Context Modeling and Reasoning using Ontologies (July 2007)

Feruzan Ay, University of Technology Berlin, feruzan@cs.tu-berlin.de

***Abstract*—This paper gives an introduction to context modeling and reasoning in the area of pervasive computing. Therefore the OWL Web Ontology Language is presented and it is shown how it can be used for reasoning purposes. Three different approaches to context modeling and reasoning basing upon OWL are presented: CONON, COBRA-ONT and SOUPA.**

***Index Terms*—context awareness, ontology, mobile computing, pervasive computing, reasoning, ubiquitous computing.**

I. Introduction

Context-awareness is important for pervasive computing environments to adapt computational entities to changing situations such as the users needs and technical capabilities. The fundament for context-awareness is a formal context model which is needed to represent the context in a way computers can interpret it. In distributed environments it is especially important that this context information can also be shared between different computational entities which enables interoperability. Furthermore it is necessary to reason on the context knowledge e.g. to solve inconsistencies of sensor data or to deduce high-level situational context information from low-level sensor data. This paper gives an introduction on how ontologies can be used to solve these requirements to context modeling and reasoning.

Chapter II explains the terms context modeling and reasoning. Chapter III gives an introduction to ontologies in computer science and chapter IV presents the ontology language OWL. Chapters V to VII explicate three different approaches to perform context modeling and reasoning using ontologies: CONON, COBRA-ONT and SOUPA. Chapter VIII provides a short conclusion.

II. Context Modeling and Reasoning

A. Context Modeling

Context modeling is the specification of all entities and relations between these entities which are needed to describe the context as a whole, e.g. information on location, time, the user and its current or planned activity, and computational entities.

A special problem for context modelling in distributed, heterogeneous environments is the use of proprietary representation schemes which hinder the interoperability of the different computational entities. The use of common context ontologies can solve this problem.

B. Context Reasoning

Context reasoning means to automatically deduce further, previously implicit facts from explicitly given context information.

One use case is to calculate high-level context information from low-level sensor data: e.g. when the motion detector sensor in the kitchen reports movements and the oven sensor reports that the oven is switched on, then high-level context information on the user activity can be deduced: the user is cooking.

Another use case is to check and solve inconsistencies in sensor data: e.g. when sensors report that the user PDA is located in the office while his mobile is at home, then it can not surely be deduced where the user himself is located. Further information may help to solve this inconsistency: When the key card of the user is used to open the office front door, then it can surely be asserted that the user himself is at the office (and even further, that he can not use his mobile, so all incoming calls should be forwarded to the office telephone).

How ontologies can support context reasoning will be shown later in this paper.

III. Ontology

The term “ontology” originates from philosophy and refers to the discipline that deals with existence and the things that exist. In computer science, things that “exist” are those which can be represented by data.

Different definitions for ontologies in computer science can be found in the literature [6]. As an example the translation of the definition by J. Voß [7] is given here: “An ontology is a formally defined system of concepts and relations between these concepts. Ontologies contain – at least implicitly – rules.” Three issues can be noted when putting this definition of ontologies in relation to context modeling and reasoning:

- A context model is also a system of concepts (entities) and relations, which makes an ontology a possible

mean for context modelling.
- An ontology is “formally defined”, which is a precondition for a computer to interpret it, e.g. for reasoning purposes.
- Rules can be used to implement context reasoning. This will be shown in detail later in this paper.

According to M. Gruninger and J. Lee [8] there are three main areas of applications for ontologies:
- Communication and knowledge sharing: an ontology serves as a common vocabulary of different agents (computational entities and humans).
- Logic Inferencing / reasoning: An ontology can be used to deduce implict knowledge from explict knowledge by applying rules.
- Knowledge reuse: Common ontologies (e.g. on time and spatial concepts) can be reused when building domain specific ontologies.

Typical elements of ontologies are:
- Concepts and its attributes
- Taxonomies to categorize concepts by generalization and specification
- Relations between concepts
- Axioms to define statements which are always true. They are used to prove the consistency of the knowledge modelled by an ontology and to deduce further facts (reasoning).
- Individuals (or facts) are instances of concepts and its relations.

There are different languages which are used to define ontologies, e.g. Ontolingua [9], LOOM [10] and OWL Web Ontology Language [4, 5]. OWL is introduced in detail in the next section.

IV. OWL Web Ontology Language

A. Introduction

The OWL Web Ontology Language is a specification by the World Wide Web Consortium (W3C) and serves as a fundamental component of the Semantic Web initiative [11]. OWL is based upon the Extensible Markup Language (XML), XML Schema [18], the Resource Description Framework (RDF) and RDF Schema (RDF-S) [19]. It is divided into three increasingly expressive sub languages OWL-Lite, OWL-DL and OWL-Full. OWL-DL is most often used because it provides maximum expressiveness (in opposite to OWL-Lite) without losing computational completeness and decidability (in opposite to OWL-Full).

Different tools have been developed to support the work with OWL, e.g.:
- Protégé [12] is a graphical editor.
- Jena [13] is a Java API to access OWL ontologies.
- Racer [14] is an inference engine which can also be integrated into Jena and Protégé.

B. OWL Language Elements

In this subsection the most important OWL language elements will be introduced by example. A full specification can be found in [4].

`Class` is used to define ontology concepts. The following example defines the concepts “Person” and “University”:

```
<owl:Class rdf:ID=“Person“/>
<owl:Class rdf:ID=“University“/>
```

`subClassOf` allows to define specialized concepts and can therefore be used to build taxonomies: The following example defines the concept “Student” as a special person:

```
<owl:Class rdf:ID=“Student“/>
  <rdfs:subClassOf rdf:resource“#Person“/>
</owl:Class rdf:ID>
```

`DataTypeProperty` is used to define attributes of concepts. The range of a `DataTypeProperty` is defined by XML Schema Types. The following example defines that the concept “Person” has an attribute “Name” which must be a character string:

```
<owl:DatatypeProperty rdf:ID=”name”>
  <rdfs:domain rdf:resource=”#Person”/>
  <rdfs:range rdf:resource =
“http://www.w3.org/2001/XMLSchema#string“/>
</owl:DatatypeProperty>
```

`ObjectProperty` defines relations between concepts. They can be marked as transitive, symmetrical or functional. Two relations can be marked as inverse to each other. Furthermore relations can be specialized by using `subPropertyOf` in analogy to `subClassOf` for concepts. The following example defines the relation “studies_at” with the concept “Student” as domain and the concept “University” as range. It is inverse to an other relation “has_student”:

```
<owl:ObjectProperty rdf:ID=”studies_at”>
  <rdfs:domain rdf:resource=”#Student”/>
  <rdfs:range rdf:resource=”#University”/>
  <owl:inverseOf rdf:resource =
“has_student“/>
</owl:ObjectProperty>
```

Both relations and attributes can be refined by cardinality restrictions and further claims on the values.

It is also possible to define facts upon the previously defined concepts, attributes and relations by instancing them. These instances are called individuals. The following example states that “TUB” is a university and “FeruzanAy” is a student named “Feruzan Ay” who studies at the “TUB”.

```
<University rdf:ID="TUB"/>
<Student rdf:ID="FeruzanAy">
  <name rdf:resource="Feruzan Ay"/>
  <studies_at rdf:resource="TUB"/>
</Student>
```

Furthermore it is possible to define specialized concepts depending on the value of relations and attributes. For instance the concept „TUBStudent“ can be defined as a special „Student“ whose relation „studies_at“ points to the individual „TUB“:

```
<owl:Class rdf:ID="TUBStudent">
  <owl:intersectionOf
rdf:parseType="Collection">
    <owl:Class rdf:about="#Student" />
    <owl:Restriction>
      <owl:onProperty
rdf:resource="#studies_at"/>
      <owl:hasValue rdf:resource="#TUB" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Some advanced language elements to define concepts which can not be described in detail in this paper are `disjointWith`, `intersectionOf` and `unionOf`.

OWL also provides means to version ontologies and to mark concepts, attributes, relations and individuals as identical to elements from other ontologies. This supports the interoperability of systems which are using different ontologies.

C. Reasoning with OWL

According to the semantics of the different OWL language elements further (afore implicit) knowledge can be deduced from explicit OWL facts. According to `subClassOf`, `subPropertyOf`, `intersectionOf`, `unionOf` and `disjointWith` etc. it can be calculated which individuals belong to a concept or vice versa which concepts match for a given object (compare Table 1).

TABLE 1 – PART OF OWL REASONING RULES [3]

Transitive-Property	(?P rdf:type owl:TransitiveProperty) $\wedge$ (?A ?P ?B) $\wedge$ (?B ?P ?C) $\Rightarrow$ (?A ?P ?C)
subClassOf	(?a rdfs:subClassOf ?b) $\wedge$ (?b rdfs:subClassOf ?c) $\Rightarrow$ (?a rdfs:subClassOf ?c)
subProperty-Of	(?a rdfs:subPropertyOf ?b) $\wedge$ (?b rdfs:subPropertyOf ?c) $\Rightarrow$ (?a rdfs:subPropertyOf ?c)
disjointWith	(?C owl:disjointWith ?D) $\wedge$ (?X rdf:type ?C) $\wedge$ (?Y rdf:type ?D) $\Rightarrow$ (?X owl:differentFrom ?Y)
inverseOf	(?P owl:inverseOf ?Q) $\wedge$ (?X ?P ?Y) $\Rightarrow$ (?Y ?Q ?X)

For instance if it is explicitly stated that Feruzan Ay is a student (as in the examples from the previous subsection), then the implicit fact that Feruzan Ay is also a Person can be deduced, because a student is a special person (`subClassOf`). Furthermore the values of attributes and relations for an individual can be calculated if they are not explicitly given - according to `subPropertyOf`, `inverseOf` and markers for transitive or symmetrical properties. For instance from the example fact in the previous subsection that Feruzan Ay studies at the TU Berlin it can deduced that the TU Berlin has the student Feruzan Ay (`inverseOf`).

Further examples for reasoning using OWL will be given during the presentation of CONON, COBRA-ONT and SOUPA in the next sections.

V. CONON: „ONTOLOGY BASED CONTEXT MODELING AND REASONING USING OWL“

A. Introduction

“Ontology Based Context Modeling and Reasoning using OWL” introduces an OWL ontology named CONON, which stands for “Context Ontology”. CONON is supposed to be used in pervasive computing environments to enable context modeling and logic-based context reasoning. Furthermore it supports interoperability of different devices as a common vocabulary.

CONON defines generic concepts regarding context and provides extensibility for adding domain specific concepts. Logic reasoning is used in order to perform consistency checks and to calculate high-level context knowledge from explicitly given low-level context information.

The authors also present the results of a performance study which was arranged to evaluate the feasibility of logic based context reasoning in pervasive computing environments. This is especially important because in these environments computational resources such as processing power and memory are often limited.

B. Structure of CONON

Figure 1 shows the overall structure of CONON. It consists of an upper ontology which is extended by several domain specific ontologies for intelligent environments such as “home”, “office” or “vehicle”. CONON is implemented by using OWL.

The upper ontology holds general concepts which are common to the most sub domains and can therefore be flexibly extended. The most general concepts (all extending the common super concept `ContextEntity`) are `CompEntity` (computational entity), `Location`, `Person` and `Activity`. Each has a sub set of still abstract sub concepts, e.g. `Location` is distinguished between `IndoorSpace` and `OutdoorSpace`. `CompEntity` for instance can be a `Service`, `Application`, `Device`, `Network` or `Agent`. Each concept is associated with attributes and relations to other concepts (using OWL `DatatypeProperty` and OWL

`ObjectProperty`). The hierarchical structure of sub and super concepts is obtained by using OWLs `subClassOf`.

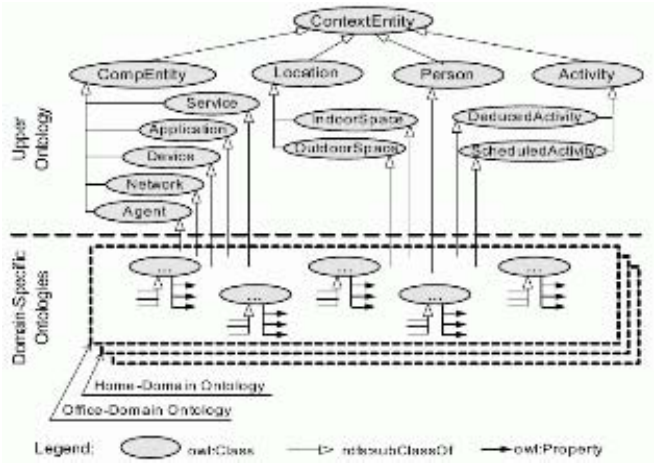

Fig. 1: CONON context ontology (extract) [3]

Figure 2 shows an excerpt from the "smart home" application domain. Generic concepts from the upper ontology (gray) are specialized by concepts which are specific to a smart home (white) each having specific attributes and relations. E.g. the `Room` is an `IndoorSpace` specific to a smart home domain because it would not be needed in a "vehicle" application domain. "Smart home" devices are `TV` and `DVDPlayer` which would probably not be needed in an "Office" application domain, where `Beamer` and `DesktopComputer` are more important.

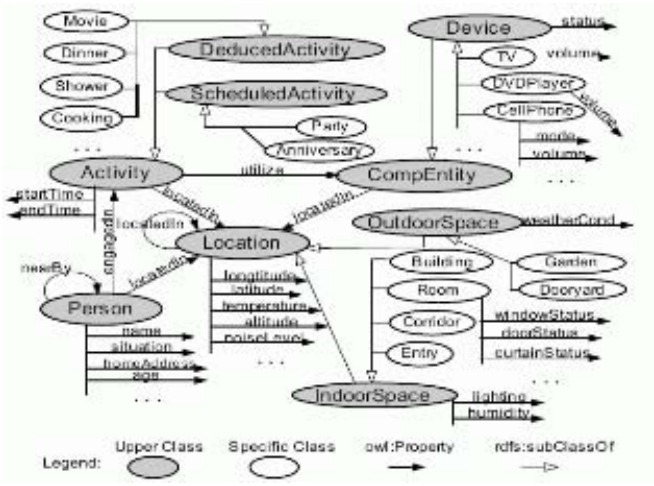

Fig. 2: CONON home domain (extract) [3]

C. Reasoning with CONON

There are two distinct ways to perform reasoning with CONON:

- Ontology reasoning by description-logic rules which are integrated in the OWL semantics, e.g. for transitive and inverse relations (compare Table 1).
- User-defined reasoning by first-order-logic rules which are stated "outside" of OWL.

For both ways a reasoner (an inference engine) basing upon the Jena toolkit [13] was implemented.

Table 2 shows an example for CONON reasoning by OWL rules: The input for the reasoner consists of the OWL rules, which are given by the semantics of the OWL language elements (which are `TransitiveProperty` and `inverseOf` in this example) and a set of explicit facts also encoded by OWL and the CONON ontology concepts: Mr. Wang is in the bedroom and the bedroom is in the house (`locatedIn`). According to CONON the relationship `locatedIn` is transitive and inverse to the relationship `contains`. From this explicit knowledge and the OWL rules implicit knowledge can be deduced: Mr. Wang is at home. His Home contains the bedroom and Mr. Wang and the bedroom contains Mr. Wang.

TABLE 2 – EXAMPLE FOR CONON REASONING BY OWL RULES [3]

INPUT	DL Reasoning Rules	(?P rdf:type owl:TransitiveProperty) $\wedge$ (?A ?P ?B) $\wedge$ (?B ?P ?C) $\Rightarrow$ (?A ?P ?C) (?P owl:inverseOf ?Q) $\wedge$ (?X ?P ?Y) $\Rightarrow$ (?Y ?Q ?X)
	Explicit Context	<owl:ObjectProperty rdf:ID="locatedIn"> <rdf:type="owlTransitiveProperty"/> <owl:inverseOf rdf:resource="#contains"/> </owl:ObjectProperty> <Person rdf:ID="Wang"> <locatedIn rdf:resource="#Bedroom"/> </Person > < Room rdf:ID="Bedroom"> < locatedIn rdf:resource="#Home"/> </ Room>
OUTPUT	Implicit Context	<Person rdf:ID="Wang"> <locatedIn rdf:resource="#Home"/> </Person > <Building rdf:ID="Home"> < contains rdf:resource="#Bedroom"/> < contains rdf:resource="#Wang"/> </Building> <Room rdf:ID="Bedroom"> < contains rdf:resource="#Wang"/> </Room>

Table 3 shows examples for the reasoning with the help of user-defined rules outside of OWL. The example illustrates that by applying first-order-logic rules a more flexible way of reasoning can be performed in order to calculate high-level context, such as on the information of the activity of the user. E.g. if the user is located in the bedroom, the light level of the bedroom is low and the draperies are closed then it can be deduced that the user is actually sleeping. Note that although the rules are not stated using OWL the rules are supposed to work on the knowledge modeled by the CONON OWL ontology. The user-defined reasoning should be performed after the ontology reasoning because this allows to include the facts received from the ontology reasoning in the input for the user-defined reasoning.

TABLE 3 – EXAMPLE FOR CONON REASONING BY USER-DEFINED RULES [3]

Situation	Reasoning Rules
Sleeping	(?u locatedIn Bedroom) ∧ (Bedroom lightLevel LOW) ∧ (Bedroom drapeStatus CLOSED) ⇒ (?u situation SLEEPING)
Shower-ing	(?u locatedIn Bathroom) ∧ (WaterHeater locatedIn Bathroom) ∧ (Bathroom doorStatus CLOSED) ∧ (WaterHeater status ON) ⇒ (?u situation SHOWERING)
Cooking	(?u locatedIn Kitchen) ∧ (ElectricOven locatedIn Kitchen) ∧ (ElectricOven status ON) ⇒ (?u situation COOKING)
Watching-TV	(?u locatedIn LivingRoom) ∧ (TVSet locatedIn LivingRoom) ∧ (TVSet status ON) ⇒ (?u situation WATCHINGTV)
Having-Dinner	(?u locatedIn DiningRoom) ∧ (?v locatedIn DiningRoom) ∧ (?u owl:differentFrom ?v) ⇒ (?u situation HAVINGDINNER)

D. CONON Reasoning Performance

Figure 3 displays the results of the reasoning performance study for both user-defined reasoning and ontology reasoning. The run time for a growing number of RDF triples which is a mean to express the ontology knowledge size was measured on different CPUs.

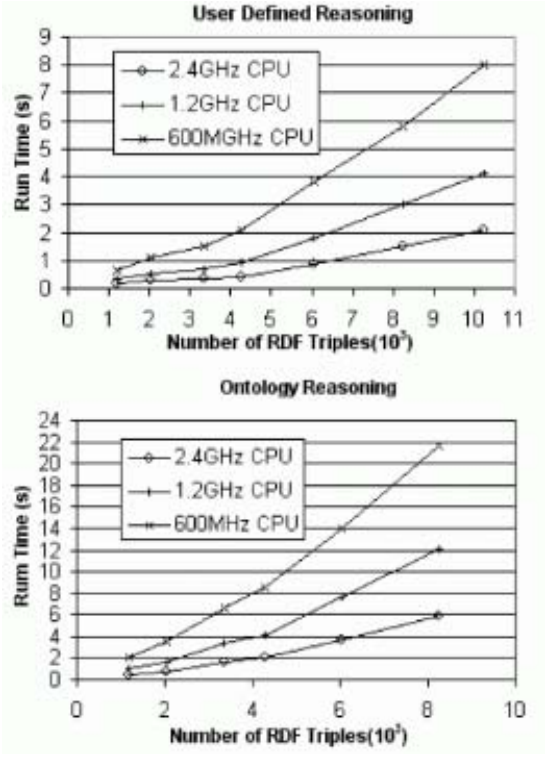

Fig. 3: CONON reasoning performance [3]

The user-defined reasoning is faster compared to the ontology reasoning. The performance of both is depending on the ontology knowledge size and CPU power: the reasoning becomes slower for bigger ontology knowledge (the consumed time grows faster than linear) and of course the reasoning is faster on faster CPUs.

The authors conclude that context reasoning is (although time-consuming) still feasible for non-time-critical applications, e.g. it is not important whether the mobile phone needs some seconds to switch to mute mode when the user goes to bed. For time-critical applications such as navigation systems or security systems the data size and rule complexity must be reduced. The partitioning into several smaller domain specific ontologies helps to do so, because only the concepts and their corresponding rules that are actually necessary will be calculated. Concerning the overall architecture the authors recommend to decouple the context reasoning from the context usage: the reasoning is performed by a strong server while small devices such as mobile phones receive the pre-calculated high-level context from the server for direct use.

VI. COBRA-ONT: „AN ONTOLOGY FOR CONTEXT-AWARE PERVASIVE COMPUTING ENVIRONMENTS“

A. Introduction

The paper “An Ontology for Context-Aware Pervasive Computing Environments” [1] presents COBRA-ONT which is a collection of OWL ontologies for context-aware systems. COBRA-ONT (like CONON) is supposed be used as a common vocabulary in order to overcome the obstacle of proprietary context information models which hinder the interoperability of different devices. Furthermore the semantics of OWL are used for context reasoning.

B. CoBrA

COBRA-ONT is central part of CoBrA, a “broker-centric agent architecture in smart spaces” where it supports context reasoning and interoperability as mentioned above. The center of this architecture is context broker agent which is a server that runs on a resource-rich stationary computer. It receives and manages context knowledge for a set of agents and devices in its vicinity which is the “smart space”. It can perform reasoning to deduce high-level context information from low-level sensor data, detect inconsistencies in the context information and observes users’ security policies when passing user information to other agents. Agents and devices can contact the context broker and exchange information by the FIPA Agent Communication Language.

In this architecture the context broker solves two important problems of pervasive computing environments:

- Small devices with minor computational resources such as smart phones are relieved from storing context information and performing context reasoning, because that is done by the context broker. (This is the same

approach as proposed by the authors of "Ontology Based Context Modeling and Reasoning using OWL".)

- The broker centrally observes security policies.

The problem of the broker agent being a bottle-neck can be solved by a so-called broker-federation, which is a network of context broker agents.

C. COBRA-ONT Use-Cases

In the following two use-cases are given to illustrate in which scenarios CoBrA and COBRA-ONT can be used:

- The sensor agent detects a bluetooth mobile in room 210 and sends this information to the broker using COBRA-ONT concepts. The broker deduces that the owner of this mobile phone is also located in room 210.
- After a lecture given by a professor a student takes pictures of the speaker. Before the camera sends the pictures to the photo agent at home it checks via the broker whether it is allowed to publish pictures of the event. The photo agent acquires location and event information from the broker to embed it in the pictures meta-information before putting it in the according photo albums.

D. COBRA-ONT Conceptualization

This subsection gives and introduction to COBRA-ONTs conceptualization. COBRA-ONT consists of four sub ontologies: Place, Agent, Agent's Location and Agent's Activity whose concepts and concepts' attributes and relations are given in tables 4 – 5. The most important parts of the ontologies will be explained in the following. A more detailed explanation can be found in [1].

TABLE 4 – COBRA-ONT CONCEPTS [1]

"Place" Related	Agents' Location Context
Place AtomicPlace CompoundPlace Campus Building AtomicPlaceInBuilding AtomicPlaceNotInBuilding Room Hallway Stairway OtherPlaceInBuilding Restroom Gender LadiesRoom MensRoom ParkingLot	ThingInBuilding SoftwareAgentInBuilding PersonInBuilding ThingNotInBuilding SoftwareAgentNotInBuilding PersonNotInBuilding
"Agent" Related	**Agent's Activity Context**
Agent Person SoftwareAgent Role SpeakerRole AudienceRole IntentionalAction ActionFoundInPresntation	PresentationSchedule Event EventHappeningNow PresentationHappeningNow RoomHasPresentationHappeningNow ParticipantOfPresentation-HappeningNow SpeakerOfPresentationHappeningNow AudienceOfPresentationHappeningNow PersonFillsRoleInPresentation PersonFillsSpeakerRole PersonFillsAudienceRole

TABLE 5 – COBRA-ONT ATTRIBUTES AND RELATIONS [1]

"Place" Related	Agent's Location Context
latitude longitude hasPrettyName isSpatiallySubsumedBy spatiallySubsumes accessRestricted-ToGender lotNumber	locatedIn locatedInAtomicPlace locatedInRoom locatedInRestroom locatedInParkingLot locatedinCompoundPlace locatedInBuilding locatedInCampus
"Agent" Related	**Agent's Activity Context**
hasContactInformation hasFullName hasEmail hasHomePage hasAgentAddress fillsRole isFilledBy intendsToPerform desiresSomeone-ToAchieve	participatesIn startTime endTime Location hasEvent hasEventHappeningNow invitedSpeaker expectedAudience presentationTitle presentationAbstract presentation eventDescription eventSchedule

The central concept of the Place ontology is the concept `Place` with attributes such as `latitude` and `longitude` to describe its location. It is the union of the concepts `AtomicPlace` and `CompoundPlace`. The `CompoundPlace` has a relation `spatiallySubsumes` whose domain is `Place`. By this relation places can be spatially "nested". Its inverse relation is `spatiallySubsumes`. There are special `AtomicPlaces` such as `Room`, `Stairway` and `ParkingLot`. Special `CompoundPlaces` are `Campus` and `Building`.

The Agent ontologies' most important concepts are `Agent` and its direct, disjoint specializations `Person` and `SoftwareAgent` to represent human agents and software agents respectively. An agent has attributes such as a name (`hasFullName`) or an email address (`hasEmail`). Agents can be assigned with roles (`Role`) such as speaker (`SpeakerRole`) or audience (`AudienceRole`) during an event by `fillsRole`. A role is used to reason about the intention of an agent, because a role can be associated with `IntentionalAction` by the relation `intendsToPerform`. However, `IntentionalAction` is not specified by COBRA-ONT.

The Agent's Location ontology adds the `locatedIn` Relation to the `Agent` concept. `locatedIn` points to `Places`. The relation specializes to `locatedInAtomicPlace` and `locatedInCompoundPlace` for `AtomicPlace` and `CompoundPlace`, respectively. Two axioms can be stated here: An agent can not be at different `AtomicPlaces` at the same time. But an agent is at two different `CompoundPlaces` if and only if one of the compound places subsumes

(`spatiallySubsumes`) the other. These axioms allow to reason about the consistency of knowledge about the current location of an agent – as illustrated by the second use-case of context reasoning in chapter II.

For each specialization of `Place` there is also a specialization of `locatedIn` in the Agent's Location ontology, e.g. for `Building` there is the relation `locatedInBuilding`. Additionally agents can be categorized according to their location. For example `PersonInBuilding` describes all people who are in a building while `SoftwareAgentInBuilding` describes all software agents in a building. There are also concepts for the complements (e.g. `SoftwareAgentNotInBuilding` and `PersonNotInBuilding`). There are corresponding category concepts for all other specializations of `Place`.

The Agent's Activity ontology describes events (concept `Event`) which happen at places (relation `hasEvent`) and which are attended by agents (relation `participatesIn`). `Events` also have schedules, which is expressed by the relation `eventSchedule` whose range is `PresentationSchedule`. A schedule has attributes e.g. for start time, end time and title of the presentation. The relations `invitedSpeaker` and `expectedAudience` both with the range `Person` describe the speaker and audience of a presentation. The concept `PresentationEventHappeningNow` comprises all presentations which are currently (now) taking place according to the start and end time. Basing upon this concept further concepts can be defined which allow to reason on a persons current activity (e.g. `SpeakerOfPresentationHappeningNow` and `AudienceOfPresentationHappeningNow`) and also which Places are currently occupied because of a presentation (e.g. `RoomHasPresentationHappeningNow`).

E. COBRA-ONT Inference Engine

Basing upon the above explained COBRA-ONT OWL ontology the authors have implemented a prototype of an inference engine named F-OWL. It can perform reasoning on RDF and OWL facts. It is possible to integrate domain-specific rules to support consistency detecting and solving and also rules for interpreting sensing inputs. The authors do not provide further details on this.

F. Conclusion

The authors claim to have shown that OWL is suitable to build a common knowledge representation for context-aware systems. In the future COBRA-ONT will be revised in order to reuse or at least map to existing common ontologies (e.g. on time and space) which increases the interoperability with other ontologies. More rules will be integrated and a prototype called "EasyMeeting" for intelligent meeting rooms will be implemented.

Actually the paper "An Ontology for Context-Aware Pervasive Computing Environments" does not fully show how it is possible to implement the two presented use-cases by using COBRA-ONT. Furthermore COBRA-ONT seems incomplete at some places, e.g. there is a concept `IntentionalAction` for which reasoning is to be performed but which is actually not specified. At other places COBRA-ONT already seems too specific for a common knowledge representation for context-aware systems, i.e. the presentation concepts.

VII. SOUPA: „Standard Ontology for Ubiquitous and Pervasive Applications"

A. Introduction

SOUPA, the Standard Ontology for Ubiquitous and Pervasive Applications [2] was developed by the same authors as COBRA-ONT. SOUPA is more comprehensive than COBRA-ONT because it deals with more areas of pervasive computing. It also addresses COBRA-ONT's problems regarding ontology reuse: The SOUPA sub-ontologies map many of its concepts via owl:equivalentClass to concepts of existing common ontologies (such as FOAF [15], DAML-Time [16], OpenCyc Spatial Ontologies [17] etc.) to enable interoperability with other ontologies. There are two use-cases given for SOUPA: CoBrA and its "EasyMeeting" prototype for smart rooms and MoGATU which is a peer-to-peer data management framework for pervasive environments.

As illustrated by figure 4 SOUPA consists of two parts: SOUPA core and SOUPA extension. The SOUPA core holds generic ontologies which provide a common vocabulary for different pervasive computing environments. The SOUPA extension contains additional ontologies for domain specific vocabulary.

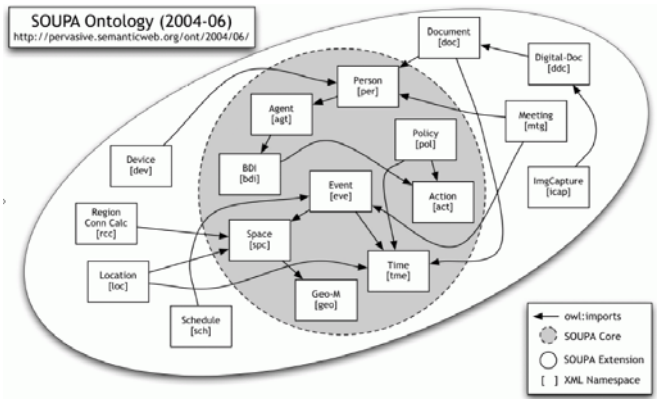

Fig. 4: SOUPA Ontology [1]

B. SOUPA sub-ontologies

The SOUPA Person ontology provides typical concepts for contact information and a person's profile. The central concept is Person which is marked as equivalent to the Person concept in FOAF. It has attributes and relations for describing a persons basic profile (e.g. name, date of birth), contact information (e.g. email, telephone) and social and professional

relations.

The SOUPA Policy and Action ontologies provide concepts for security and privacy issues. Central concepts are `Action` (e.g. to publish location information) and `Policy` which enables to allow or forbid actions.

The SOUPA Agent and BDI ontologies describe agents (`Agent`) which are either users or computational entities. Agent-oriented technologies use the idea of belief, desire and intention. Beliefs represent facts which are known to the agent (and which are not necessarily true). Desires are the goals of the agent. Intentions are the plans of the agent to reach his goals. Belief, desire and intention are represented by the concepts `Fact`, `Desire` and `Intention` of the SOUPA BDI ontology, respectively. A `Fact` is represented by an arbitrary RDF triple. `Desire` is unspecified by the BDI ontology. An `Intention` consists of a set of `Plan`s, which is a specialized `Action` from the SOUPA Action ontology. An Intention additionally has preconditions and effects, represented by the not further specified BDI ontology concepts `Precondition` and `Effect`.

The SOUPA Time ontology is developed on the basis of the DAML Time ontology and the Entry sub-ontology of Time [16]. It provides concepts to express time and temporal relations, which are used e.g. to describe temporal properties of events. The essential concepts are `TemporalThing`, which represents everything that is temporal, and its sub concepts `TimeInstant` and `TimeInterval`, which represent points and periods of time, respectively. For `TimeInstant` there are several relations to represent the order of two points of time, e.g. `sameTimeAs`, `before` and its inverse counterpart `after`. For `TimeIntervals` there relations such as `startsSoonerThan` and `startsLaterThan`.

The SOUPA Space and Geo Measurement ontologies built on the basis of OpenCyc and OpenGIS [17] provide concepts to represent geographical regions, spatial coordinates and relations and corresponding conversions. The central concept is `SpatialThing`, which is used for everything that has spatial properties. The relation `hasCoordinates` allows to provide `SpatialThing`s with exact coordinates (e.g. given by GPS). `GeographicalRegion`s can be spatially nested by the `spatiallySubsumed` relation which is the inverse of `spatiallySubsumedBy`. `GeopoliticalEntity` (e.g. USA) has the relation `controls` which points to `GeographicalRegion` (e.g. Alaska). The Geo Measurement Ontology provides typical geo-vocabulary, e.g. longitude, latitude, altitude etc.

The SOUPA Event ontology describes events which have both temporal and spatial properties. The `Event` concept describes activities and schedules and it is fused with `SpatialThing` and `TemporalThing` resulting in the concept `SpatialTemporalEvent`.

The SOUPA extension ontologies are built on the basis of the SOUPA Core to support special application scenarios. Not all of them can be described in this paper. The Meeting and Schedule ontologies for instance provide typical concepts in relation to meetings, schedules and their participants. The Document and Digital Document ontologies allow to describe meta information for documents such as size and creation date.

C. SOUPA Applications

To demonstrate that SOUPA is feasible for supporting pervasive computing applications the authors are prototyping two use-case scenarios. One is CoBrA which was already introduced earlier in chapter VI. The other is MoGATU which is explained in the following.

MoGATU is a framework to support peer to peer information exchange in pervasive computing environments. All devices of the framework are treated similarly as semi-autonomous participants which communicate using ad-hoc IEEE802.11 Bluetooth networks. Each device is an information manager, consumer and provider itself, which is a completely different approach than CoBrA where a central broker agent is needed. The devices manage the following information:

- information on other participants in vicinity: kind of device and the service or information provided
- information received from other participants
- user profile and BDI

All information is stored and communicated using the SOUPA ontology.

To illustrate the application of MoGATU the following use-case is given: Bob schedules a meeting with Jane in the shopping mall. Its palm automatically calculates the route to the mall and while going there it communicates with other cars to receive information on possible traffic jams – and recalculates the route accordingly. When arrived at the mall the palm knows there is still time left until the meeting and (from Bobs profile) Bob needs new shoes. The palm communicates with the agent of the local shoe dealer, negotiates a deal (20% off) for a pair of shoes of Bob's favorite brand and leads Bob to the shoe store.

VIII. CONCLUSION

It was shown that ontologies can be used to support ontology modeling and reasoning by applying the general application areas for ontologies (knowledge sharing, logic inferencing and knowledge reuse) to the domain of context in pervasive computing environments. Three different existing approaches basing upon OWL were introduced. SOUPA, which was developed using the experiences from COBRA-ONT, has the most comprehensive conceptualization and supports interoperability by mapping its concepts to other

common ontologies. CONON shows that context reasoning is a calculation intensive task which makes it necessary to reduce the needed number of concepts. This is achieved by SOUPA and CONON by partitioning the conceptualization into different sub ontologies. All three approaches make use of the semantics of the ontology language to perform logic reasoning.

However, the real benefit of using ontology for context information in pervasive computing environments, which lies in the interoperability of different devices, will not become effective before there is a widely-accepted standard context ontology. The introduced approaches are a good starting point for the future work to establish such a standard.

REFERENCES

[1] H. Chen, and T. Finin: "An Ontology for Context Aware Pervasive Computing Environments"

[2] H. Chen, F. Perich, T. Finin, and A. Joshi: „SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications."

[3] X. H. Wang, T. Gu, D. Q. Zhang, and H. K. Pung: „Ontology Based Context Modeling and Reasoning using OWL"

[4] M. K. Smith, C. Welthy, and D. L. McGuinness: "OWL Web Ontology Language Guide", available online: http://www.w3.org/TR/owl-guide

[5] S. Bechhofer, F. v. Harmelen, J. Hendler, I. Horrocks, et al.: „OWL Web Ontology Language Reference", available online: http://www.w3.org/TR/owl-ref

[6] T. Wahl: "Konzeption und Realisierung einer Ontologie zur Modellierung und Ableitung von Geschäftsprozessen", TU Berlin, 2005

[7] J. Voß: „Begriffssysteme – Ein Vergleich verschiedener Arten von Begriffssystemen und Entwurf des integrierenden Datenmodells", HU Berlin, 2004

[8] M. Gruninger, and J. Lee: „Ontology – Applications and Design", in Communications of the ACM, 2002

[9] T. R. Gruber: "Ontolingua: A Mechanism to Support Portable Ontologies", Stanford University, 1992

[10] R. M. MacGregor: "Inside the LOOM description classifier", in ACM SIGART Bulletin, 1991

[11] T. Berners-Lee, J. Hendler, et al.: "The Semantic Web", in Scientific American, 2001

[12] Stanford Medical Informatics: "Protégé", http://protege.stanford.edu

[13] Hewlett-Packard Development Company: "Jena – A semantic web framework for Java.", http://jena.sourceforge.net

[14] V. Haarslev and R. Möller: "RACER User's Guide and Reference Manual", http://www.sts.tu-harburg.de/~r.f.moeller/racer/racer-manual-1-7-7.pdf, 2005

[15] D. Brickley and L. Miller: FOAF vocabulary specification, in RDFWeb Namespace Document, RDFWeb, xmlns.com, 2003.

[16] F. Pan and J. R. Hobbs: "Time in OWL-S", in Proceedings of AAAI-04 Spring Symposium on SemanticWeb Services, Stanford University, California, 2004.

[17] D. B. Lenat and R. V. Guha: "Building Large Knowledge-Based Systems": Representation and Inference in the CycProject. Addison-Wesley, February 1990

[18] Bray, T., J. Paoli, et al.: Extensible Markup Language (XML) 1.0 (Third Edition) - W3C Recommendation. http://www.w3.org/TR/2004/REC-xml-20040204, 2004.

[19] Manola, F. and E. Miller: RDF Primer - W3C Recommendation. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/, 2004.