AN ONTOLOGY FOR MOBILE SITUATION AWARE SYSTEMS

Paul O'Brien

University of Queensland Business School
Email: p.obrien@business.uq.edu.au

ABSTRACT

This paper introduces a novel design artefact, namely a generic situation management ontology based on situation theory. This ontology contributes to the foundation knowledge base of mobile service delivery systems for future research and systems design. It demonstrates the applicability, and feasibility of using situation theory in the design of reactive information systems. The support within the ontology for context based filtering for situation detection also contributes to the efficiency of implementation and operation of situation driven reactive information systems.

Highly mobile people (HMPs) require flexible, reactive service delivery due to their regularly changing location and activities and the lack of a wired network connection. A mobile service delivery system should be able to detect relevant events that occur such as change of location, availability of new last-minute specials, sales opportunities and safety issues and then reactively take action in response to these events. This paper describes a generic situation management ontology that was developed in OWL using the ontology development tool, Protégé. The ontology is combined with domain specific classes in the travel domain to create a travel situation management ontology that can be used as the basis for a ubiquitous mobile travel service application. Using a typical independent traveler scenario, the travel situation management ontology is instantiated to demonstrate its effectiveness.

Keywords Situations, Situation Awareness, Situation Management, Context, Mobile Information Systems

INTRODUCTION

Today, personal computers are available that are powerful, portable, use little energy, do not need special environments and are cheap. The original computers were good for large scale information processing. They were not suited to the tasks that highly mobile people (HMPs) typically carry out as they go about their work or play. "They have not been aware of our needs or even of whether we were in the room with them" (MIT, 2004). The next generation of computing devices will be faster, cheaper, smaller, more powerful, and more portable than their predecessors. Wireless data technologies are maturing and improving in bandwidth, performance and reliability (WIMAX,

2006) so the next generation of portable computing devices will be networked via high speed, ubiquitous wireless technologies.

Problems with Current Automated Service Delivery Techniques

Current automated service delivery systems tend to be location-aware, context-aware within a particular room or single location, or temporally aware at a single location or within a single system. Their designs also tend to be restricted by consideration of narrow mobile bandwidth and limited handset capability and are generally designed for a specific task or a specific domain. Context dimensions of time alone or location alone or even a combination of time and location do not allow independent decisions to be made about situations that occur. Location, type of location, time, type of time, local bindings and user preferences are required as a minimum for effective decision-making for HMPs. Domain specific and application specific dimensions are not generalisable to other domains or applications.

Application and domain knowledge is essential for any working system, however, this information can be provided by creating a domain and application specific instance of a generic situation ontology that includes, of necessity, ontology elements that relate to context. By designing the architecture and system logic to be driven by a situation ontology, new domains and applications can be accommodated by creating a new *instance* of the situation ontology, thus eliminating the need to hard-code what is essentially object data into the application itself.

To automate the delivery of services the system must also be able to identify situations that require action (reactive situations) from the set of existing situations, and apply the appropriate action rules.

The focus of this research is therefore the design of a generic mobile situation management ontology that can be used as the basis for an architecture for a ubiquitous reactive service delivery system for HMPs.

Research Questions

The research problem that this work addresses is how to improve the delivery of services to HMPs continuously while they are travelling. Research questions arising from this problem include:

1. How can a highly mobile person's need for a mobile service (a reactive or proactive situation) be detected?
2. What mechanisms can be used to reduce the complexity of the detection and response to reactive situations?
3. How can the concepts related to situations be represented in an information system?
4. Is it possible to design a generic mobile situation management ontology that is domain independent?

Previous work by the author investigated the first two of these questions (O'Brien & Burmeister 2003; O'Brien & Colomb 2005). This paper focuses on the last two questions; the design of a generic ontology for mobile situation management that provides a base for the future development of a prototype of a ubiquitous mobile service delivery system.

Philosophy of Approach

The key to the provision of an effective, high added value mobile service is the ability to detect relevant events and to independently take the action that is most appropriate for the resulting situation. This situation triggered action approach relies on active context filtering to minimise the processing of situations (O'Brien, 2006). A review of previous work suggests that an automated ubiquitous mobile service delivery systems requires:

- An agreed ontology of situations that allows the specification of situations, contexts and situation action rules
- An agreed domain ontology for the application domain in which the system will be operating.

The mobile situation management ontology described in this paper forms the basis for the implementation of a system that addresses these requirements.

Methodology

The goal of this work is to find a satisfactory solution, not necessarily the best or optimal solution from amongst all the possible solutions. Generalisability of the artefacts produced by this work is not a key consideration as they will be the starting point for a line of future research in situation aware mobile service delivery systems and will be refined in future research. For these reasons, a *design research* approach, as proposed by Hevner et al (2004) has been used. Design research aims to create innovative artefacts that extend human or organisational understanding or capabilities (Hevner et al., 2004).

Considering this approach, the following tasks were undertaken:

1. To develop an awareness of the problem, interviews were undertaken with a small number of HMPs (O'Brien & Burmeister, 2003);
2. One service was selected from each of Angehrn's four virtual market spaces. The selected services are representative of the types of services in each space (O'Brien & Colomb, 2005). This approach is recommended by Hevner et al (2004);
3. Design of a generic situation ontology that can be used in the development of mobile service delivery systems.;
4. Evaluation of the generic situation management ontology by,
    - instantiating a travel domain ontology using a typical HMP scenario to ensure that the situation ontology has adequate expressiveness for real world situations.
    - developing an academic situation management ontology to demonstrate the adaptability of the generic mobile situation management ontology.

Where necessary minor design changes were made.

MOBILE SITUATION MANAGEMENT

Context & Context Awareness

Decisions and assertions that are made regarding HMPs must consider their current state AND context. For example, information about current location and activity only allows decisions and assertions to be made about this location at this time. It does not allow decisions to be made about other times and locations.

Lenat (1998) provides a detailed definition of the twelve dimensions of context that he considers to be essential for automated reasoning. Include are the dimensions of time and space, but also a number of other dimensions such as granularity and culture. There has been considerable debate in the literature about the value of these additional dimensions but there is general agreement that time and location are essential.

Assertions about something are generally only applicable in specific contexts. Therefore, an answer to a question about something must be a pair <context, answer>. That is, "the answer is …… in this context". For example, two possible answers to the question "Who is the Prime Minister" would be <1972-75, Gough Whitlam> and <1996 – 2007, John Howard> (Lenat, 1998)

Lenat's (1998) experience with contexts or micro-theories has shown that it results in simpler assertions and extensive re-use of assertions but also imposes the additional burden of choosing the most appropriate context from a large number of small contexts. Consequently, a trade-off needs to be made between coding time within the contexts and time to select the best context. Mobile data technologies allow data to be captured that simplifies the selection of the most appropriate context and hence, simplifies the automated decision-making process.

Dey et al (2001) define a context aware system as a system that "uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task".

Although a person may be in a particular physical environment, for example, his bedroom, the activity that he is engaged in within his bedroom environment determines what in the bedroom is important to him. If he is relaxing and reading then the light being on is important, however, if he is sleeping it is important for the light to be off and, perhaps for the telephone to be turned off. Knowledge of the user's current activity and preferences are therefore essential for context-aware automated decision-making. Despite time being identified as a primary dimension of context, Dey et al (2001) did not provide for temporal dimensions of context in their framework and toolkit for context-aware applications. As awareness of both temporal and location dimensions of context are essential for mobile service delivery systems, both have been included in the situation management ontology developed in this paper.

Gross and Specht (2001) show that delivery of services to highly mobile, or nomadic, users can be enhanced by using context awareness, user modelling and by localisation of service delivery. They show that context aware systems enable the delivery of services to mobile users that are filtered or adapted depending on the user and service contexts. However, the types of services that highly mobile users require are not heterogeneous. Angehrn (1997) proposed that these services can be categorised as information, communication, transactional and distribution services. Kanter (2000) and Schmidt (1999) further showed that the adaptation and/or filtering that is required differs according to the category of the service being delivered. By categorising services into Angehrn's four categories, a hierarchy of service types can be defined within each of the four dimensions

thereby facilitating more functionally cohesive service definitions and consequently more functionally cohesive software modules and/or agents to deliver the services.

Gross & Specht (2001) further assert that action should be taken and/or information should be provided 'if an event occurs in the same context as a user's current context'. While this does provide a level of filtering that eliminates actions for events that are "out of context", it may still result in irrelevant information or unnecessary actions. A further level of decision-making is required. Information and/or actions should only be taken if the contexts are the same *and* a particular state of affairs exists within that context(s). That is, a *reactive situation* exists.

Situations

Loke (2006) provides a very good summary of previous work on context aware systems and situations. He clearly distinguishes context and activities from situations, which he asserts is a similar but higher level concept. He argues that "situations can be useful….in that the situation abstraction allows the modeller to effectively carve the world up into manageable pieces". These situations can be combined to compose more complex situations.

There are a number of different definitions of the concept of a situation in the literature. While they are similar, they differ according to the viewpoint of the researcher. As my work is from an information systems viewpoint, I have focussed on the information systems definition of situations. Barwise (1987), Akman and Surav (1996), Adi et al (2002) and Fischer et al (2002) provide definitions that initially appear to be quite different, however they all take context into account, either explicitly or implicitly, and they all consider the state of things in the context.

From a theoretical perspective, situations do not necessarily require any action. Situations that do require some action to be taken, that is, reactive situations, require action rules to be defined. In this work I have therefore adopted the following, slightly modified version of Barwise's (1981) definition: "a situation is a state of affairs in a particular context" but I draw on Etzion & Adi's work to define reactive situation.

Rules need to be defined to specify the appropriate actions that should be taken in response to occurrences of each reactive situation (Adi et al., 2002). Where the same response is required for a number of reactive situations, these situations form a class or type of situation (Cherry, 2001) for which reaction rules must be defined.

Barwise (1987) also shows that situations are first class citizens (or objects), that can have properties and relationships with other objects including other situations. The relationships can be expressed by constraints or background conditions. Barwise and Perry (1983) found that states of affairs required spatio-temporal parameters to address seemingly incoherent situations such as Paul is talking and Paul is not talking which can both be true in different contexts that are not always persistent. This supports Lenat's (1998) position that the truth of any state of affairs can only be determined if we know its context.

Situation Theory

Situation theory was introduced by Barwise (1981). Barwise's ideas were then applied to logic by Barwise and Perry (1983). Perry (1993) then built on his early work with Barwise. He clearly distinguishes *situations* from *worlds*. "A world determines the answer to every issue, the truth-value of every proposition. A situation corresponds to the limited parts of reality we in fact percieve, reason about, and live in." (Perry, 1993) Consequently, situations only allow limited decision-making about events and states of affairs within particular contexts and domains.

Situation semantics is normally applied to the theory of natural languages, but it can be useful to define real world context dependent situations. Situation semantics is based on the concept that we classify parts or views of reality by considering the things, their properties and the relationships between them in that part of reality. The events that occur in a situation alter the values of the properties of things in that part of reality, that is, they change the state of affairs, in that context. A *state of affairs* may exist within a context because of events outside that context, but determination of whether a situation *exists* depends only on the *state of affairs* of the things within that context.

In situation theory, a key concept is that of a *type of situation*. The type of situation in which your car has a flat tyre (S_0) and the type of situation in which a car can't move (S_1) are examples. One type of situation may be related to or depend on another. For example, every time one situation occurs, another situation will also occur. Barwise and Perry (1983) say that S_0 *involves* S_1; cars with flat tyres can't move. They call these situations *constraints*, that is, a car that has a flat tyre is *constrained* from moving.

Devlin's (1991) book built on the work of Barwise and Perry and introduced a modern notation and terminology for describing situations such as these. In Situation Theory, *infons* are basic discrete pieces of information. Using Devlin's (1991) notation, an infon is represented as:

$$\langle\langle P, a_1, \ldots, a_n, i \rangle\rangle$$

where P is an n-place relation, a1, …, an are objects of the relation P and i is the polarity (0 or 1) indicating whether the relation holds.

Situations are related to infons by the supports operator ($\models$). $s \models \alpha$ indicates that the infon α is supported in situation s, or in other words, s makes it the case that α is true. For example, $s \models$ «bites, Fido, Mary, l, t, 1 » represents the situation where Fido bites Mary at some location (l) and time (t).

Devlin's notation and terminology are used to represent situations in this paper with the exception that the words True or False are used to represent polarity rather than the numbers 1 and 0. This has been done simply for clarity and to avoid confusion related to the representation of Boolean values in different programming languages.

Devlin (1991) also specified a number of basic types that are necessary for situation specification. They include TIM to specify time, LOC to specify location, IND to specify parameters, REL to specify relationships, SIT to specify situations and INF to specify infons. For example,

Ss = [s' | s' $\models$ << slaps, a', b', l', t', 1 >>]
St = [s' | s' $\models$ << touches, a', b', l', t', 1 >>]

Means that if a' slaps b' at l' (a LOC type) & t' (a TIM type) then a' also touches b' in the same l' & t'.

Factual infons and constraints are also required because some facts are only true under certain background conditions. A constraint C may only allow situation S_2 to be true when S_1 is true if the background condition B is true, that is $C = S_1 \rightarrow S_2 \mid B$ can also be written as an infon:

<<involves, S_1, S_2, B, 1 >> OR
<<involves, S_1, S_2, 1 >> | B

Akman and Surav (1996) show that contexts provide a useful way to apply context specific "rules and pre-suppositions related to a particular point of view" to simplify the specification and detection of situations. They show that when binding parameters in situations, "only objects that are in one of the current contexts need to be considered." Furthermore, they show that contexts can be considered as objects, that is, as "first class citizens". This means that they can be used in the same way as any other object so 'they can be denoted by constants in the logical language" and "variables can range over them (Akman and Surav, 1996)". Combining the work of Devlin and Akman &

Surav, I have combined TIM and LOC, together with other dimensions where necessary, into a single, complex type, context (CON). This simplifies the specification, filtering and comparison of situations.

Situation Management

Reactive systems are systems that react automatically to changes in the environment. They have been used in command and control, active databases, system management tools, customer relationship management systems and electronic commerce systems. This paper extends previous work that tended to provide mechanisms for reacting to single events only to one which reacts to situations.

To be able to detect situations some key information must be included in the situation definition or in related objects:

- The events that can participate in situation detection;
- Context during which situation detection is relevant;
- The semantic conditions that must be satisfied in order to detect a situation;
- Whether and under what conditions an event is consumed (Adi et al., 2002)

Adi and Etzion's event class does not include any location attributes. For mobile users it must be extended to include at least location attributes and most likely, other attributes of context. This is particularly important when inferring events from the world's state and concrete events such as user movement. Adi and Etzion's lifespan class defines a set of events that can initiate a lifespan, a set of events that can terminate a lifespan, conditions for initiation and termination and maximum lifespan length. To take account of location I have added a locationspan class with similar attributes to lifespan but related to location, namely,

- A set of events that can initiate a locationspan;
- A set of events that can terminate a locationspan;
- The conditions for initiation and termination of a locationspan;
- maximum locationspan radius;
- granularity of location.

I have also added a granularity of time attribute to the lifespan class.

Adi and Etzion group collections of semantically associated event instances into event groups. They are used to match different event instances that refer to the same entity or concept. An event collection is the collection of event instances that must be considered for situation detection. They are only evaluated if they occur during the time that a context that is associated with situation detection is active. I have used event collections to group together sets of events that result in a context and/or situation transition. A single event collection may include temporal events, location-based events and other service and/or user events each of which causes the same transition.

Although my work is based on Adi & Etzion's concepts, there are many differences, namely,

- Context is used to filter out irrelevant situations rather than just lifespan;
- An object reference attribute is used to relate a situation to event collection objects rather than simply an event collection name;

- Background conditions associated with a situation are specified using references to situation objects rather than using a simple situation expression.
- Adi & Etzion's Situation Manager uses lifespan to filter out irrelevant situations. If there is no open lifespan for a particular situation, the situation is ignored. This is clearly insufficient because a situation may still be irrelevant, even if there is an open lifespan, if there has been a change in some other dimension of context such as location, or even a change in user preferences. Therefore, for HMPs, situations need to be filtered by context, not just lifespan.

Cherry (2001) proposed the use of pre-specified rules, *situation-action rules*, to specify the actions that are required to be taken in response to the detection of a reactive situation.. He shows that rules can be defined for classes, or types, of situations rather than individual situations, thereby reducing the number of rules required. I have taken this approach, using Semantic Web Rule Language (SWRL) to specify the rules (SWRL, 2004).

ONTOLOGIES

In Artificial Intelligence (AI), an ontology is "the specification of a conceptualisation. That is, defined terms and the relationships between them, usually in some formal and preferably machine readable format" (Hendler, 2001). Guarino (1998) points out that the AI concept of an ontology generally refers to "an engineering artefact, constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary words." These formal ontologies provide a means of representing, or at least approximating, a conceptualisation of the world.

An Ontology of Situations

Matheus et al (2003) proposed a core ontology for situation awareness that defines a situation as a collection of goals, situation objects and relations between situation objects. Situation objects are "entities in a situation – both physical and abstract – that can have characteristics (i.e. Attributes) and can participate in relationships" (Matheus et al., 2003). This ontology was designed to be the core of a situation awareness system that assumes that the recognition of the existence of a particular situation will occur in the mind of a human operator who interfaces to the system through a human-computer interface. The inclusion of goals as attributes of a situation is inconsistent with the theoretical definition of a situation as "a state of affairs within a given context" (Barwise, 1987). A situation is independent of, but of course related to, the goals and action rules that determine the actions that should be taken when that situation occurs. However, from a practical perspective, if inclusion of goals as attributes reduces implementation time and/or efficiency then there is no reason to not include them.

Devlin (1991) defined a basic ontology of situation theory, based on Barwise and Perry's work (Barwise,1987; Barwise and Perry, 1983) and documented it using a new notation that he developed. Devlin's situation theory ontology was later used to develop a number of new situation-oriented languages (Tin et al., 1995) including Baby-Sit (KEE, 1993; Tin and Akman, 1994).

Barwise and Perry's situation theory and Devlin's notation are now well accepted within the situation theory research community. The adaptation of this ontology for use on the worldwide web provides a solid theoretical basis for the development of a situation management ontology for use in semantic web applications. OWL has become the language of choice for most of the recent work that involves the development of ontologies for the worldwide web (Fukazawa et al., 2006;

Jacobson et al.,2005; Matheus 2005). Since it was published, OWL has also been successfully used by Matheus (2005) to develop his core ontology for situation awarenesss. For these reasons, it was used to implement the core situation management ontology in this work.

Relationship between Events, Context and Situations

In this work, I have combined Barwise's TIM and LOC parameters, and a number of other dimensions of context into a complex parameter, context (or CON). For example the following situation type, specified using Devlin's notation,

$S = [\hat{s} \mid \hat{s} \models \text{«cancelled, Flight, Í, t', True»}]$

could be rewritten as,

$S = [\hat{s} \mid \hat{s} \models \text{«cancelled, Flight, c', True»}]$

Where S = situation type, ŝ = situation, ⊨ = supports, c' = context (composed of Í = location and t' = time) and « …. » = an infon.

The relationship between events, states, contexts, situations and actions is summarised below:

$A \rightarrow e$
$e \rightarrow \delta s$
$\delta s \Rightarrow \delta C \text{ OR } \delta s \Rightarrow \delta S$
$\delta C \Rightarrow \delta S$
$\delta C \Rightarrow S_R \mid B$
$\delta S \Rightarrow S_R \mid B$
$S_R \rightarrow A1, A2 \ldots An$

where e = event, δC = change in context, δs = change in state, S = situation, S_R = reactive situation, δS = change in situation, B = background situation (a situation as described in Section 0 above), A = action, → = "causes", ⇨ = "*may* cause" and | = "subject to".

Service or HMP actions cause events to occur. Events cause changes in the state of a HMP or a service. The change in state may cause a change in context, which in turn may cause a change in situation to occur. The change in state may also directly cause a change in situation without a change in context. A change in context may cause a change in situation. A change in context may cause a reactive situation to occur subject to a background situation existing. Similarly a change in situation may cause a reactive situation to occur subject to a background situation existing. In response to the occurrence of a reactive situation one or more actions are taken.

An event that can cause a transition to a new context is included in an event collection for that transition. Background conditions cause normal and/or reactive situations to occur.

SITUATION MANAGEMENT EXAMPLE

A simple example of a typical mobile user scenario and how the situations that occur would be handled by the system are given below.

> *The date is 26 March 2007. Fred is traveling around the islands of the South Pacific on business. He is currently in Brisbane and is scheduled to travel to Vanuatu tomorrow and then on to the Solomon Islands on 28 march 2007. Fred is unaware that a serious conflict has broken out in Solomon Islands. He is not contactable because he is in a high level meeting. The US State Department issues a travel warning for the Solomon Islands. The*

system detects the warning, selects relevant contexts and checks to see if there are any HMPs registered in these contexts and sends the required alert through to them. The system then checks Fred's itinerary for planned Solomon Islands trips in the near future. The system identifies that a reactive travel safety situation exists for Fred's planned Solomon Islands trip and takes appropriate actions to cancel this leg of his trip. The next time Fred switches on his handheld device, the alert is displayed advising him of the situation and the changes to his itinerary.

The scenario above assumes that Fred has set his preferences to allow the system to automatically change his itinerary without consulting him. Each HMP will have different preferences regarding this and other automatic responses. HMP's must be able to set their own preferences regarding what actions the system is allowed to take before consulting them including level of automation of itinerary changes, granularity of location and time for alerts, type of alerting preferred etcetera. The generic situation ontology described in this paper includes a class SERVICE_PREFERENCE that stores the HMPs preferences for a particular type of service, such as the Transport Service that would manage the situation described in the scenario above. The mobile situation management ontology must include information regarding these preferences so that system designers can use them to determine what, if any, independent action should be taken.

The class (or type) of reactive situation, itineraryChangeRequired (SR1) can be specified by the following situation type definitions:

$S_1 = [\hat{s} \mid \hat{s} \models \langle\langle\ travelSafetyWarning, c', True \rangle\rangle]$
$S_2 = [\hat{s} \mid \hat{s} \models \langle\langle hasFlightReservation, p', c', True\ \rangle\rangle]$
$SR_1 = [\hat{s} \mid \hat{s} \models \langle\langle itineraryChangeRequired, p', c', True\ \rangle\rangle]$
$C_1 = [S_1 \Rightarrow SR_1 \mid S_2]$

The *specific instances* of the situations that relate to the example scenario above are:

$s_1 \models \langle\langle\ travelSafetyWarning, SolomonIslands070328, True \rangle\rangle$
$s_2 \models \langle\langle hasFlightReservation, Fred, SolomonIslands070328, True\ \rangle\rangle$
$sr_1 \models \langle\langle itineraryChangeRequired, Fred, SolomonIslands070328, True\ \rangle\rangle$

Although this is a very simple example it does demonstrate how the system would handle a typical HMP scenario.

A MOBILE SITUATION MANAGEMENT ONTOLOGY

A working mobile situation management system requires the definition of a task ontology for situation management and a domain ontology for the domain in which situations need to be managed. A high level, or generic, situation management ontology can be specified independent of the domain in which it will be used. The primary concern of this high level situation management ontology is to specify the concepts that are important to the generic task of managing situations. Similarly, domain ontologies can be initially specified at a high level for a particular domain.

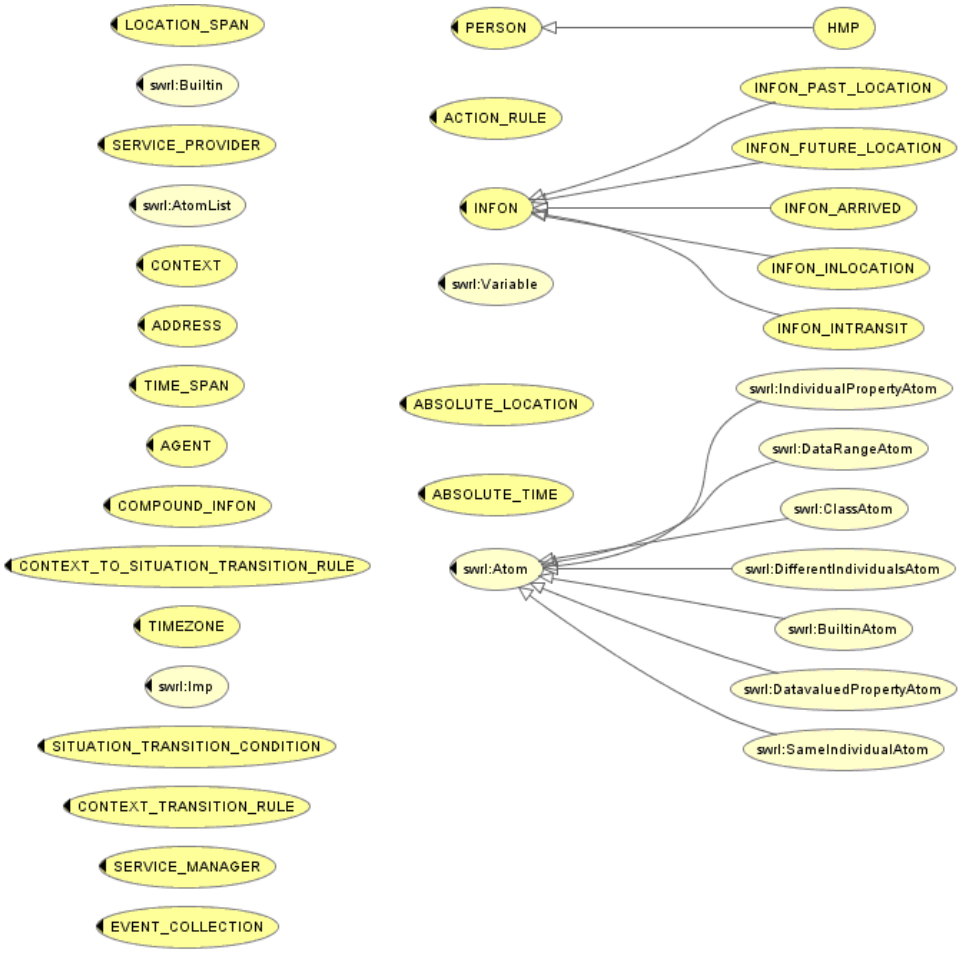
The high level situation management and domain ontologies must then be combined and specialised to create a situation management ontology for a particular domain. For example, a nuclear power station safety system would require that specific situations, and the required responses to those situations, related to nuclear power station safety be defined. While the generic situation management ontology provides the underlying mechanism for reacting to any situation, the specific situations related to nuclear power stations and the rules defining how to react to them must be specified within the domain ontology of nuclear power stations. Situation constraints can be

specified within the ontology itself or they can be specified using RuleML (RuleML, 2003) or SWRL (SWRL, 2004).

The generic, high level situation management ontology is described below. An example application ontology that combines this generic situation management ontology with a domain ontology for the travel domain can be found in Appendix 2. A concrete ontology containing instantiations related to the travel domain, detailed OWL specifications of these three ontologies, and sample rules defined in Semantic Web Rule Language (SWRL) are provided in O'Brien (2006).

GENERIC SITUATION MANAGEMENT ONTOLOGY

The generic situation management ontology is shown in Figure 1.



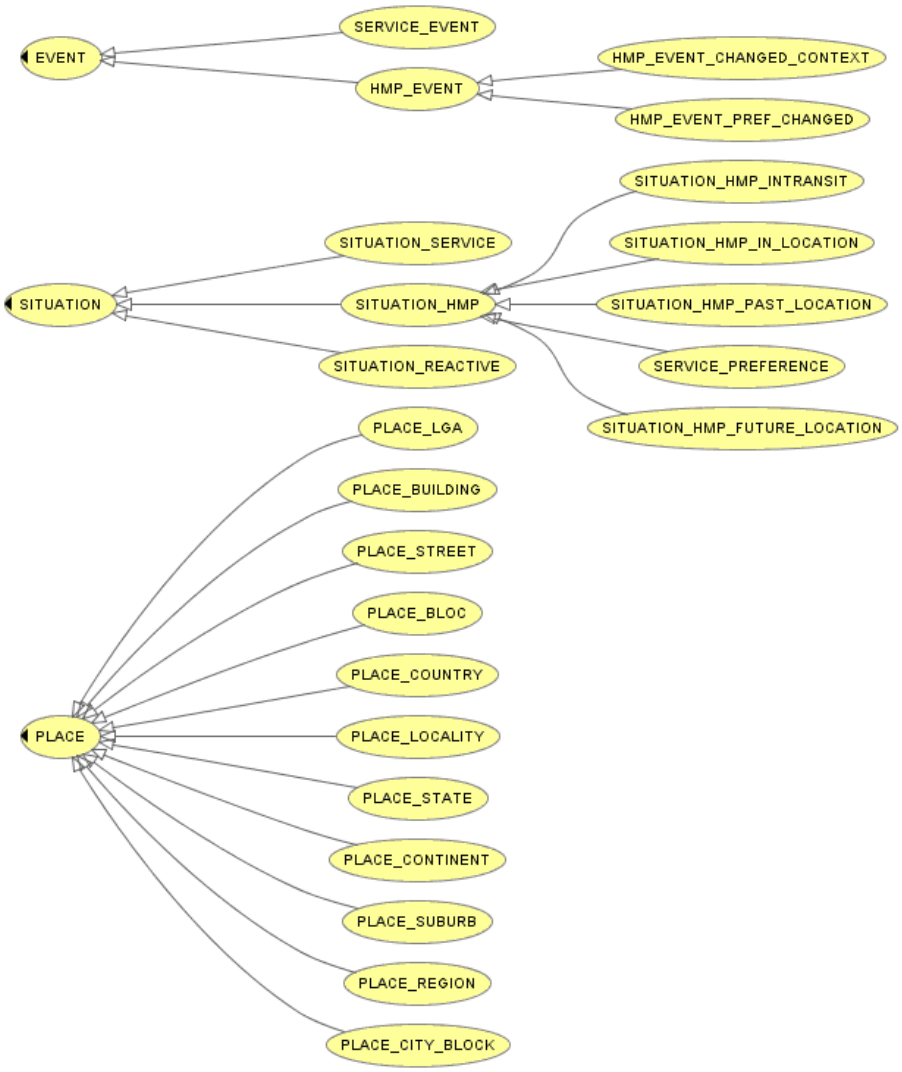


Figure 1 Generic Situation Management Ontology Hierarchy

Each of the classes in the Generic Situation Management Ontology and their properties, together with suggestions on how to use each are described in Appendix 1.

More detailed descriptions of each of the classes and the full OWL specification of the ontology can be found in O'Brien (2006).Generic Situation Action Rules

The situation-action rules are stored on a server and are accessed via a unique URI that is encoded into the ACTION_RULE class. They can be specified in a number of different formats such as

RuleML, a description logic language such as Prolog or in semantic web rule language (SWRL, 2004). SWRL is recommended for all future rule development as it is now a World Wide Web Consortium Recommendation and is therefore likely to become the defacto standard in the future. Specific action rules for particular reactive situations have not been defined in this work, however, the logic of reactive situation detection and action can be described as:

```
if ServiceSituation(c) then
  for all HMPSituations(c') or ServiceSituation (c')
        where (c = c' or  c intersects c')
    if HMPSituations(c') is a background condition on
       ReactiveSituation (c') or ServiceSituations(c') is a
         background condition on ReactiveSituation (c')
       then ReactiveSituation(c) is true
  endif
endif
```

where ServiceSituation(c) is a situation that exists because of a HMP's state of affairs of a service in context c, HMPSituation(c') is a situation that exists because of a HMP's state of affairs in context c', background condition is a required condition of a defined situation constraint and ReactiveSituation(c) is a consequential situation that exists that requires some action to be taken.

Similarly,

```
if HMPSituation(c) then
  for all HMPSituations(c') or ServiceSituation (c') where (c = c' or  c intersects c')
        if HMPSituations(c') is a background condition on ReactiveSituation (c')
                or ServiceSituations(c') is a background condition on ReactiveSituation (c')
then ReactiveSituation(c') is true
  endif
endif
```

Problems and Issues Encountered

As can be expected in an "inherently iterative and incremental activity" such as design science research (Hevner et al., 2004), some problems and issues were encountered when developing the ontology. These are discussed below together with the adaptations of the original design that were made or are recommended to address the problems and issues.

ONTOLOGY DEVELOPMENT

In the process of developing the generic situation management ontology some unexpected, but resolvable, issues and problems were encountered. They were generally related to the ability to represent things and relationships in different ways. Nothing was encountered that could NOT be represented in the ontology. This section discusses these issues and the decisions that were made in relation to them.

Specification of location

As previously discussed, a key dimension of context is spatial location. To effectively represent spatial location in the real world requires the ability to specify a point in space, absolute location, and an area or volume in space, that is, a location span is necessary. For example real world things

such as rooms, buildings, city blocks, cities, countries and continents all have a two dimensional footprint, and some are three dimensional.

Specification of a point in two dimensional space can be achieved by simply specifying a geographical co-ordinate such as latitude and longitude. I decided to use latitude and longitude to specify absolute location as it is universally understood and accepted. To allow specification of a point in three dimensional space an additional dimension of altitude was added as it is required in some domains, such as air travel.

Specification of an area on the earth's surface can be achieved in many ways such as a bounding rectangle, a bounding circle or a bounding polygon. A simpler, and more flexible geocoding system, Natural Area Coding (NAC) was chosen as it can be used to specify an area on the earth of virtually any size, and if necessary, a volume of any size anywhere in the universe (NAC, 2006).

While it is a proprietary encoding scheme, it has been gaining wide acceptance and is flexible enough to enable specification of everything from a continent to an absolute location. However, since conversion between NAC format and latitude/longitude can be achieved with simple algorithms I decided to use NAC encoding for specification of areas in conjunction with latitude and longitude. I have called this dimension of context *locationspan*.

Specification of Time

Another key dimension of context is time, or temporal location. Similar to spatial location, a temporal location can be absolute, that is a specific point in time, or it can span many points in time, that is, it can have a duration. As local time is relative to location, a universal absolute time is necessary to allow comparison of time in different locations. The most common absolute time for civil usage is Co-ordinated Universal Time (UTC). I have used UTC as my ontology is expected to be primarily used to implement civil applications.

In addition to specification of absolute time, the local timezone must be specified to allow appropriate reaction to situations. For example, opening and closing times of hotels and restaurants, arrival and departure times of flights and flight curfews are generally determined by local time, not by UTC. The Timezone entity in the generic ontology allows the number of hours ahead of UTC to be specified for the local timezone.

Duration, which I have called *timespan* in this work, is specified by an absolute start time specified in UTC and an absolute end time, also specified in UTC. For ease of reading and automation an additional property, *granularity*, is included, which can take values such as second, minute, hour, half-day, day etcetera.

Specification of Contexts

As the reaction to situations can depend on the temporal and spatial subsumption relationship between the user context and the service context, the context class includes object properties for specifying which contexts it subsumes (*SpatiallySubsumes* and *TemporallySubsumes*) and contexts by which it is subsumed (*IsSpatiallySubsumedBy* and *IsTemporallySubsumedBy*).

For the situation type S where S = [s | |= <<isPrimeMinister, c', John Howard, True>>], an example of a situation that is a member of S would be one in which the context was 31 March 1996 and Australia because the timespan "2 March 1996 to today" subsumes 31 March 1996. A context of 31 March 1996 and Brisbane would also create a situation that supports the infon because Australia subsumes Brisbane.

To specify the above, each context with a locationspan of Australia would have Brisbane included in its set of SpatiallySubsumes properties and each context with a locationspan of Brisbane would have Australia included in its set of IsSpatiallySubsumedBy properties. Similarly the timespan of John Howard's term of office as Prime Minister would temporally subsume all days, months and years within his term and this would be specified in the TemporallySubsumes property of each context that has a timespan of Howard's term in office.

Context Property of Infons and Situations

The difference between an infon and a situation is that an infon can exist without being situated. When it becomes situated it becomes a fact. For example, the infon <<isPrimeMinister, John Howard, True>> and <<isPrimeMinister, John Howard, False>> are both valid infons although they appear to be contradictory. This is because they are not situated. When we put either of these infons into a particular context, it becomes either a truth (or fact) or a falsehood depending on the context. If the context is the United Kingdom in 2006 then it is clearly false but it is true, or factual, in Australia in 2006. A situation is a state of affairs (infon) in a particular context, so the infon

<<isPrimeMinister, John Howard, True>>

is supported in all contexts where country equals Australia, and the date is between 2 March 1996 and today. That is,

S = [s | |= <<isPrimeMinister, c', John Howard, True>>]

An infon in a specific context is a situation. For this reason the infon class of the situation ontology does not have a context property. The context property is specified in the situation class, together with the associated infons and background conditions. This simplifies the specification of the generic situation ontology and the action rules associated with it.

Specification of Situation and Context Transition Rules

Event collections cause a transition from one context to another but do they also cause a transition from one situation to another? Events can cause a change in context and they can directly cause a new situation to occur with or without a change in context. An event will create a new state of affairs, that is a new infon, in an existing context. The combination of this new infon in an existing context will result in a new situation if that state of affairs did not previously exist. Therefore, both context transition rules and situation transition rules should include any relevant event collections. Situation constraints are still required to specify the transitions.

Ontology Instantiation

When instantiating the ontology for the travel domain it became clear that instantiating all of the objects and their properties for the sample scenario would be a repetitive task that would not provide any additional insights in the evaluation of the ontology. Although the entire scenario was instantiated as part of the project, only one instantiated object of each class is included in O'Brien (2006) as examples. Similarly for the second concrete ontology for the academic domain.

Rule Specification

As production of a prototype was outside the scope of this work, detailed action and context subsumption rules were not developed. However, consideration was given to the rule specification methods that were available. The contemporary choices for web based situation management systems include RuleML, Semantic Web Rule Language (SWRL) and the use of situation constraints. It was clear from the development of the situation ontology that the use of situation

constraints to specify complex rules was not an appropriate choice. As an SWRL add-on is now available for the Ontology Editor, Protégé (Protégé, 2006), and Protégé can export its data in a format that is compatible with agent and expert system development tools such as the Java Expert System Shell (JESS) (JESS, 2006), I recommend that SWRL be used to specify the rules for the prototype.

Scalability

Rules and actions are defined for classes or types of reactive situations, not individuals. The complexity of the rules versus the granularity of the situation classes and contexts is a trade-off. The choice of granularity depends on the power of the end-user devices, the context granularity of the service to which the user subscribes and the users desired level of automation. If the context granularity is kept large, the complexity of the situation detection process is minimised, however, the complexity of the action rules will increase as more logic will be required to determine the most appropriate actions. This is appropriate for today's mobile infrastructure where the system components would execute on large internet based servers and handheld devices are of limited capability. In the future, as handheld devices become more powerful, the context granularities could be reduced to provide more specific situation detection and management.

Also, many of the current service providers only provide information at quite coarse spatio-temporal granularity, making it pointless to have a situation detection and management system that uses a finer granularity. For example, the Australian Department of Foreign Affairs and Trade (DFAT) issues travel warnings by country, not by city, state or region. However, transport situations need to be at the granularity of city and day to be able to manage flight cancellation and delays, train trips etcetera.

SUMMARY

This research contributes a novel design artefact, namely a generic situation management ontology, thus satisfying the first type of contribution. It also contributes to the foundation knowledge base of mobile service delivery systems by providing a generic situation management ontology, based on situation theory, that can be used in future research and systems design. This demonstrates the applicability, and feasibility of using situation theory in the design of reactive information systems.

Furthermore, the support within the ontology for context based filtering for situation detection contributes to the efficiency of implementation and operation of situation driven reactive information systems.

Hevner et al (2004) state that any artefacts produced by design science research "must be implementable". Representations of contexts, situations and rules must be available in an agreed digital form before an automated reactive system can be developed. Recently XML has become the standard for encoding data that will need to be shared across heterogeneous systems. However, XML by itself does not carry any semantic information related to the data that is included in the document. The semantic information comes from the agreed meaning of the tags surrounding the data. As the recent research in contexts, situations, ontologies and rule specification has been carried out in a number of different projects, in a number of institutions by researchers with quite different viewpoints on the subject areas, there is neither a single ontology nor a single markup language that covers all these areas. Bearing this in mind, the situation management ontology developed in this work, which covers contexts, situations and situation action rules were encoded in the de facto standard ontology and rule markup languages, OWL and SWRL. This ensures that they are relevant to current and future work being undertaken in web-based information systems. Ontology

development and management tools such as Protégé (Protégé, 2006) that have been used in this research have built-in support for OWL and SWRL, simplifying the development of OWL based ontologies. Furthermore, the ontologies developed in these tools can be exported in a form suitable for use in agent development systems such as JESS. Thus, the requirement of implementability is satisfied.

In summary, this paper presents a generic situation management ontology and generic action rules that are:

- based on Etzion and Adi, Cherry and Lenat's work;
- semantically consistent;
- specified in the XML based, industry standard specification languages, OWL and SWRL;
- able to be adapted to specific domains such as travel and academic support services

This work has also introduced the concept of a reactive situation management system that utilises contexts to eliminate irrelevant situations from consideration for action. The aim is to improve system efficiency and accuracy. It has brought together concepts from ontology theory, context theory and situation theory in the design of a general situation management ontology that can extended to a domain specific situation management ontology. Although the travel domain was used to exemplify how the system works, it is not restricted to the travel domain. As the system is ontology driven, it can be adapted to a new domain by replacing the travel related ontology tags with tags related to the new domain. To further validate the generic ontology, an academic situation management ontology was developed by simply replacing the travel specific components in the ontology with academic domain components. See Appendix 3. Very few changes to the generic travel domain ontology were required to change it to the generic academic domain ontology, confirming that the generic situation management ontology is flexible enough to be adapted to any domain. This was a key goal of this research. More detailed descriptions of each of the classes and the full OWL specification of the concrete academic ontology can also be found in O'Brien (2006).

LIMITATIONS

Due to the limited time and resources available, some limits must be set to the scope of any research project. Similarly, technical, financial and organisational factors also limit their outputs and the implementability of the outputs in practice. In this section, the key limitations and their effects are discussed.

Design Evaluation and Generalisability

As no working prototype has been developed to test the effectiveness and efficiency of the proposed ontology, it was not possible to verify the effectiveness of the ontology in an operational system. However, the ability to easily specify the required concepts and rules has been demonstrated. All that is left to be done to develop a working prototype is the coding to implement the required agents and service managers. While this is not a trivial task, it is also not a difficult task for an experienced software developer using modern software development tools. For this reason, this work has concentrated on the novel and complex tasks of ontology design. A consequence of this is that testing of the design had to be done on paper by working through a scenario. While this has indicated that the design is effective, more rigorous testing will require the development of a live working prototype.

Recommendations for Future Work

This work provides a foundation on which future work can be carried out to assess the effectiveness of context filtered situation management systems. There are a number of areas where work still needs to be carried out before working systems can be deployed in practice.

The next step will be the development of a working prototype of a mobile situation management system that is based on the generic situation management ontology. This will most likely be developed using JESS and JATlite as the ontologies that have been developed in this work can be exported in a format that can be directly accessed by these tools. Once developed, the effectiveness of the prototype system can be evaluated through controlled experiments.

REFERENCES

Adi, A., et al. (2002) "The Situation Manager Component of Amit - Active Middleware Technology", Lecture Notes in Computer Science (2382), London, Springer-Verlag, pp 158-168.

Akman, V. and Surav, A. (1996) "Steps Towards Formalizing Context", Artificial Intelligence Magazine, Vol 17 No 3, pp55-72.

Barwise, J. (1981) "Scenes and other situations", Journal of Philosophy, Vol 78 No 7, pp369- 397.

Barwise, J. (1987) The Situation in Logic, CSLI Lecture Notes 17, pp 59-77, Centre for the Study of Language and Information, Stanford University, California

Barwise, J. and Perry, J. (1983) Situations and Attitudes, MIT Press, Cambridge.

Cherry, G. (2001) Extending and Formalizing The Object-Oriented Paradigm with Situation-Driven Modeling, Thought**Tools Corp. http://www.sdml.com accessed 5 September 2007.

Devlin, K. (1991) Logic and Information, Cambridge University Press, Cambridge.

Fischer, K. et al. (2002) "Multimedia Workplace of the Future MAP". http://informatiksysteme.pt-it.de/ mti-2/cd-rom/projects/**map**/beitrag_MAP.pdf accessed 7 September 2007.

Fukazawa, Y. et al. (2006) "Situation-Aware Task-Based Service Recommendation", In Proceedings of the Fourth International Conference on Mobile Systems, Applications and Services, mobisys 2006, Uppsala, Sweden.

Guarino, N.(1998) "Formal Ontology and Information Systems". In Formal Ontology in Information Systems (N. Guarino ed.), IOS-Press, Amsterdam.

Hendler, J. (2001) "Agents and the Semantic Web", IEEE Intelligent Systems, Vol 16 No 2, pp 30-37.

Hevner, A. et al. (2004) "Design Science in Information Systems Research", MIS Quarterly, Vol 28 No 1, pp 75-105.

Jakobson, G. Lewis,et al. (2005) "Overview of Situation Management at SIMA 2005". In Proceedings of SIMA 2005, Atlantic City, New Jersey.

JESS (2006) "JESS, The Rule Engine for the Java Platform, The Java Expert System Shell". http://www.jessrules.com/ accessed 6 September 2007.

KEE (1993) "KEE tm (Knowledge Engineering Environment) Software Development System, Version 4.1", Intellicorp, Inc., Mountain View, CA.

Lenat, D. (1998) "The Dimensions of Context-Space", Cycorp, Austin. www.cyc.com accessed 8 September 2007.

Loke, S. (2006) "Context-Aware Pervasive Systems: Architectures for a New Breed of Applications", CRC Press, Boca Raton.

Matheus, C. (2005) "Using Ontology-based Rules for Situation Awareness and Information Fusion". In Proceedings of the W3C Workshop on Rule Languages for Interoperability, Washington.

Matheus, C. et al. (2003) "A Core Ontology for Situation Awareness". In Proceedings of Sixth International Conference on Information Fusion, pp 545-552, Cairns, Australia.

MIT (2004) "MIT Oxygen Project". http://oxygen.lcs.mit.edu accessed 6 September 2007.

NAC (2006) "Natural Area Coding, NAC Geographics Inc. http://www.nacgeo.com/ accessed 10 September 2007.

O'Brien, P. (2006) "An Architecture for Ubiquitous Mobile Service Delivery",PhD thesis, School of ITEE, University of Queensland, Australia. http://adt.library.uq.edu.au./public/adt-QU20060929.120135 accessed 12 October 2007.

O'Brien, P. & Burmeister, J. (2003) "Ubiquitous Travel Service Delivery", International Journal of Information Technology and Tourism, Vol 5 No 4, Cognizent, New York.

O'Brien, P. & Colomb, R. (2005) "Generic Context Driven Situation Detection and Management". In Proceedings of the IASTED International Conference on Internet and Multimedia Systems and Applications 2005, Grindelwald, Switzerland.

Perry, J. (1993) The Problem of the Essential Indexical and Other Essays, Oxford University Press, New York.

Protégé. (2006) "The Protégé Ontology Editor and Knowledge-Base Framework", Stanford University. http://protege.stanford.edu/ accessed 5 October 2007.

Ruleml (2003) http://www.dfki.uni-kl.de/ruleml accessed 1 October 2007.

SWRL (2004) "SWRL: A Semantic Web Rule Language Combining OWL and RuleML". http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/ accessed 15 September 2007.

Tin, E. et al (1995) "Towards Situation-Oriented Programming Languages", ACM SIGPLAN Notices, Vol 30 No 1, pp 27-36.

WIMAX (2006) The WIMAX Forum. http://www.wimaxforum.org accessed 6 October 2007.

APPENDIX 1 GENERIC SITUATION MANAGEMENT ONTOLOGY

CLASS DESCRIPTIONS

ABSOLUTE_LOCATION

The Geocode for an absolute geographical location. Latitude and longitude are standard Geocode values expressed in degrees. Both are mandatory. Altitude is optional. All three are single precision floating point number. Absolute location is used to specify a point on or above the surface of the earth. Absolute location is needed to specify centres, the start and end of a locationspan and the corners of areas and volumes.

ABSOLUTE_TIME

Absolute time specifies the absolute time in UTC irrespective of the location. This can be combined with Timezone to determine local time.

ACTION_RULE

This class is used to initiate an SWRL action rule. The URI of the rule is specified together with the name of the function that is to be executed and any required parameters for the function.

ADDRESS

This is the street address of a place, typically a building or landmark. It is composed of the names of the country, city, suburb, street name and number and the postcode. All are strings. Street, city and country are mandatory.

AGENT

The Agent class is used to specify the unique identifier, name and URI of software agents that provide a virtual presence for HMPs and service providers. Up to 20 functions can be specified for each agent, allowing requests such as reservation requests, to be sent to the service provider.

COMPOUND_INFON

This class allows an infon to be specified that is a logical OR or AND combination of two individual infons. If negation is required for one of the infons, this can be achieved by setting the opposite polarity in that infon.

CONTEXT

The context class is used to fully define a unique context. Each instance of the class has a unique context identifier that can be used for querying and comparison. Each context instance must have one or more timespans and one or more locationspans specified. If the context subsumes other contexts these can be specified with the CONTEXT_Subsumes object property. If the context will change from this context to other contexts or situations, the rules under which these transitions will occur can be specified in the CONTEXT_Transition_Rules object property.

CONTEXT_TO_SITUATION TRANSITION_RULE

This allows one or more transitions rules to be specified for transitions from the context in which this rule is specified to situations. Each transition rule is composed of the destination situation and the conditions under which the transition occurs.

CONTEXT_TRANSITION_RULE

This allows one or more transition rules to be specified for transitions from the context in which this rule is specified to other contexts. Each transition rule is composed of the destination context and the set of events each of which causes a context transition (the event collection).

EVENT

The event class is used to define an event that causes a change of state of a HMP or service. The time the event occurs, the time it is detected, the certainty that the event occurred and the length of time that the event will remain active are common properties to both the HMP_EVENT and SERVICE_EVENT subclasses.

These properties can be used to determine whether an event should be added or removed from an event collection.

EVENT_COLLECTION

All the events that can individually cause a transition to a particular new context are included in a set of events, EVENT_COLLECTION_Events.

The EVENT_COLLECTION_Group_Membership_Condition object property is used to specify the CONTEXT_TO_SITUATION_TRANSITION_RULEs and/or CONTEXT_TRANSITION_RULEs that determine what the new situation or context will be if one of the events in the event collection occurs.

HMP

The highly mobile person class (HMP) is a sub-class of PERSON and is used to specify whether a particular HMP wishes to have visible and/or audible alerts and to set the service time and place granularity preferences.

HMP_EVENT

Defines an event that causes a change of state of a HMP. This is a sub-class of PERSON that includes one additional property that identifies the HMP that generated the event.

HMP_EVENT_CHANGED CONTEXT

Defines an event where a HMP's current context changes. This is a sub-class of HMP_EVENT that includes additional properties that specify the context the HMP has moved from and the context the HMP has moved to.

HMP_EVENT_PREF_CHANGED

Defines an event where a HMP's service preferences change.

This is a sub-class of HMP_EVENT that includes additional properties that specify new service preferences.

INFON

This class allows one unit of information to be specified, as defined by Barwise (1987). The properties of this class are:

- INFON_Element that allows one or more objects to be specified as parameters. Any object can be specifed as an element.

- INFON_Polarity is a Boolean that takes the values TRUE or FALSE and is applied to the relation property.
- INFON_Relation specifies the relationship between the elements.

INFON_ARRIVED

An infon that asserts that something such as a HMP has arrived in a location. This is a sub-class of INFON.

INFON_FUTURE_LOCATION

An infon that asserts that something such as a HMP will be in a location in the future. This is a sub-class of INFON. It is used by SITUATION_HMP_FUTURE_LOCATION class to specify the future location in which a HMP situation could exist.

INFON_INLOCATION

An infon that asserts that something such as a HMP is currently in a location. This is a sub-class of INFON. It is used by SITUATION_HMP_IN_LOCATION class to specify the current location in which a HMP situation exists.

INFON_INTRANSIT

An infon that asserts that something such as a HMP is travelling between two locations. This is a sub-class of INFON. It is used by SITUATION_HMP_INTRANSIT class to specify the context that a HMP has left. The context that the HMP is transiting to is specified in the SITUATION property SITUATION_Context.

INFON_PAST_LOCATION

An infon that asserts that something such as a HMP was in a location in the past. This is a sub-class of INFON. It is used by SITUATION_HMP_PAST_LOCATION class to specify the past location in which a HMP situation existed.

LOCATION_SPAN

This class is used to specify a geographical area. The area is specified by a corner which is an ABSOLUTE_LOCATION object, a width and a length in metres. These three properties are mandatory. Additional optional properties can be used to specify other LOCATION_SPANs that are spatially subsumed by this object, other LOCATION_SPANs that spatially subsume this object, altitude in metres and Natural Area Code (NAC).

PERSON

A human being who is currently living, will be alive in the future or was alive in the past. The properties of this class include a unique person identifier, the person's name and the contexts in which the person was, is or will be alive.

PLACE

A type of physical location. The location is specified using a LOCATION_SPAN object and a name.

PLACE_BLOC

A group of countries that form a political, economic or social group. This is a sub-class of PLACE that includes an object property that allows a PLACE_COUNTRY object to be specified for each country in the bloc.

PLACE_BUILDING

A physical structure. This is a sub-class of PLACE that includes an object property that allows the address of the building to be specified.

PLACE_CITY_BLOCK

A city block typically bounded by a number of streets. This is a sub-class of PLACE that includes an object property that allows a PLACE_STREET object to be specified for each of the streets that bound the block.

PLACE_CONTINENT

One of the seven continents. This is a sub-class of PLACE that includes an object property that allows a PLACE_COUNTRY or PLACE_BLOC object to be specified for each country or bloc on the continent.

PLACE_COUNTRY

A sovereign state. This is a sub-class of PLACE that includes properties that allow cultural and economic details of the country to be specified. These properties are its name, international three letter country code, international telephone country code, three character currency code for each currency that is legal tender, official spoken languages, religions practiced, and the LOCATION_STATE objects that it contains.

PLACE_LGA

A local government area. Typically a city or shire. This is a sub-class of PLACE that includes properties that allow the timezone of the LGA and the suburbs that it contains to be specified.

PLACE_LOCALITY

A part of a city. This is a sub-class of PLACE that includes properties that allow the postcode of the locality and the city blocks that it contains to be specified.

PLACE_REGION

A part of a state that generally contains a number of LGAs. This is a sub-class of PLACE that includes a property that allows the LGAs that it contains to be specified.

PLACE_STATE

A province of a country. This is a sub-class of PLACE that includes a property that allows the LGAs and Regions that it contains to be specified.

PLACE_STREET

A named road within a city. This is a sub-class of PLACE.

PLACE_SUBURB

A named area of a city that generally contains a number of localities. This is a sub-class of PLACE that includes a property that allows the localities that it contains to be specified.

SERVICE

An information, communication, distribution or transaction service provided by a service provider. This is specified within the SERVICE_PROVIDER Class.

SERVICE_EVENT

Defines an event that causes a change of state of a service provided by a service provider. This is a sub-class of EVENT that includes a property to specify the SERVICE_MANAGER object that was the source of the event.

SERVICE_MANAGER

A software agent or program that manages one or more service providers of a particular type. This class includes properties that allow the type of service that the agent manages, the service providers that it manages, the URI of the agent and a unique service manager identifier, to be specified for each service manager.

SERVICE_PREFERENCE

This is a sub-class of SITUATION_HMP that allows situations that result from HMP service preference settings to be specified. Properties are provided to set the granularity of time and space for a particular service type, that is, a particular service manager.

SERVICE_PROVIDER

A physical provider of a particular service such as an airline. Properties are provided to specify its name, its street address, the software agent (SA) that interfaces to it, its code (such as IATA code), the contexts in which it operates, the service manager that manages it, and the URI of the service provider's website.

SITUATION

A state of affairs that is true in a particular context. This class allows a situation as defined in Situation Theory (Barwise 1987) to be specified. All of the elements that are required to define a situation using Devlin's (1991) notation can be specified using the properties of the SITUATION class. The background condition or constraint required for the situation to exist, the Context in which the situation exists, the infons the situation supports, the conditions under which a transitions will be made from this situation to other situations and a unique situation identifier can all be specified.

SITUATION_HMP

A state of affairs of a HMP that is true in a particular context. This is a sub-class of SITUATION that allows a situation that is associated with a HMP to be specified. This is achieved by setting the HMP property.

SITUATION_HMP_FUTURE_LOCATION

A state of affairs, or an infon, where it is true that a particular HMP will be in a location at a particular time in the future. This is a sub-class of SITUATION_HMP.

SITUATION_HMP_INTRANSIT

A state of affairs, or an infon, where it is true that a particular HMP is in transit between two locations. This is a sub-class of SITUATION_HMP.

SITUATION_HMP_IN_LOCATION

A state of affairs, or an infon, where it is true that a particular HMP is in a location at a particular time. This is a sub-class of SITUATION_HMP.

SITUATION_HMP_PAST_LOCATION

A state of affairs, or an infon, where it is true that a particular HMP was in a location at a particular time in the past. This is a sub-class of SITUATION_HMP.

SITUATION_REACTIVE

A situation that requires some action to be taken. This is a sub-class of SITUATION. The additional property SITUATION_REACTIVE_Reaction allows one or more reactions to be specified for each reactive situation.

SITUATION_SERVICE

A state of affairs, or an infon, of a service that is true in a particular context. This is a sub-class of SITUATION. The service that the situation is related to is specified by setting the SITUATION_SERVICE_Service object property to the service manager for this type of service. The severity of the situation can be set through the SITUATION_SERVICE_Severity property. This can be used to match user preferences with service situations by severity level.

SITUATION_TRANSITION_CONDITION

A rule defining what background conditions can cause a transition from the current situation to a new situation. One or more background conditions or constraints can be set any one of which, if true, will cause a transition the new situation. The background conditions are themselves situations.

TIME_SPAN

A period of time. This class allows the start and end time, the type of time and the granularity of the time period to be specified. Granularity can be set to a value in an enumerated type that has common English descriptions of time periods such as second, minute, hour etcetera. Similarly, type of time can be set to a value in an enumerated type that has common English descriptions of time periods in which particular types of activities take place such as worktime, leisuretime, sleeptime etcetera. Additional descriptions can be added to the enumerations as required. Properties can also be set to specify other time periods that this time period subsumes or that it subsumes.

TIMEZONE

A standard international timezone. This class allows a timezone code to be associated with a period of time ahead of UTC. As all times are specified in UTC, objects of this class can be used to convert UTC to local time.

APPENDIX 2: ADDITIONAL TRAVEL DOMAIN SPECIFIC CLASSES

Based on the identified high value services and a real world independent traveller scenario, the generic situation management ontology was extended to include additional classes that are specific to the travel domain. The new classes that were added are described below.

HMP_EVENT_FARE_PAID

Event where HMP pays fare for a reservation

HMP_EVENT_RES_MADE

Event where HMP pays makes a reservation

INFON_HAS_RESERVATION

Infon describing state of affairs where a HMP has a reservation

PLACE_BUILDING_AIRPORT

A building categorised as an airport

PLACE_BUILDING_HOTEL

A building categorised as an hotel

RESERVATION

A reservation for accommodation or transport services

RESERVATION_ACCOMMODATION

A reservation for accommodation

RESERVATION_TRANSPORTATION

A reservation for transport services

RESERVATION_TRANSPORTATION_AIR

A reservation for air transport services

RESERVATION_TRANSPORTATION_Bus

A reservation for bus transport services

RESERVATION_TRANSPORTATION_TRAIN

A reservation for train transport services

SERVICE_EVENT_ARRIVAL_DELAY

Event where arrival of something has been delayed to time later than the scheduled time

SERVICE_EVENT_DEPARTURE_DELAY

Event where departure of something has been delayed to time later than the scheduled time

SERVICE_EVENT_HEALTH_WARNING

Event where a health warning has been issued by a health information service provider

SERVICE_EVENT_SAFETY_WARNING

Event where a safety warning has been issued by a travel safety information service provider

SERVICE_EVENT_TRIP_CANCELLED

Event where a scheduled trip has cancelled

SITUATION_HMP_RESERVATION_REQUIRED

State of affairs in a particular context that requires action to be taken to make a reservation for a HMP

SITUATION_SERVICE_ACCOMMODATION

A state of affairs of an accommodation service provider in a particular context

SITUATION_SERVICE_ACCOMMODATION_AVAILABLE

A state of affairs of an accommodation service provider where accommodation is available in a particular context

SITUATION_SERVICE_ACCOMMODATION_PRICE_CHANGE

A state of affairs of an accommodation service provider where the price of accommodation has changed in a particular context

APPENDIX 3: ADDITIONAL ACADEMIC DOMAIN SPECIFIC CLASSES

SITUATION_REACTIVE_CANCEL_LECTURE

A state of affairs, or an infon, where it is true that a particular lecture must be cancelled in a particular context.

SITUATION_SERVICE_EVACUATION_REQUIRED

A state of affairs, or an infon, where it is true that evacuation of a particular building is required in a particular context.

SITUATION_SERVICE_LECTURER_UNAVAILABLE

A state of affairs, or an infon, where it is true that a particular lecturer is unavailable in a particular context.

SITUATION_SERVICE_LECTURE_CANCELLED

A state of affairs, or an infon, where it is true that a particular lecture has been cancelled in a particular context.

SITUATION_SERVICE_NOPARKING

A state of affairs, or an infon, where it is true that no vacant car parking spaces are available in a particular context.

SITUATION_SERVICE_TRAFFIC_DELAY

A state of affairs, or an infon, where it is true that a traffic route is congested and experiencing delays in a particular context.

SITUATION_SERVICE_FIRE_ALARM

A state of affairs, or an infon, where it is true that a fire alarm is ringing in a particular context.

COURSE

A course of study.