Deliverable D3.1

Ontology and Language

Call: FP7-ICT-2013-10
Objective: ICT-2013.5.3 ICT for smart and personalised inclusion
Contractual delivery date: M6
Actual delivery date: 15.4.2015
Version: V2
Author: Dean Kramer, Middlesex University
Contributors: Juan Augusto, Middlesex University
Reviewers: Andreas Braun, Fraunhofer Lars Thomas Boye, Tellu AS
Dissemination level: Public
Number of pages: 27
## Contents

Executive summary .................................................................................................................. 3

1. Introduction ............................................................................................................................... 4

2. Project Strategy ......................................................................................................................... 4

3. Ontology ................................................................................................................................... 5

   3.1 Literature Review ................................................................................................................... 6

   3.1.1 Summary ........................................................................................................................ 10

   3.2 POSEIDON Ontology .......................................................................................................... 10

       3.2.1 SOUPA ......................................................................................................................... 10

       3.2.2 Person Ontology .......................................................................................................... 12

       3.2.3 Location Ontology ...................................................................................................... 13

       3.2.4 Development Tools ..................................................................................................... 14

4. Language .................................................................................................................................. 15

   4.1 Literature Review ................................................................................................................ 15

   4.1.1 Summary ........................................................................................................................ 20

   4.2 POSEIDON Language ......................................................................................................... 21

5. Conclusions .............................................................................................................................. 22

6. References ............................................................................................................................... 22
Executive summary

The purpose of this deliverable is to document two context-awareness aspects of the POSEIDON project: the ontology, and language.

In the report introduction, we give a brief introduction to the document, and define what context is. We use the definition of context as proposed by the early work on context-awareness. We introduce the project strategy for the ontology and language which takes an iterative, and recursive process. This process is self-informing through each prototype and pilot.

Ontologies are used in computer science as an explicit specialisation of a conceptualisation. In this report, we introduce ontologies, and describe different ontologies existing in literature. This is followed by the description of the current POSEIDON ontology, which extends the SOUPA ontology due to it being comprehensive. This extension includes more user properties and relationships between primary and secondary users. We also extend the spatial ontology to include places of interest where the user might be located.

Context models/definition languages can be a number of types including: key-value, markup scheme, graphical, object-oriented, logic based, and ontology based. Different approaches existing in literature were described and discussed, with them summarised in a table according to a number of metrics. Due to the lack of temporal handling in most context models/languages, which is often required in learning and reasoning, we chose to use the C-SPARQL language for the POSEIDON project. This language is an extension to the SPARQL query language to include temporal handling, and the ability to reason over dynamic ontological knowledge from RDF streams.

Using C-SPARL, we can integrate more easily both our static ontology knowledge as defined in the ontology section, and dynamic context data. This language will be extended over the course of the project to handle more features that are needed for our project.
1. Introduction

In this document, we will present the context ontology and definition language used in the POSEIDON project. Both of these elements form important roles in context-aware systems.

When considering context-aware systems, it is important to have a clear understanding of what a context is. For the duration of the POSEIDON project, we use the context definition defined by Dey (2001):

*Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves.*

While context can be many different aspects, in the POSEIDON project, we currently consider the following types of context: user (preferences, needs, activity), device (connectivity, battery level), and environmental (location). These different categories are driven either by the system, or by the user. As the project progresses, we anticipate that POSEIDON will consider a larger array of context types.

In the course of this project, both the ontology and context definition language will be used for modelling and defining the different contexts in the context-aware system. Each of these observable or inferable contexts will then drive the adaptation of the system to suit the users throughout their day.

The remainder of this document is broken down into the following sections: Section 2 introduces the project strategy taken for the development of the ontology and language. In Section 3, we introduce ontologies, describe ontologies currently known in literature, and present the current POSEIDON ontology. Then, in Section 4, we introduce different context modelling and language approaches currently in literature, and present the current POSEIDON context definition language. Finally, Section 5 concludes this deliverable.

2. Project Strategy

The context ontology and language play an important role in the definition of POSEIDON context-aware services and systems. Therefore, it is important to establish a strategy outlining how this deliverable is developed within the project.

In the POSEIDON project, the learning and reasoning module (LRM) in deliverable D3.2 is set to be developed and delivered incrementally. This module is driven by the ontology and the contexts that are defined in the language. When developing the ontology and language, we should ensure that the different entities proposed are usable in the system and the project. By developing a highly complex ontology straight away, it is possible it could be underutilised, or prove unsuitable during the course of the development of the LRM. This can also be said when developing a context language. Instead, we believe the ontology and language should be developed incrementally with the LRM.

As depicted in Figure 1, the ontology and system inform each other iteratively, as it provides the knowledge base which the LRM can use to provide context-awareness. By developing the ontology incrementally, the results of the system prototype can therefore inform if any refinements to the ontology are necessary.
3. Ontology

In this section, we shall introduce and discuss the different elements of the context ontology used in POSEIDON. First, we reflect on what an ontology is, and examine existing ontologies in the state of the art.

The term ontology originates from a branch of philosophy known as metaphysics, which deals with the nature and the organisation of reality. In computer science, ontologies are used as an explicit specification of a conceptualisation in knowledge-based systems (Gruber, 1993), including the semantic web (Ding et al., 2007). Ontologies allow us to structure information, including concepts, and their relationships.

The development of ontology based context models can provide benefits including (Wang et al., 2004):

- **Knowledge Sharing**: A common set of concepts about context can be used during the interaction of different computational entities.
- **Logic Inference**: Using ontologies, existing logic reasoning mechanisms can be used by context-aware systems to infer higher level context information from low-level raw contextual data. These mechanisms can also check and solve inconsistent context information.
- **Knowledge Reuse**: Large-scale context ontologies can be formed of existing ontologies of different domains. This enables great reuse for the developer, not requiring starting from scratch.

A number of different formal languages for developing ontologies exist, including the Web Ontology Language (OWL).

As part of the OWL specification, there are three different variants of OWL providing different levels of expressiveness:

- **OWL-Lite** is the lightest and simplest of the variants. It provides support for class hierarchy, individual, property specification. There is also limited support for constraint modelling.
- **OWL-DL** combines all features of OWL-Lite and in addition includes the ability to add description logic. It provides maximum expressiveness, while retaining computational completeness.
• **OWL-Full** uses the same vocabulary as OWL-DL, but is based on different semantics to OWL-DL. OWL-Full is a union of both RDF\(^1\) and OWL-DL. Models expressed in OWL-DL however are not computational complete.

Because of the relations between the different OWL variants, the following rules are applicable:

1. Every legal OWL-Lite ontology is also a legal OWL-DL ontology.
2. Every legal OWL-DL ontology is also a legal OWL-Full ontology.
3. Every valid OWL-Lite conclusion is also a valid OWL-DL conclusion.
4. Every valid OWL-DL conclusion is also a valid OWL-Full conclusion.

OWL documents can contain a collection of different entities including:

- **Classes**: A Class is a collection of similar or related characteristics. A class can contain individuals using *Class Extensions*. Each class can have any number of different instances, and can be a *subclass* of another. Each subclass inherits characteristics from its parent *superclass*.

- **Axioms**: These can be used for specifying core knowledge that can be used for different constraints in the model.

- **Properties**: A property is a directed binary relation that specifies class characteristics. Properties can be categorised into two types including:
  - *Datatype properties*: These are relations between instances of classes and RDF literals or XML schema datatypes.
  - *Object properties*: These are relations between instances of two classes.

- **Individuals/Instances**: Each of these represents particular object, or instance of a particular class. These instances can belong to any number of classes.

### 3.1 Literature Review

Ontologies for general purpose user modelling include the General User model Ontology (GUMO) (Heckmann *et al.*, 2005). In this ontology different dimensions of the user can be modelled. These user dimensions include characteristics, emotional state, personality, and physiological state. The aim of this ontology is to allow context-aware systems to share a common vocabulary, simplifying the exchange of user model data.

Concepts from the GUMO ontology were extended in the User Navigation Ontology (UNO)(Kikiras *et al.*, 2006). These extended concepts include mental ability, mobility ability, sensory ability, spatial ability, demographics, and preferences. Unlike GUMO which only provides a core knowledge base, UNO is also used in inference procedures because of its use of restrictions, and conditions in the user classes.

\(^1\) Resource Description Framework (RDF) http://www.w3.org/RDF/
User ontologies for people with disabilities include those proposed in the Cloud4all EU project (Madrid et al., 2012). This project is designed to help developers make cloud services with context-aware adaptable accessibility features. A number of different user related concepts are included, including condition, activity, equipment, and characteristic.

Ontologies for ambient assisted living (AAL) systems supporting elderly people have been developed (Zografistou, 2012). A core ontology is proposed which is used to model general aspects of smart environments. This higher level ontology can be used for describing abstract concepts including location, environment, simple events, person, activity, computer entity (compEntity), and time. Second, a collection of ontologies focused on concepts related to AAL. This collection included ontologies for Person Profile, Health, and Time. The person profile ontology allows for the modelling of a person’s status, habits, impairments, contact profile, and preferences. Within the health ontology, concepts including disease, symptoms, treatments, and restrictions can be modelled. Each ontology is developed in OWL-DL.
Other health related ontologies include OntoHealth (Librelotto et al., 2010, 2011). This ontology was designed to include vocabularies applicable in a pervasive hospital environment. This ontology includes concepts for Patient, Bed, Tray, Nurse, MedicalHistory, Dosage, Physician, Exam, Medicine, and Treatment. Also included, the authors propose an architecture that uses the ontology to provide context-aware behaviour within the hospital.

Ontology based context models for intelligent environments have been proposed (Gu et al., 2004; Wang et al., 2004). This ontology primarily tackles the limitations of previous approaches including modelling classification, dependencies, and quality of context. Included also in the ontology is the ability to support automated context reasoning. Context reasoning is enabled by the use of first-order logic predicates, which can be used for context aggregation, and deduction.
Huq et al. (2007) propose a context ontology for smart meeting spaces, based on limitations in CONON. In this ontology, concepts including Environment, Service, and Platform are introduced. The Environment concept encapsulates context information regarding environmental conditions e.g. temperature. All services provided to participants in a meeting are identified in the Service concept. Lastly, the Platform concept holds information about different hardware and software.

Ontologies for Ambient Intelligence have been proposed, including GAIA (Ranganathan et al., 2003), BOnSAI (Stavropoulos et al., 2012), CoDAMoS (Preuveneers and Berbers, 2005), and OntoAMI (Santofimia et al., 2009). In GAIA, contexts including physical (location and time), environmental (weather, light and sound levels), informational (stock quotes, sports scores), personal (health, mood, schedule, activity), social (group activity, social relationships), application (email, websites visited), and system (network traffic, status of printers) contexts are included in the ontology. The CoDAMoS ontology defined four main concepts: user, platform, service, and environment. The BOnSAI ontology directly imports the CoDAMoS ontology, and has classes that can be categorised in context-related, service-related, hardware-related, and functionality-related.

Chen et al. (2004) proposed a general purpose context model used for pervasive applications, named SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. SOUPA is built using a collection of reference vocabularies including Friend of a Friend (FOAF) (Dumbill, 2002), DAML-Time\(^2\) and the Entry Sub-ontology of Time, OpenCyc, Regional Connection Calculus (RCC), COBRA-ONT (Chen et al., 2003), MoGATU Belief-Decide-Intention (BDI) ontology (Perich, 2004), and the Rei policy ontology. SOUPA is broken in to two distinct ontologies, SOUPA Core, and SOUPA Extension as shown in Figure 5. SOUPA Core is a set of universal ontology vocabularies which can be used in generic pervasive applications. In this set, different concepts including person, agent, belief-desire-intention, action, policy, time, space, and event can be expressed. SOUPA Extension ontologies on the other hand extend the core ontologies with vocabularies to support specific pervasive domains. This collection includes

---

\(^2\) https://www.cs.rochester.edu/~ferguson/daml/
meeting, document, image capture, and region connection calculus concepts. Each of the ontologies are expressed in OWL-DL.

![SOUPA Ontology (2004-06)](http://pervasive.semanticweb.org/ont/2004/06/)

3.1.1 Summary
Ontologies have been a popular choice for modelling knowledge in context-aware systems. These ontologies have largely been developed to suit different domains including healthcare, smart meeting places etc. Because of this, most ontologies proposed already are too specialised for our domain and provide minimal overlap. We therefore need an ontology that includes a sufficient number of concepts that can be extended for our needs. For these reasons, we have chosen to use the SOUPA ontology as our base ontology, which we can build on and extend.

3.2 POSEIDON Ontology
In this section, we present the ontology which we intend to use with the context-aware systems developed later in the project.

As we summarised in the previous section of this document, we intend to use the SOUPA ontology as a base for the POSEIDON ontology. We therefore first describe the SOUPA ontology in more detail.

3.2.1 SOUPA
The SOUPA ontology (Chen et al., 2004) is broken up into a number of different OWL documents. These different documents within the core ontology include:

- **Person**: This ontology allows for the definition of different contact information and profile for a given person. These OWL classes shares many features with the Person class in the FOAF ontology, and is therefore considered equivalent. User profile properties include name, gender, age, and birth date. For contact information properties including email address, mailing address, homepage, phone numbers, and instant messaging chat identifiers can be specified for a user.

- **Agent & BDI**: Within these ontologies, different vocabularies are included to allow the developer to model different computational entities. These agents can be defined based on their belief, intention, desire, and action.
• **Space:** Using this ontology, the developer can define relationships between different spatial entities, and their relationships with others.

• **Geo-M:** This ontology is complimentary to the space ontology, allowing for the definition of different properties of a given space. These defined vocabularies include altitude, distance, latitude, longitude, and surface area.

• **Time:** This ontology is intended for the expression of different time and temporal relations. Vocabularies of the DAML-time and the entry sub-ontology of time are adopted in the ontology. Time can be represented in the ontology as either time instant, and time interval.

• **Event:** Event activities can be modelled using this ontology, which is an extension of both the space and time ontology. These events can be described along with their occurrence, time, and location.

• **Policy & Action:** These ontologies are used to enable security and privacy within the pervasive systems, influenced by the Rei policy language (Kagal et al., 2003). The policy ontology allows the execution of actions to be restricted or guided by the user.

These core ontologies can be used as a solid foundation for most domains. However, in specific domains, there can be a need for additional ontologies. These ontologies make up the SOUPA extension ontology, and include the following documents:

• **Device:** Different information about computing devices can be defined in this ontology. It also links each device with specific users.

• **Location:** Describing sensed location context of a person or an object is handled in this ontology. Location descriptions can contain both temporal and spatial properties.

• **Meeting & Schedule:** This ontology defines vocabularies for scheduling meetings and events including participants, and location.

• **Image Capture:** Information relating to captured images are defined in this ontology. This includes information about where the photo was taken and the device it was taken with.

• **Document & Digital-Document:** This ontology can be used to describe meta-data regarding different documents. This includes creation date, author, file size, and file type.

• **Region Connection Calculus:** An ontology based on the Region Connection Calculus (Randell et al., 1992) to assist qualitative spatial reasoning by the definition of spatial relations between entities.

The following shows a partial ontology description of the person John Doe:
Using the SOUPA ontology, we extend two specific ontologies to include concepts that are more relevant to our domain, the Person ontology, and the Location ontology.

3.2.2 Person Ontology

The Person ontology is a basic ontology for modelling the different users in the system. In Figure 6, we illustrate the taxonomy of classes in the ontology. The oval shapes correspond to classes, and the grey square boxes correspond to class data properties.

Figure 6 POSEIDON Person Ontology
This ontology is made up of the following classes:

- **Person**: This class is the central class of the ontology. There are three groups of users that correspond to the original groups of users described in our other documentation. The first is the primary user, which is a person with Down’s Syndrome. The secondary users correspond to people that are directly supporting the user including family, and carers. Lastly, the tertiary user are users that are involved less frequently with the primary user including teachers, employers etc. Generally, each person has a first name, last name, date of birth, and gender. For the primary user, we also define the level of independence that individual has. This level will be an enumeration of a finite set of values gathered from the Down’s Syndrome organisations of Germany, Norway, and United Kingdom.

- **Contact Profile**: This class is used for storing different information related to how each person can be contacted, including phone numbers, address, and email addresses.

- **Preference**: This is used to model different preferences the user might have for the system. These preferences can be used to help shape the different adaptive behaviours of the system, and devices.

- **User Needs**: Different primary users can have different needs that need to be carried out. These needs play a role in ensuring certain events or conditions must be met by the user, system, or device.

- **Emotion**: This is used for modelling the different emotional states of the primary user. By monitoring emotional state, we know if emergency assistance might be required by the user.

Figure 7 illustrates how the different classes in the ontology relate to each assisted person in the system.

![Figure 7 POSEIDON Person ontology relationships](image)

**3.2.3 Location Ontology**

To describe the different locations the user might encounter, we use the Location Ontology. In Figure 8, we illustrate the taxonomy of the ontology. The oval shapes correspond to classes, and the grey square boxes correspond to class data properties.
This ontology extends the SOUPA location ontology with the different types of locations the primary user may go to, or via. These include:

- **Educational-Related**: This class corresponds to locations where the primary user receives some kind of education. Examples include school, further education, and higher education institutions.
- **Leisure-Related**: This class corresponds to places of leisure and socialising. Examples include sports related venues, cinemas, parks etc.
- **Transport-Related**: This class corresponds to locations which related to a particular type of transportation. Examples include bus stops, train stations etc.
- **Work-Related**: This class corresponds to locations related to places of work for the primary user.

Each SpatialThing has a set of coordinates including longitude, latitude and altitude contained within an object property. Also extended is the ability to define a name and an address to each SpatialThing.

### 3.2.4 Development Tools

Each of the ontologies were developed using Protégé. Protégé\(^3\) is a free open-source platform and toolset for developing and editing ontologies. While in the past, a frame based ontology editor was available, this platform’s primary editor is OWL based. There are two different versions of this OWL based editor; a web based application, and a Java based desktop application.

---

\(^3\) [http://protege.stanford.edu/](http://protege.stanford.edu/)
4. Language
In this section, we shall introduce and discuss the different elements of the context language used in POSEIDON. First, we consider the state of the art in context modelling languages. This literature review is driven both by pre-existing literature surveys including Bettini et al. (2010), Bolchini et al. (2007), Strang and Linnhoff-Popien (2004) and newer publications.

4.1 Literature Review
Context Models and languages proposed prior have predominantly been one of the following (Strang and Linnhoff-Popien, 2004):

- **Key-Value Models.** These are the simplest form of context models, involving a name and context value pairs. These have frequently been used in service frameworks.
- **Markup Scheme Models.** Hierarchical data structures are formed using these models, consisting of markup tags, attributes, and content.
- **Graphical Models.** Context can be modelled graphically, using different graphical notations including UML, Object-Role Modelling (ORM), and other domain specific modelling languages.
- **Object-Oriented Models.** These models take advantage of various object-oriented concepts and techniques including encapsulation, and inheritance. Using this approach, different objects represent different types of context, and often encapsulate the context processing and representation.
- **Logic Based Models.** These models use facts, expressions, and rules to define formal models. Different facts can be inferred separately and then can be used in existing rules to derive higher level context knowledge.
- **Ontology Based Models.** As introduced earlier, ontologies can be used to describe taxonomies of concepts, including relationships. These models can allow for different context reasoning techniques, and inference rules.

While we introduced ontology based models earlier, context reasoning using these models has been based primarily around the use of query and inference languages including the SPARQL Protocol and RDF Query Language\(^4\). Using SPARQL, the developer can query different parts of the ontology model, and either just select the result, or alter the ontology model by updating previous instance values, or create new model concepts, and instances. One outstanding issue with these query languages are they used to query predominantly static data, and lack temporal qualities that are sometimes required in context reasoning. Della Valle et al. (2009) introduced the subfield of stream reasoning as solution, whereby different streams of highly changing ontological knowledge is reasoned over in real time. A latter stream reasoning language proposed includes C-SPARQL (Barbieri et al., 2009, 2010b, 2010a). This language builds on SPARQL to include temporal extensions to allow different real time streams of ontological knowledge to be reasoned over.

The Context modelling Language (CML) allows the developer to model context information and requirements graphically (Henricksen et al., 2002; Henricksen and Indulska, 2004, 2006). CML is formulated using concepts from Object-Role Modelling (ORM). The language includes constructs for describing information types, type classification, metadata for quality, and type dependencies. An example context model is depicted in Figure 9, designed for communication applications. Context aggregations are handled using a form of predicate logic using logical connectives and special forms of the universal and existential quantifiers.

\(^4\) http://www.w3.org/TR/rdf-sparql-query/
Modelling languages based on Universal Modeling Language (UML) have been proposed including ContextUML (Sheng and Benatallah, 2005) and the Music Context Model (Reichle et al., 2008). ContextUML is designed to support model-driven development of context-aware webservices. With ContextUML, a meta-model defining the abstract syntax of the language was presented. This meta-model is used within the tasks of modelling context, and context-awareness. The context model aspect describes the different context information that is used by the system, including the type and source of the context data. The context-awareness model aspect on the other hand describes the context-awareness mechanisms of the system. The ContextUML notation (concrete syntax) defines the semantics of the language using class stereotypes and Object Constraint Language (OCL) in UML. The MUSIC context model was proposed as part of a modelling framework for pervasive applications. In this model, three different abstraction layers are identified for handing the different phases in context management. These abstraction layers include the conceptual layer, the exchange layer, and the functional layer. The conceptual layer enables the definition of context entities using UML and OWL. UML based context models as depicted in Figure 10 are supported with a context meta-model. In the meta-model, context elements abstract context information which can contain a number of context values. Context elements and context values can have metadata associated with them, with the internal representation of context information being contained in representation.
Different markup based context models have been proposed including Comprehensive Structured Context Profiles (CSCP) by Held et al. (2002), with an example illustrated in Figure 11. This model is based on RDF, and was designed to meet the structuring shortcomings of the Composite Capability/Preference Profiles Language (CC/PP) (Klyne et al., 2001) regarding structuring. Context information is expressed using service profiles, which describes context information relevant to different sessions.

Other markup modelling languages include the VERITAS Virtual User Model, as proposed in the MyUI project (Peissner et al., 2011). This user model is based on UsiXML, a user interface extended markup language (UsiXML Consortium, 2007; Limbourg et al., 2004). In this model, different properties of the user can be modelled including general preferences, anthropometric, limb details, vision, hearing, speech, cognition, and behaviour are modelled.
Domain Specific Modelling Languages include PervML (Muñoz et al., 2004; Serral et al., 2008, 2010). PervML models are composed of six mandatory models; including a services model, structural model, interaction model, functional model, component structure model, and binding providers model, with a user model as an optional model. In the services model, all system provided services are modelled, including pre and post conditions using OCL. Instances of every service in the system are expressed in the structural model. The interaction model is then used for specifying the communication needed during a system event, using sequence diagrams. Description of the different devices used by the system is handled in the binding providers model. In the component structure model, devices and software systems are assigned to system components. Finally, the functional model is used for modelling service actions to be executed. These models are then transformed to Java code using model-to-code transformation language MOFScript.

Textual Domain Specific Languages have been proposed for defining contexts including MLContext (Hoyos et al., 2013). This defined context information is then used to generate software artefacts using model-to-text transformations. MLContext models can be transformed both into OWL-DL ontologies for Open Context Platform (OCP), and Java classes for the Java Context Aware Framework (JCAF). Context information in MLContext models can correspond to different types within the proposed taxonomy including physical, environmental, computational, personal, social, and tasks.

Context languages harnessing general purpose inference engines have been proposed including JCOOLS (Park et al., 2013). JCOOLS is a toolkit for generating context-aware applications using a context information schema, and rules developed in the DROOLS Guvnor Editor. The context information schema is composed of three smaller schemas including an object, relationship, and action. Context entities are described in the object schema, while the relationship schema describes the relationships between the different entities in the object schema. Finally, the action schema describes the actions that need to occur when there is a change in the context information.

A mix of key-value, object oriented, and markup approaches have been used within mobile context-aware systems, e.g. ContextEngine created by Kramer et al. (2011). In this approach the developer either creates their context model programmatically by calling different system interfaces of the management system, or by using XML. Each context in the model corresponds to an implemented component of the system, which can infer its state based on different context value definitions. These definitions link a customisable context state to the raw context data. Contexts can be composed to form higher level context information by aggregating context values from atomic contexts.

Formal logic based modelling languages for defining contexts exist, including the Calculus of Context-Aware Ambients (CCA) (Siewe et al., 2011, 2009), CONAWA (Kjærgaard and Bunde-pedersen, 2006), SCAFOS (Katsiri et al., 2010), and an algebra of contextualised ontologies (Cafezeiro et al., 2008). The CCA proposes a logical language for expressing context properties using context expressions. Context expressions can be composed to form complex expressions and formulas using first order operators. Samples of smaller context expressions are shown in Figure 11. The CONAWA calculus was inspired by the ambient calculus by Cardelli (1999), and extends it in a number of ways. First, it extends the syntax with constructs and capabilities allowing ambients navigate in complex context information. Second, it extends the semantics of different ambient capabilities to handle contexts, and context trees. The algebra of contextualised ontologies aims for a uniform representation of entities & context, and places an emphasis on the relationships between them. Different modular constructs are proposed to be applied to contextualised entities to combine entities and contexts coherently. These
constructs are broken down into three classes including: Entity Integration, Context Integration, and Combined Integration.

As presented earlier, ontologies can be used for modelling context information. Different rule engines can be used in conjunction with the ontology for context inferences. An example of this includes the use of the SWRL rule engine in Zografistou (2012). Every semantic web rule is built by the use of implications between an antecedent, known as the body, and a consequent, known as the head. The body and head can be made up of conjunctions of atomic elements.

In other domains including Software Product Line (SPL) Engineering, there have been context modelling languages proposed for Dynamic Software Product Lines (Fernandes et al., 2011; Acher et al., 2009). These notations are based on Feature Models (Kang et al., 1998), the de facto formalism used in SPLs for modelling product commonality and variability. In the feature model, the different contexts and their values can be modelled using existing tree relationships including mandatory, optional, or, and alternative (xor). This tree can also be supplemented with proportional formula for defining cross tree constraints, and rules for context aggregation and adaptation. As feature models can be encoded in propositional logic (Benavides et al., 2010), automatic reasoning can be carried out using SAT solvers.
While not designed for context modelling, other context-aware related languages exist for handling Event-Condition-Action (ECA) rules. These languages include programming languages (Alferes et al., 2006; Daniele, 2006), markup languages (Beer et al., 2007), and languages for the semantic web and RDF (Poulovassilis et al., 2006; Nakagawa et al., 2012). These rules typically are built of three components: events, conditions, and action. The event of a rule specifies the trigger that causes the rule to fire. The condition is a logical evaluation, which when satisfied, causes an action to be performed. Finally, the action is performed. These languages are not listed in the section summary, but are here for completeness.

4.1.1 Summary

In Table 1, we summarise the different languages reviewed in this section. Ontology based context models are not included in this table as they have been discussed in the previous section of this document.

<table>
<thead>
<tr>
<th>Language/Approach</th>
<th>Model Type</th>
<th>Tool Support Available</th>
<th>Open Source/Extendable</th>
<th>Platform Independence</th>
<th>Semantics-Expressiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERITAS Virtual User Model (Peissner et al., 2011)</td>
<td>Markup</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>C-SPARQL (Barbieri et al., 2009, 2010a)</td>
<td>Logical</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>RDF</td>
</tr>
<tr>
<td>CML (Henricksen and Indulska, 2006)</td>
<td>Graphical</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Predicate Logic</td>
</tr>
<tr>
<td>ContextUML (Sheng and Benatallah, 2005)</td>
<td>Graphical/OO</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Object Constraint language</td>
</tr>
<tr>
<td>Music CM (Reichle et al., 2008)</td>
<td>Graphical/OO</td>
<td>Yes</td>
<td>Yes</td>
<td>Partial</td>
<td>UML Relationships</td>
</tr>
<tr>
<td>CSCP (Held et al., 2002)</td>
<td>Markup</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>RDF</td>
</tr>
<tr>
<td>PervML (Serral et al., 2008)</td>
<td>Graphical/OO</td>
<td>Yes</td>
<td>Yes</td>
<td>Partial</td>
<td>UML Relationships</td>
</tr>
<tr>
<td>MLContext (Hoyos et al., 2013)</td>
<td>Logical</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Value Aggregation</td>
</tr>
<tr>
<td>CCA (Siewe et al., 2011)</td>
<td>Logical</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>First Order Logic</td>
</tr>
<tr>
<td>CONAWA (Kjærgaard and Bunde-pedersen, 2006)</td>
<td>Logical</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Lambda Calculus</td>
</tr>
<tr>
<td>Algebra of Contextualised Ontologies (Cafezeiro et al., 2008)</td>
<td>Logical</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Propositional Logic</td>
</tr>
<tr>
<td>JCOOLS (Park et al., 2013)</td>
<td>Markup/OO</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Drools Rules</td>
</tr>
</tbody>
</table>
We can see there are a variety of different approaches for modelling context. While in the table many approaches are noted to have tool support, we should clarify that this support is mostly indirect. Many of the languages/approaches are built on top of pre-existing modelling and logical languages, including UML and feature modelling. Because of this, these approaches can be carried out in existing UML and feature modelling editors. Also, with the open source column, because the meta-model for the UML based languages is normally presented in their papers, we can recreate and expand their languages. We also see that in two cases, partial platform independence is observed. These languages can produce artefacts for different platforms, but because of their models use Object Orientation, they support only languages that are Object-Oriented e.g. Java. All approaches have a method of aggregating contexts, found in the semantic-expressiveness column. Some of the approaches are more expressive than others. For example, value aggregations generally just deduce a higher level context if all of the containing context values are evaluated to be true (Kramer et al., 2011). This then forces the developer to only use context values that can only be evaluated to be true or false, and lacks the expressiveness found in other approaches e.g. first order logic.

### 4.2 POSEIDON Language

As discussed in the previous section, there are many different candidate languages that we could use or extend. As our work is involving ontological knowledge, temporal reasoning is an important component, we currently are using the C-SPARQL language (Barbieri et al., 2009, 2010b, 2010a) as our context rule development language.

CSPARQL is essentially a language extension of the SPARQL\(^5\) query language, and includes temporal quantifiers to define how often should the rules be fired, and how many streams should be used. RDF streams are sent in the form of ordered pairs of RDF triples and a timestamp, also known as a RDF quadruple. While sharing many similarities with SPARQL, C-SPARQL extends the `FROM` clause, as shown in Figure 14.

![Figure 14: C-SPARQL FROM Clause Syntax Extensions (Barbieri et al., 2010b)](http://www.w3.org/TR/rdf-sparql-query/)

When using SPARQL, query rules are applied to whole datasets, or specific graphs, but do not include streams. In C-SPARQL many different RDF Streams can be declared for different sensors, and data

\(^5\) [http://www.w3.org/TR/rdf-sparql-query/](http://www.w3.org/TR/rdf-sparql-query/)
sources, so one must define to which RDF Stream you wish to reason over. A window is a collection of data elements that are taken from the stream for a single query execution. This window can be either based on a certain period of time, or a particular number of triples. In addition to reasoning over RDF streams, C-SPARQL includes the ability to include static knowledge into a query using a static RDF file.

RDF Streams are not only an input into the C-SPARQL reasoner, but also can be an output, whereby streams can be created using data from other streams, and static RDF documents, as shown in Figure 15.

\[
\text{Registration} \rightarrow \text{‘REGISTER STREAM’ QueryName} \\
\text{[‘COMPUTED EVERY’ Number TimeUnit] ‘AS’ Query}
\]

*Figure 15: C-SPARQL Register Stream Syntax Extensions (Barbieri et al., 2010b)*

An example of C-SPARQL being used for one of our context rules includes the following:

```
REGISTER QUERY batteryContextIsLow
CONSTRUCT { ?s ?p "BATTERY_LOW" }
FROM STREAM http://poseidon-project.org/context-stream/battery
[RANGE 1s STEP 5s]
WHERE { ?s ?p ?o 
FILTER (?o < 20) }
```

In this rule, we are simply defining the rule for low battery. This is defined as having a value that is less than 20% of battery remaining.

5. Conclusions

In this document, we explored the state of the art for both ontologies, and context definition languages. Then the current POSEIDON ontology was described with the Person and Location ontology being the prime ontologies that have to be specified before creating the first prototype. Currently, we are using the C-SPARQL language for our context definitions, which enable us to reason over our static and rapidly changing ontological data.

6. References


