema>
</CHContent>
</ctxContent>
```

Figure 3: XML for the concept hierarchy

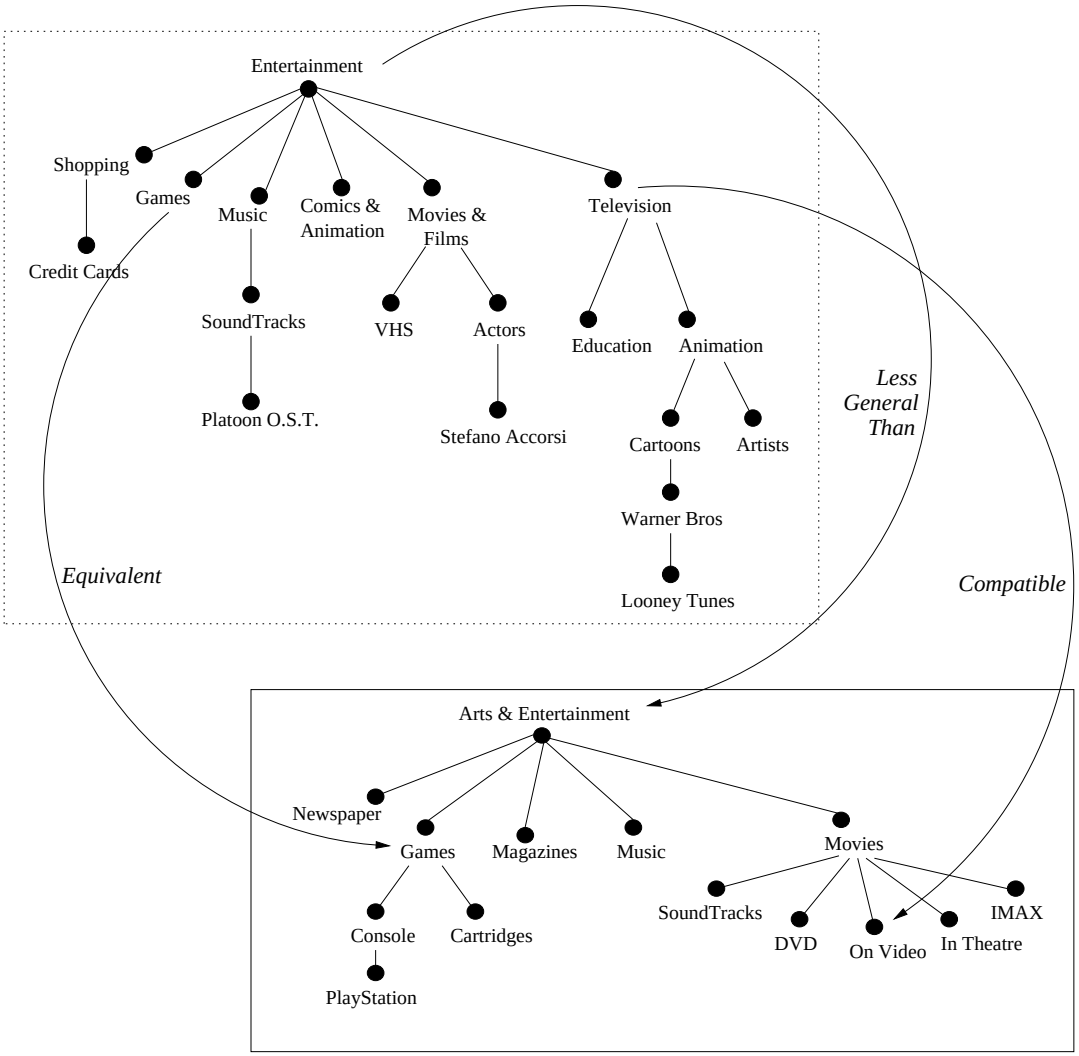

Figure 5: Mapping between 'Entertainment' and 'Arts and Entertainment'

$\langle C_t, E_t \rangle$, is a tuple of relations $\left\langle \xrightarrow{\sqsupseteq}, \xrightarrow{\sqsubseteq}, \xrightarrow{\perp}, \xrightarrow{*}, \xrightarrow{\equiv} \right\rangle$ each of which is a subset of $C_s \times C_t$.

Intuitively $c_1 \xrightarrow{\sqsupseteq} c_2$ means that c_1 is more general than c_2 (e.g., animal is more general than dog); $c_1 \xrightarrow{\sqsubseteq} c_2$ means that c_1 is less general than c_2; $c_1 \xrightarrow{\equiv} c_2$ means that $c_1 \xrightarrow{\sqsubseteq} c_2$ and $c_1 \xrightarrow{\sqsupseteq} c_2$; $c_1 \xrightarrow{\perp} c_2$ means that c_1 is disjoint from c_2 (e.g., mountain is disjoint from sea), $c_1 \xrightarrow{*} c_2$ means that c_1 is compatible with c_2 (e.g., cars are compatible with hi-fi , as there are hi-fi for cars).

Distributed Description Logics (Borgida & Serafini 2002) provides a basic formal framework in which multiple concept hierarchies related mappings can be described.

The *concrete representation* of a mapping is an XML document which structure is described by the another XML-schema. Similary to contexts, a concrete mapping is composed of two parts: a header, which contains attributes of the mapping (e.g., an identifiers, the ids of the source and target contexts, and so on); and a content, namely the actual mapping between the explicit representations of the source and target contexts.

An example

Figure 5 shows a graphical representation of a subset of the possible relations between concepts of 'Entertainment' and concepts of 'Arts and Entertainment'.

As contexts, mappings are represented as an xml file, instance of a defined XML-Schema, and composed by two parts: the header and the set of relations. For the example of Figure 5 the header is the following, while the content of the mapping is represented in Figure 6. Note that there are examples of $\xrightarrow{\sqsubseteq}$ (Less General Than), $\xrightarrow{*}$ (Compatible), $\xrightarrow{\equiv}$ (Equivalent) relations between concepts. The definition of all the other relation is similar except for the name.

```
<mapHeader mapId="map01">
<owner>
```

```
<mapContent>
<CH2CHContent>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://.../Yahoo2Epinions"
        xmlns:ctxSource="http://.../entertainmentYahoo"
        xmlns:ctxTarget="http://.../artsEntertainmentEpinions">
  <complexType name="lessGeneralThan">
    <choice>
      <element name="pair1">
        <complexType>
          <sequence>
            <element type="ctxSource:Entertainment" name="source"/>
            <element type="ctxTarget:artsEntertainment" name="target"/>
          </sequence>
        </complexType>
      </element>
    </choice>
  </complexType>
  <complexType name="equivalent">
    <choice>
      <element name="pair1">
        <complexType>
          <sequence>
            <element type="ctxSource:Entertainment_Games" name="source"/>
            <element type="ctxTarget:artsEntertainment_Games" name="target"/>
          </sequence>
        </complexType>
      </element>
    </choice>
  </complexType>
  <complexType name="compatible">
    <choice>
      <element name="pair1">
        <complexType>
          <sequence>
            <element type="ctxSource:Entertainment_Television" name="source"/>
            <element type="ctxTarget:artsEntertainment_Movies_OnVIdeo" name="target"/>
          </sequence>
        </complexType>
      </element>
    </choice>
  </complexType>
</schema>
</CH2CHContent>
</mapContent>
```

Figure 6: XML for mapping

```
  <agentId>Agent02</agentId>
</owner>
<group>
  <groupId>Entertainment</groupId>
</group>
<security>
  <accessRights>modify</accessRights>
  <encription>none</encription>
</security>
<history>
  <version>v1.0</version>
  <generatedFrom>
    <operationType>scratch</operationType>
    <elementUri/>
  </generatedFrom>
  <generatesList/>
</history>
<mapSource ctxId="cxt01" ctxVersion="v1.0"/>
<mapTarget ctxId="ctx02" ctxVersion="v1.0"/>
</mapHeader>
```

Conclusions

In this paper we described a theoretical framework for contexts and mapping for distributed knowledge management. We also provide a concrete language that allow to specify contexts and mappings. In a further work (also presented to this workshop) we have outlined the main ideas of a linguistic based algorithm for automatic discovering of mappings between contexts.

References

Benerecetti, M.; Bouquet, P.; and Ghidini, C. 2000. Contextual Reasoning Distilled. *Journal of Theoretical and Experimental Artificial Intelligence* 12(3):279–305.

Boland, J., and R.V.Tenkasi. 1995. Perspective making and perspective taking in communities of knowing. *Organizational Science* 6(4):350–372.

Bonifacio, M.; Bouquet, P.; and Traverso, P. 2002. Enabling distributed knowledge management. managerial and technological implications. *Novatica and Informatik/Informatique* III(1).

Borgida, A., and Serafi ni, L. 2002. Distributed description logics. In Horrocks, I., and Tessaris, S., eds., *Proceedings of the 2002 Intl. Workshop on Description Logics (DL2002)*. Toulouse: CEUR-WS. ONLINE: `http://CEUR-WS.org/Vol-53/`, ARCHIVE: `ftp://SunSITE.Informatik.RWTH-Aachen.DE/pub/publications/CEUR-WS/Vol-53.tar.gz`.

Büchner, A.; Ranta, M.; Hughes, J.; and Mäntylä, M. 1999. Semantic information mediation among multiple product ontologies. In *Proc. 4th World Conference on Integrated Design & Process Technology*.

Corcho, O., and Pérez, A. G. A roadmap to ontology specifi cation language. In *Knowledge Engineering and Knowledge Management. Methods, Models and Tools. Prooceedings of the 12th International Conference, EKAW 2000*, 80–96.

Drucker, P. 1994. *Post Capitalist Society*. Cambridge University Press.

Ghidini, C., and Giunchiglia, F. 2001. Local models semantics, or contextual reasoning = locality + compatibility. 127(2):221–259.

Johnson-Laird, P. N. 1983. *Mental Models*. Cambridge University Press.

Kuhn, T. 1979. *The structure of Scientific Revolutions*. University of Chicago Press.