Towards a hybrid approach to context modelling, reasoning and interoperation

Karen Henricksen
CRC for Enterprise Distributed Systems Technology (DSTC)
kmh@dstc.edu.au

Steven Livingstone and Jadwiga Indulska
School of Information Technology and Electrical Engineering,
The University of Queensland
{srl,jaga}@itee.uq.edu.au

ABSTRACT

Increasing interest in ontologies in the last few years has led to a shift in the research focus of the context-awareness community. This shift reflects the potential of ontology-based approaches to improve upon previously used context modelling approaches by providing improved support for interoperation and sophisticated types of reasoning. Unfortunately, recently developed context-aware systems built using ontologies fall short of this potential. The goals of this paper are to investigate the reasons for this shortfall, to evaluate the most appropriate uses of ontology languages and tools in context-aware systems, and to explore the creation of a new hybrid solution that combines ontology concepts with our previously developed approach to context modelling and reasoning.

1. INTRODUCTION

The challenges associated with constructing context-aware applications for ubiquitous computing environments, and the importance of appropriate abstractions for gathering and reasoning about context information, are well recognised. Early efforts to develop abstractions and infrastructure for context-aware systems focused primarily on reusable components for combining and interpreting sensor outputs to derive high-level context information [1, 2]. More recently, efforts have turned to developing models of context that integrate information from a variety of sources, support interoperation of context-aware applications and context management systems, and allow reasoning about context.

Early efforts to model context in a formal way built upon data modelling techniques developed by the information systems community, and implemented sophisticated context management systems as extensions to relational databases [3–5]. The strengths of these solutions included efficient query processing and support for advanced query types[1].

These formal approaches were the first to provide reasoning support. Our modelling approach, in particular, was developed to enable reasoning using a form of three-valued logic designed to accommodate ambiguous and otherwise imperfect context information [6].

[1]For example, Harter et al. [3] supported spatial queries, while we provided situation and preference evaluation in addition to simple fact queries [5].

More recently, the increasing popularity of ontologies has led to new ontology-based models of context. These aim to better support interoperation by formally defining common concepts (and relationships and mappings between these), and to leverage the existing tool support for tasks such as ontology definition and checking. The ontology-based approaches also have the potential to support sophisticated ontology-based reasoning; however, immature standards and tools, and computability and tractability problems associated with reasoning over ontology languages such as OWL Full [7] and SWRL [8], mean that this potential has not yet been realised. Further, the ontology-based approaches generally do not provide a natural way to reason over some common types of context information, including imperfect information.

As there are clear advantages to both the ontology-based and earlier context modelling approaches, we argue that each approach has a place in context-aware systems. We are currently exploring the integration of the two modelling techniques to form a hybrid solution that combines interoperability support and various types of ontology-based reasoning with the advantages of our previously developed context model based on Object-Role Modeling (ORM) [9]. This paper presents the early results of our efforts to map our context modelling constructs to ontologies, and to evaluate the types of reasoning we can support by augmenting the ontologies with appropriate rules. We focus our attention on the OWL standard [7] produced by the W3C (and, in particular, on OWL DL, as it is reasonably expressive while being both computationally complete and decidable) and - to a lesser extent - on SWRL (the Semantic Web Rule Language) [8], an emerging standard developed by the DAML group to combine OWL with the Rule Markup Language (RuleML). Although a variety of ontology languages and logic variants are currently used for context modelling and reasoning, as discussed in Section 3, OWL is arguably the most popular. A full evaluation of all available ontology languages and associated reasoning tools is clearly outside our scope.

The structure of the paper is as follows. Section 2 presents an overview of our previously developed context modelling approach based on ORM, while Section 3 reviews a variety of ontology-based context models. Section 4 evaluates the ways in which our context

modelling approach presented in Section 2 can be augmented with ontology concepts to form a hybrid approach that supports additional types of reasoning and enables interoperation based on mappings between context models. Finally, Section 5 concludes the paper with a discussion of our experiences and future work.

2. OVERVIEW OF OUR MODELLING APPROACH

This section briefly reviews our previously developed approach to context modelling. Further information can be found in our earlier publications [5, 6, 10].

We model context at two levels of detail, using fact and situation abstractions. Section 2.1 presents our fact-based context modelling approach, the Context Modelling Language (CML), while Section 2.2 describes how we represent abstract situations in terms of fact types modelled with CML.

2.1 Fact-based context modelling

Our fact-based context modelling approach primarily serves as a tool that enables application developers to explore and formally specify the context requirements of a context-aware application. It provides constructs for defining the objects about which context information is required and the types of information (or facts) that are of interest in relation to each object. It also allows developers to identify an appropriate source for each fact type (sensors, user profiles or derivation from other context information), specify dependencies and constraints, and explore information quality issues [6].

As discussed in Section 1, our approach is based on ORM, a graphical modelling notation developed for conceptual modelling of information systems. ORM represents object types as ellipses and relations on one or more object types as fact types, drawn as sequences of role boxes (where each role is attached to the object type that participates in the role). Each object type is assigned a name as well as a reference mode, shown in parentheses, that describes how instances of the type are represented. Fact types are annotated with uniqueness constraints, represented as double-headed arrows spanning one or more role boxes; these place restrictions on the populations of the fact types in the manner of key constraints on attributes of relations in the relational data model. A variety of other constraints are also supported, but a discussion of these falls outside the scope of this paper.

CML introduces a variety of extensions to this basic notation, illustrated in an example model shown in Figure 1. The example builds on a context model that we defined for a communication application that assists users with the selection of appropriate communication channels for their interactions with other people. For this application, the most relevant types of context information include associations of users to communication channels and devices (e.g., ownership, permissions and proximity in the case of devices), current locations of users, and current and planned activities.

The CML extensions allow fact types to be labelled as:

- *static* (s), *sensed* ($\wedge\!\!\wedge$), *derived* (*) or *profiled* ($\circ$) types, depending on persistence and source;
- *temporal* ([ ]) types that capture histories of information (e.g., user activity over a day or week); and
- *alternative* (a) types that are capable of describing ambiguous information (e.g., conflicting location reports gathered from a variety of location sensors).

CML also provides extensions to support special constraints on temporal and alternative fact types, annotation of fact types with appropriate metadata types for describing quality, and dependencies between pairs of fact types (e.g., between a person's activity and their current location, to indicate that location changes are commonly linked to activity changes).

2.2 Situation-based context modelling

In addition to modelling context at the fact level using CML, we allow situations to be defined in order to describe the context at a higher level of abstraction. These are expressed in terms of the basic fact types defined by a CML model, using a variant of predicate logic. Situations can be combined using logical connectives to form increasingly rich context descriptions. This feature makes them useful as programming abstractions that allow the software engineer to predicate application behaviour on simple situation expressions in a very natural way.

Situations are written as predicates on zero or more variables, then evaluated against a set of variable bindings and a context (represented as a set of facts) to yield one of the values *true*, *false* or *possibly true*. The *possibly true* value arises when the available context information is inadequate to determine absolute truth or falsity (e.g., because of incompleteness or ambiguity). An example situation is shown later in the paper, in Figure 5 (a). This describes the circumstances under which a person can use a given communication channel, in terms of the `RequiresDevice`, `LocatedNear` and `PermittedToUse` fact types defined in the CML model in Figure 1.

3. ONTOLOGY-BASED MODELS OF CONTEXT

Ontology-based models of context have been independently developed by several research groups, including Chen et al. [11], Gandon and Sadeh [12], Strang et al. [13] and Wang et al. [14]. This section briefly reviews these models.

The proposals of Chen et al. and Gandon and Sadeh are unique in that they use ontologies for modelling both context information and users' privacy policies. Chen et al. base their proposal around their Context Broker Architecture (CoBrA) designed for use in smart spaces and a broad set of OWL ontologies they are developing for modelling physical locations, devices, temporal concepts, privacy requirements and a variety of other domains [15]. CoBrA employs reasoning for detecting

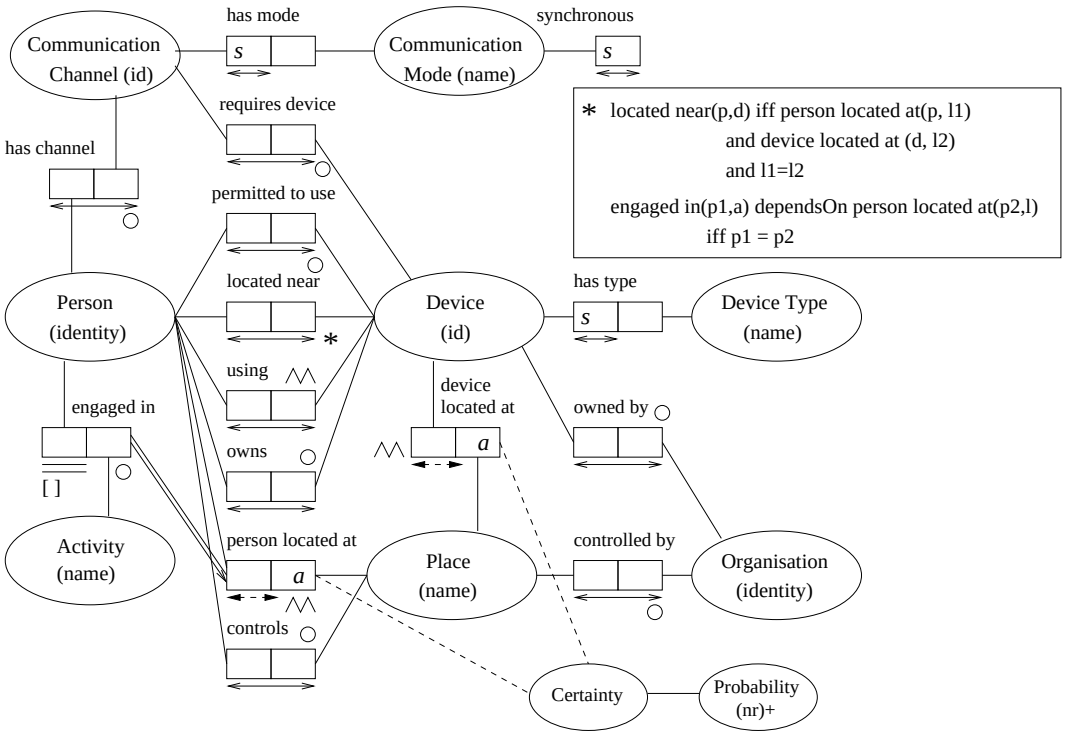

Figure 1: Modelling context for a communication application using CML.

and resolving inconsistent context information, evaluating privacy policies, and inferring additional context information based on properties such as temporal and spatial relations. As CoBrA's reasoning over context information is currently based purely on OWL without additional rule support, it is quite limited.

The e-Wallet proposal of Gandon and Sadeh collects together the context information for individual users into repositories (e-Wallets) consisting of static information, rules for mapping context attributes onto service invocations, and rules describing privacy requirements. This proposal has a much stronger privacy bias than that of CoBrA, and focuses on reasoning for enforcement of privacy policies, rather than reasoning about context information alone.

The CoOL ontology language developed by Strang et al. is based on an Aspect-Scale-Context model. Aspects represent classifications (e.g., TemperatureAspect), while scales are individual dimensions of aspects (e.g., CelciusScale, FahrenheitScale). Context information is attached to a particular aspect and scale, and quality metadata (e.g., meanError or timestamp) is associated with information via quality properties. Mappings between the scales belonging to a single aspect are defined as IntraOperations, while mappings between the scales of different aspects are supported by InterOperations.

For reasoning, Strang et al. use F-Logic and the OntoBroker reasoner. Reasoning is used for evaluating context queries, checking consistency of context information, deriving implicit information from rules and reasoning over inter-ontology relationships.

Finally, Wang et al. propose an extensible context ontology called CONON, which is designed as an extensible model that can be refined using OWL's subtyping mechanisms. CONON supports two types of reasoning: reasoning to detect and correct inconsistent context information, and reasoning as a means to derive higher level context information. The latter type of reasoning is based on both properties such as symmetry and transitivity and user-defined rules.

These ontology-based context modelling approaches currently have several shortcomings. First, the ontology standards on which they are based (and their respective reasoning tools) remain somewhat immature; OWL, for example, currently does not provide direct support for axiomatic rules, which limits the types of reasoning that are possible with OWL alone. Second, none of the approaches adequately addresses reasoning over imperfect context information (although CoOL does address the modelling of quality metadata, and most of the approaches incorporate forms of reasoning that attempt to resolve inconsistent context information). Finally, the process of creating or extending context ontologies is often complex and error prone, as the ontology languages are often verbose and unintuitive[2]. One way to address this problem is to combine the ontology-based modelling approaches with other, more natural solutions, such as graphical context modelling approaches.

[2]This argument is borne out by the OWL and SWRL examples presented later in this paper.

4. TOWARDS A HYBRID MODELLING APPROACH

As a means to overcome the current shortcomings of the ontology-based context models, we have been exploring the integration of ontology concepts with the context modelling approach presented in Section 2 to produce a superior hybrid solution for context modelling and reasoning. We began this exploration by mapping our context modelling concepts to an OWL DL ontology, and then using this to transfer the example context model shown in Figure 1 to an OWL representation. We present the results of these steps in Section 4.1. Next, we investigated forms of reasoning that could be built on top of the OWL model, including both reasoning about the context and reasoning about context models. We describe our findings in Section 4.2.

4.1 Mapping from CML to OWL DL

In the process of transferring the context model shown in Figure 1 to OWL DL, we created two ontologies. The first defines CML's modelling constructs in terms of OWL classes and properties. This ontology has two components: a set of concepts for creating CML models, and a set of concepts for creating instances of these. The first set, shown in Figure 2, defines fact types, classifications that can be attached to these, and representations for dependencies and quality annotations. The second, shown in Figure 3, defines object and fact concepts, including a temporal fact that has special properties to represent start and end times.

We used the concepts of the CML ontology to define our example model as a second ontology. A small excerpt is shown in Figure 4. Again, this ontology has two parts: one describing properties of the model (effectively, metadata), and another defining object and fact classes that support instantiation of the model.

In defining these ontologies, we made several observations about the relative merits of CML-based and OWL-based modelling. First, we were not able to completely map the CML concepts to OWL DL. In particular, we had difficultly finding adequate representations for ORM's uniqueness/key constraints, as these capture more complex cases than OWL's cardinality constraints, and relationships between alternative facts[3] (although the latter can be represented by SWRL rules).

We also found the process of creating the OWL DL model to be complex and error prone (even with appropriate tool support), and the result to be extremely verbose and unwieldy. In contrast, the CML model was easy to create, and the notation is considerably more readable by humans. Fortunately, the OWL DL representation can be produced mechanically from the CML model. We have already developed a tool that translates CML models, expressed in terms of a simple context schema notation, into a variety of alternative representations [16]. We can easily extend this tool to generate appropriate OWL ontologies.

[3]It is, of course, possible to explicitly assert relationships amongst sets of alternative facts, but this approach is inferior to the one in which the relationships are inferred.

```
<owl:Class rdf:ID="FactType"/>

<owl:Class rdf:ID="StaticFactType">
   <rdfs:subClassOf rdf:resource="#FactType"/>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#classification"/>
         <owl:hasValue rdf:resource="#static"/>
      </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#classification"/>
         <owl:cardinality
          rdf:datatype="&xsd;nonNegativeInteger">
            1
         </owl:cardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="FactTypeClassification">
   <owl:oneOf rdf:parseType="Collection">
      <owl:Thing rdf:about="#static"/>
      <owl:Thing rdf:about="#sensed"/>
      <owl:Thing rdf:about="#derived"/>
      <owl:Thing rdf:about="#profiled"/>
      <owl:Thing rdf:about="#temporal"/>
      <owl:Thing rdf:about="#alternative"/>
   </owl:oneOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="classification">
   <rdfs:domain rdf:resource="#FactType"/>
   <rdfs:range rdf:resource="#FactTypeClassification"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="dependsOn"/>
   <rdf:type rdf:resource="&owl;TransitiveProperty"/>
   <rdfs:domain rdf:resource="#FactType"/>
   <rdfs:range rdf:resource="#FactType"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="QualityParameter"/>

<owl:Class rdf:ID="QualityMetric"/>

<owl:ObjectProperty rdf:ID="qualityParameter">
   <rdf:domain rdf:resource="#FactType"/>
   <rdf:range rdf:resource="#QualityParameter"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="qualityMetric">
   <rdf:domain rdf:resource="#QualityParameter"/>
   <rdf:range rdf:resource="#QualityMetric"/>
</owl:ObjectProperty>
```

Figure 2: Subset of an OWL DL representation of the CML modelling concepts, supporting model description.

4.2 Reasoning on the OWL DL representation of context

The OWL DL representation of our context model presented in the previous section can support several types of reasoning, including:

- reasoning about the current context in order to derive additional context information;
- reasoning about the context model for consistency/error checking; and

```
<owl:Class rdf:ID="CMLObject">
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#value"/>
         <owl:cardinality
          rdf:datatype="&xsd;nonNegativeInteger">
            1
         </owl:cardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="value">
   <rdfs:domain rdf:resource="#CMLObject"/>
   <rdfs:range rdf:resource="&owl;Thing"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Fact"/>

<owl:ObjectProperty rdf:ID="role">
   <rdfs:domain rdf:resource="#Fact"/>
   <rdfs:range rdf:resource="#CMLObject"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="TemporalFact">
   <rdfs:subClassOf rdf:resource="#Fact"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="startTime">
   <rdfs:domain rdf:resource="#TemporalFact"/>
   <rdfs:range rdf:resource="&xsd;dateTime"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="endTime">
   <rdfs:domain rdf:resource="#TemporalFact"/>
   <rdfs:range rdf:resource="&xsd;dateTime"/>
</owl:DatatypeProperty>
```

Figure 3: Subset of an OWL DL representation of the CML modelling concepts, supporting model instantiation.

- reasoning about relationships between different context models to support interoperation.

We briefly discuss these three types of reasoning in the following subsections.

4.2.1 Reasoning about context

Ontology-based reasoning is used in context modelling approaches such as CONON [14] for deriving new context information on the basis of both OWL-defined concepts (such as transitive and symmetric properties) and user-defined rules. This style of derivation is not unique to the ontology-based reasoning approaches, however, and is already supported in our context modelling approach in two ways. First, CML allows derived fact types to be defined in the manner already supported by ORM. Second, our situation abstraction - described earlier in Section 2.2 - allows high-level context descriptions (situations) to be defined recursively in terms of other situations, as well as in terms of the fact types captured by a CML model. Our situation abstraction is particularly powerful, as it supports several forms of universal and existential quantification (including special forms that operate over ambiguous information), as well as arbitrary user-defined functions.

(a)

```
<cml:QualityMetric rdf:ID="Probability"/>
<cml:QualityParameter rdf:ID="Certainty">
   <cml:qualityParameter rdf:resource="#Probability"/>
</cml:QualityParameter>

<cml:FactType rdf:ID="PersonLocatedAtFactType">
   <cml:classification rdf:resource="&cml;alternative"/>
   <cml:classification rdf:resource="&cml;sensed"/>
   <cml:qualityMetadata rdf:resource="#Certainty"/>
</cml:FactType>
```

(b)

```
<owl:Class rdf:ID="Person">
   <rdfs:subClassOf rdf:resource="&cml;CMLObject/>
</owl:Class>

<owl:Class rdf:ID="Place">
   <rdfs:subClassOf rdf:resource="&cml;CMLObject/>
</owl:Class>

<owl:Class rdf:ID="PersonLocatedAt">
   <rdfs:subClassOf rdf:resource="&cml;Fact/>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty
          rdf:resource="#PersonLocatedAtPerson"/>
         <owl:cardinality
          rdf:datatype="&xsd;nonNegativeInteger">
            1
         </owl:cardinality>
      </owl:Restriction>
      <owl:Restriction>
         <owl:onProperty
          rdf:resource="#PersonLocatedAtPlace"/>
         <owl:cardinality
          rdf:datatype="&xsd;nonNegativeInteger">
            1
         </owl:cardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="PersonLocatedAtPerson">
   <rdfs:subPropertyOf rdf:resource="&cml;role"/>
   <rdfs:domain rdf:resource="#PersonLocatedAt"/>
   <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="PersonLocatedAtPlace">
   <rdfs:subPropertyOf rdf:resource="&cml;role"/>
   <rdfs:domain rdf:resource="#PersonLocatedAt"/>
   <rdfs:domain rdf:resource="#Place"/>
</owl:ObjectProperty>
```

Figure 4: An OWL representation of selected object and fact types from our example context model. (a) shows an excerpt of the model definition, while (b) defines classes with which to instantiate the model.

We performed a comparison of our situation abstraction with an ontology-based reasoning approach by attempting to map some example situations to SWRL rules. An example mapping is shown in Figure 5. Because of inherent differences between the logics supported by SWRL and our situation abstraction, perfect mappings were not always possible (e.g., when ambiguity was involved).

(a)

```
CanUseChannel(person, channel) :
  forall device
    . RequiresDevice[channel, device]
    . LocatedNear[person, device] and
      PermittedToUse[person, channel]
```

(b)

```
<ruleml:imp>
   <ruleml:_body>
      <swrlx:classAtom>
         <owlx:Class owlx:name="&cml;RequiresDevice"/>
         <ruleml:var>requiresDevice</ruleml:var>
      </swrlx:classAtom>
      <swrlx:individualPropertyAtom
       swrlx:property="&cml;RequiresDeviceChannel">
         <ruleml:var>requiresDevice</ruleml:var>
         <ruleml:var>channel</ruleml:var>
      </swrlx:individualPropertyAtom>
      <swrlx:individualPropertyAtom
       swrlx:property="&cml;RequiresDeviceDevice">
         <ruleml:var>requiresDevice</ruleml:var>
         <ruleml:var>device</ruleml:var>
      </swrlx:individualPropertyAtom>
      <swrlx:classAtom>
         <owlx:Class owlx:name="&cml;LocatedNear"/>
         <ruleml:var>locatedNear</ruleml:var>
      </swrlx:classAtom>
      <swrlx:individualPropertyAtom
       swrlx:property="&cml;LocatedNearPerson">
         <ruleml:var>locatedNear</ruleml:var>
         <ruleml:var>person</ruleml:var>
      </swrlx:individualPropertyAtom>
      <swrlx:individualPropertyAtom
       swrlx:property="&cml;LocatedNearDevice">
         <ruleml:var>locatedNear</ruleml:var>
         <ruleml:var>device</ruleml:var>
      </swrlx:individualPropertyAtom>
      <swrlx:classAtom>
         <owlx:Class owlx:name="&cml;PermittedToUse"/>
         <ruleml:var>permittedToUse</ruleml:var>
      </swrlx:classAtom>
      <swrlx:individualPropertyAtom
       swrlx:property="&cml;PermittedToUsePerson">
         <ruleml:var>permittedToUse</ruleml:var>
         <ruleml:var>person</ruleml:var>
      </swrlx:individualPropertyAtom>
      <swrlx:individualPropertyAtom
       swrlx:property="&cml;PermittedToUseDevice">
         <ruleml:var>permittedToUse</ruleml:var>
         <ruleml:var>device</ruleml:var>
      </swrlx:individualPropertyAtom>
   </ruleml:_body>
   <ruleml:_head>
      <swrlx:individualPropertyAtom
       swrlx:property="CanUseChannel">
         <ruleml:var>person</ruleml:var>
         <ruleml:var>channel</ruleml:var>
      </swrlx:individualPropertyAtom>
   </ruleml:_head>
</ruleml:imp>
```

Figure 5: Mapping a situation definition (a) to an SWRL rule (b). The semantics of the `CanUseChannel` situation are roughly as follows: a person p can use a communication channel c iff all of the requisite computing devices are near p and p has permission to use them.

As a result of our experiences, we believe that SWRL offers no significant benefits over our situation abstraction for deriving high-level context information. However, we can make several arguments in favour of our situation abstraction. As seen from the example in Figure 5, the situation logic is substantially cleaner and clearer than the SWRL rule notation. This logic makes the existential and universal quantifiers explicit, an approach that we have found to be more user-friendly than that of SWRL. All SWRL variables are implicitly assumed to be universally quantified; in order to capture existential quantification, the OWL someValuesFrom restriction must be used. SWRL also suffers from well known computability and tractability problems which imply that it is likely to be some time before good reasoning tools are available.

Because of these problems, we intend to continue using our situation abstraction in our hybrid modelling approach, and concentrate on the use of ontologies purely for reasoning about context models as discussed in the following sections.

4.2.2 Reasoning for model checking

One of the main strengths of ontology languages such as OWL is their ability to formally define relationships between concepts, such as equivalence of classes and properties, and subclass and subproperty relationships. OWL also supports a variety of constraints such as restrictions on property values (using elements such as `allValuesFrom` and `someValuesFrom`) and members of classes (using `oneOf` and `disjointWith`, etc.). These features can be used to enable reasoning about, and validation of, ontology-based context models in a manner that is currently not possible with other context modelling approaches. We discuss the use of this type of reasoning for model checking in this section, and for interoperation between models in the following section.

When mapping our CML concepts to an OWL description (as discussed in Section 4.1), we used OWL's constraint support to capture the semantics of the concepts so as to provide maximal opportunities for error and consistency checking in our CML models. Figure 2 shows a simple example in which static fact types (defined as any fact type which has a 'static' value for the classification property) are precluded from also having any other classifications, such as temporal or sensed (as the cardinality of their classification property is always one). We can define similar constraints and rules to ensure that each fact type has a valid uniqueness constraint, there are no cyclic dependencies between fact types, and so on.

We intend to incorporate

1. mapping of CML models to an OWL DL representation; and
2. validation by an OWL reasoner

into the mapping tool we described in Section 4.1, in order to detect errors in the models before they are instantiated in a context management system and mapped to model-specific programming libraries as described in [16]. This feature will substantially improve the map-

ping tool, as errors in the models that can currently remain undetected until run-time (when the context management system raises an exception) will be discovered immediately.

4.2.3 Reasoning for interoperation

We also plan to use ontology-based reasoning over context models to support interoperability in context aware systems by enabling information to be transferred between different context models. OWL supports straightforward notions of equivalence between classes and properties which can be used to overcome simple naming differences between models, but rules can also be used to handle cases in which more complex mappings are required (e.g., when different units or representations are involved).

We believe that, in general, it will be reasonably straightforward to create mappings between different CML models, and more difficult (but still possible) to create mappings between CML models and entirely different context ontologies. We are currently investigating the extension of our context management infrastructure to support the first type of mapping.

5. DISCUSSION AND CONCLUDING REMARKS

The current ontology-based approaches to context modelling suffer from several shortcomings. The task of defining new context ontologies is cumbersome and error prone, and most of the previously defined ontologies lack support for representing and reasoning over imperfect context information. Further, they do not fully exploit the reasoning capabilities of ontology languages such as OWL and SWRL; most use reasoning as a means to derive additional context information based on a combination of simple rules and properties such as transitivity and commutativity, and to detect and correct inconsistencies in context information. However, we believe that the most interesting applications of ontology-based reasoning are not those that involve context information, but rather those that are concerned with context models. In this paper, we presented the results of our early efforts to extend our previously developed context modelling approach with OWL DL and SWRL descriptions of our models, and highlighted the utility of reasoning over these descriptions as a means to support model checking and interoperation.

Unfortunately, our efforts to validate this approach have been somewhat hampered by both the immaturity of the standards that we have been evaluating and a lack of tool support for reasoning (particularly in the case of SWRL). We will continue to monitor the status of the standards and tools, and will extend our current tool support for transformation and checking of CML-based context models in parallel with the latest developments in the ontology arena.

6. ACKNOWLEDGEMENTS

The work reported in this paper has been funded in part by the Co-operative Centre for Enterprise Distributed Systems Technology (DSTC) through the Australian Federal Government's CRC Programme (Department of Education, Science and Training).

REFERENCES

[1] Dey, A.K., Salber, D., Abowd, G.D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction **16** (2001) 97–166

[2] Chen, G., Kotz, D.: Context aggregation and dissemination in ubiquitous computing systems. In: 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), Callicoon (2002)

[3] Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P.: The anatomy of a context-aware application. In: 5th International Conference on Mobile Computing and Networking (MOBICOM), Seattle (1999) 59–68

[4] Henricksen, K., Indulska, J., Rakotonirainy, A.: Generating context management infrastructure from context models. In: 4th International Conference on Mobile Data Management (MDM) - Industrial Track, Melbourne (2003)

[5] Henricksen, K., Indulska, J.: A software engineering framework for context-aware pervasive computing. In: 2nd IEEE Conference on Pervasive Computing and Communications (PerCom), Orlando (2004)

[6] Henricksen, K., Indulska, J.: Modelling and using imperfect context information. In: Workshop on Context Modeling and Reasoning (CoMoRea), 2nd IEEE Conference on Pervasive Computing and Communications (PerCom), Orlando (2004) 33–37

[7] McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language - Overview. W3C Recommendation (2004)

[8] Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML, Version 0.5 (2003)

[9] Halpin, T.A.: Conceptual Schema and Relational Database Design. 2nd edn. Prentice Hall Australia, Sydney (1995)

[10] Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling context information in pervasive computing systems. In: 1st International Conference on Pervasive Computing (Pervasive). Volume 2414 of Lecture Notes in Computer Science., Springer (2002) 167–180

[11] Chen, H., Finin, T., Joshi, A.: Semantic web in the context broker architecture. In: 2nd IEEE Conference on Pervasive Computing and

Communications (PerCom), Orlando (2004) 277–286

[12] Gandon, F.L., Sadeh, N.M.: Semantic web technologies to reconcile privacy and context awareness. Web Semantics Journal **1** (2004)

[13] Strang, T., Linnhoff-Popien, C., Frank, K.: Integration issues of an ontology based context modelling approach. In: IADIS International Conference WWW/Internet (ICWI), Argarve (2003)

[14] Wang, X.H., Zhang, D., Gu, T., Dong, J., Pung, H.K.: Ontology based context modeling and reasoning using OWL. In: Workshop on Context Modeling and Reasoning (CoMoRea), 2nd IEEE Conference on Pervasive Computing and Communications (PerCom), Orlando (2004)

[15] Chen, H., Perich, F., Finin, T., Joshi, A.: SOUPA: Standard ontology for ubiquitous and pervasive applications. In: 1st International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous), Boston (2004)

[16] McFadden, T., Henricksen, K., Indulska, J.: Automating context-aware application development. In: Workshop on Advanced Context Modelling, Reasoning and Management, 6th International Conference on Ubiquitous Computing, Nottingham (2004)