WeKnowIt

Emerging, Collective Intelligence for Personal, Organisational and Social Use

FP7-215453

D1.3

User Modelling and Dialogue Management

Dissemination level	Confidential
Contractual date of delivery	Month 33, 30-12-10
Actual date of delivery	17/01/2011
Workpackage	WP1, Personal Intelligence
Task	T1.3
Type	Report
Approval Status	Final
Version	1
Number of pages	49
Filename	D1_3_2011-01-11_v05_usfd_User_modelling

Abstract:

This deliverable describes the technologies developed to support user modelling and facilitate dialogue within the WKI personal knowledge management system.

co-funded by the European Union

History

Version	Date	Reason	Revised by
0.1	09/12/2010	Adding the introduction	USFD
0.2	30/12/2010	Incorporating additional contributions	USFD
0.3	10/01/2011	Remaining sections	USFD
0.4	11/01/2011	Context model appendix	USFD
0.5	14/01/2011	Internal review changes	USFD
0.6	17/01/2011	Final edit	USFD

Author list

Organization	Name	Contact Information
USFD	Simon Tucker	s.tucker@dcs.shef.ac.uk
USFD	Grégoire Burel	g.burel@dcs.shef.ac.uk
USFD	Vita Lanfranchi	v.lanfranchi@dcs.shef.ac.uk
USFD	E. Cano	A.Cano@dcs.shef.ac.uk
USFD	V. Uren	V.Uren@dcs.shef.ac.uk
TID	Manuel Escriche	mev@tid.es

Executive Summary

This deliverable describes the technologies and methodologies that have been implemented by WeKnowIt within T1.3, following on from T1.1 and T 1.2. The deliverable reports how technologies implemented by WeKnowIt meet the multi-user, multi-modal and multi-visual requirements of WeKnowIt applications. In particular, the deliverable discusses the contributions made by WeKnowIt for modelling users and their relations within the different components of the system such as the information uploaded to the system.

WeKnowIt is exploring Collective Intelligence as a construct of multiple layers of intelligence cooperating as a single whole. Personal Intelligence is one such layer and is primarily concerned with allowing individual users to provide knowledge and access intelligence using WeKnowIt applications.

The previous deliverables, D1.1 (Report on Interaction Model) and D1.2 (Personal Knowledge Management Systems), outlined the requirements of technologies to support the management of Personal Intelligence and how the technologies were developed to meet these requirements.

This deliverable describes the user modelling perspective adopted in WeKnowIt, with details on how user and context can be modelled to provide a focussed interaction. A multi-dimensional context has been defined using personal profiles and preferences to drive a dialogue in order to produce a system able to (i) focus the user activity on most appropriate actions and information required for the specific situation, (ii) reduce the device interaction complexity and (iii) provide the most appropriate information.

The technologies described in this document are necessarily generic in that they are intended to support any WeKnowIt application – discussions about how specific instances of such applications additionally address the requirements (for example in the case of the demonstrator applications) will be left for their respective deliverables.

Abbreviations and Acronyms

CSG	Consumer Social Group
DCTerms	Dublin Core Metadata Initiative Terms
ER	Emergency Response
E-WKI	The WeKnowIt system as seen by users
FOAF	Friend Of A Friend
GeoOWL	W3C Geospatial Vocabulary
OPO	Online Presence Ontology
PDA	Personal Digital Assistant
SIOC	Semantically Interlinked Online Communities
WKI	WeKnowIt

Table of Contents

List of Figures

1. Introduction

The aim of the WeKnowIt project is to explore Collective Intelligence – a form of intelligence derived from a number of different layers of intelligence acting and interacting with a common knowledge store. The Personal Intelligence layer contributes to Collective Intelligence through the actions and reasoning of individuals. These individuals may be part of larger organisations or notional groups but the focus of Personal Intelligence is how the independent actions of these individuals can contribute to the overall Collective Intelligence.

In such context, it is mandatory to provide an integrated model of the users that contributes to the collective intelligence and the knowledge created by them. Besides representing the user information, it is also important to model and learn behaviours and interests. Such behaviours can be used for simplifying the user experience (for instance, if the mobile phone battery of a user is running low, a priority should be established on the tasks to perform) or personalising the interface. In a similar fashion, interests tracking can also be used for improving the presentation of the knowledge available to the user. For instance, by knowing the user interests it is possible to help him to discover relevant information that suit the situation.

This document presents the different research contribution made by WeKnowIt in the area of user modelling. Particularly, it presents the different implemented approaches for modelling the WeKnowIt users, the context in which they are using the application and which information they are interested in at a given instant. Likewise, this document is divided in two different sections and an appendix. In the first section (1), the different approaches are presented by comparing them to the state of the art. First, the CURIO user model is introduced (1.1.2). Then, a contextual model of the user is presented (2). Finally, a method, based on attention detection techniques, is discussed (3). In the second section (3), we present the different services that are integrated in the WeKnowIt platform. Finally, the document is concluded and the future work outlined (4).

2. User Modelling and Profiling

A key requirement for a system to manage personal intelligence is that it should be able to support focussed interaction based on user and context details. This section describes how it is possible to build a model of the user and of the context and how this connects to the interaction model, the other models present in WeKnowIt and the various pieces of information stored as a part of that interaction.

1. User and Interaction Modelling

As mentioned in D5.3.1, the requirement for representing users in the WeKnowIt system is fulfilled by the development of a general ontology that represents the users of the system and their interaction within it. In this context, we created the Collaborative User Resource Interaction Ontology[1] (CURIO) (Ireson & Burel 2010) for modelling user attributes, preferences, system contributions and context.

In WeKnowIt, users are the central piece of the system. Accordingly, we have decided to design the CURIO ontology around the two fundamental concepts of *Users*, *Resources* and *Events*. For each use case, it is necessary to model how users interact with their environment. For example, in the CSG scenario, it means that users must be able to contribute to events happening during their travelling or subscribe to interesting information. In the ER scenario, users also participate in events or submit documents that must be associated with them. In general, other important features include the relations between different users of the system and, the storage of personal information such as names, location, etc.

The following sections discuss the contribution of the CURIO user model within WeKnowIt and how it compares and extends existing work. In the first section, we present the different work that has been done for modelling user personal information and user generated content. The second part focuses on the description of CURIO ontology from a user modelling perspective. Finally, we describe its current state of integration and conclude.

1.1.1. State of the Art

Since the emergence of Web 2.0 platforms, the place of user information within applications is becoming a priority (Oreilly 2007; Alexander 2006). In such context, many models have been developed during the past years for fitting the requirement of user modelling (Kobsa 2007). Such models

[1] CURIO, http://purl.org/net/curio/ns#

have generally focus on either the user preferences and attributes or the place of such users in a community (for instance, their contributions to a community).

As the WeKnowIt platform has been developed as a Web platform, we focus our analysis on Web models (or more specifically Web 2.0 models) of users information. In particular, a special attention is taken to ontological user models.

User Modelling

As WeKnowIt is designed to be an intelligent application, the user model is integrated into a global information model based on semantic technologies. In the area of user semantic modelling, sets of different ontologies have been created during the recent years (Brickley & Miller 2005; Sauermann et al. 2007). In particularly three different models have been prevalent: FOAF[2] (Friend of a Friend) (Brickley & Miller 2005), vCard RDF[3] and PIMO[4] (Personal Information Model) (Sauermann et al. 2007).

The FOAF ontology has been developed since 2007 and is the most used ontology on the Internet. The ontology covers a wide range of user attributes such as the user name, gender, location, social relationships, age, and created documents. It also includes some extended functionalities like email addresses, topic of interest, social network accounts.

The vCard ontology is a direct RDF mapping of the vCard specification that has been around since 1995. The vCard standard has been created for holding personal user information in the context of personal information exchange. In this context vCard RDF supports more descriptive information of user attributes compared to FOAF such as the user full address. However, this ontology remains less used compared to FOAF and does not support relationships information or user contributions.

The PIMO ontology was created for the European project Nepomuk for modelling user information within the Nepomuk semantic desktop (Sauermann et al. 2006). Compared to FOAF and vCard, it supports the information contained in vCard and FOAF and supports additional relations such as Tasks and Events. Unfortunately, the Nepomuk ontologies have received little interest outside the project and thus provide a limited ability to share information on the Internet.

[2] Friend of a Friend, **http://xmlns.com/foaf/spec/**

[3] vCard RDF, **http://www.w3.org/Submission/vcard-rdf/**

[4] Personal Information Model, **http://www.semanticdesktop.org/ontologies/pimo/**

User Generated Content Modelling

In the context of user generated content modelling, little work has been done for standardising this type of information. However, the Semantically-Interlinked Online Communities initiative[5] (Figure 1: The SIOC ontology) (Bojars et al. 2008) has produced a standard that integrates with the FOAF ontology. The SIOC ontology provides a general framework for managing user contributions within a system. For instance, users can post information in the form of forums or posts on a blog. Such activities are modelled as a core feature of SIOC.

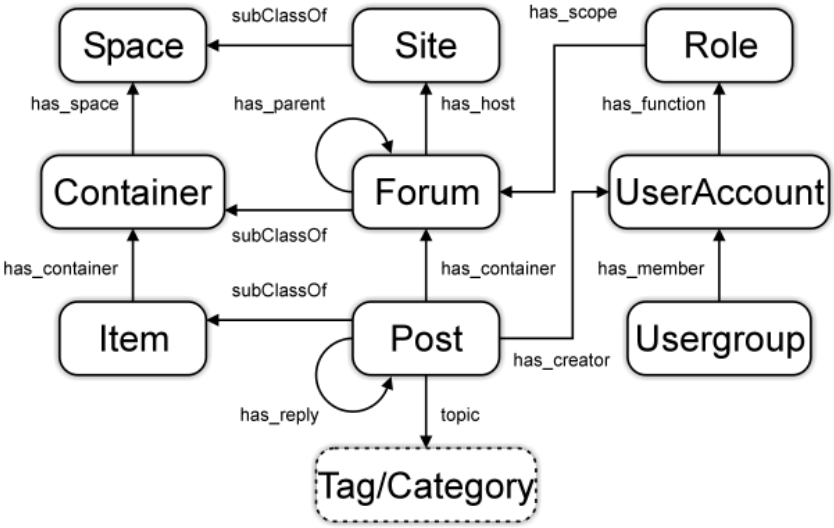

Figure 1: The SIOC ontology

The general model provided by SIOC remains too generic in the case of WeKnowIt. For instance, SIOC is unable to support the concept of events without extension. Such kind of modelling is mandatory in WeKnowIt.

As a consequence, it is a good idea to base the WeKnowIt user model on the SIOC ontology since it supports generic community features and is based on the FOAF ontology.

1.1.2. The CURIO User Model

One of the main advantages of reusing popular existing ontologies is to provide interoperability, i.e. the easy integration of WeKnowIt information both into and from external systems. For instance, it simplifies the integration of external social network information into the system.

The CURIO ontology (Figure 16: The CURIO Ontology) supports all the features of FOAF and SIOC. It extends SIOC by reusing and connecting a set of ontologies together. CURIO can be considered as an *aggregative*

[5] SIOC, http://rdfs.org/sioc/spec/

ontology since it aggregates and connect many concepts from different standard ontology in a formalised way. This vocabulary reuses most of its concept from the SIOC Vocabulary but incorporate many classes and properties from other ontology such as Dublin Core[6] (DCTerms), GeoOWL[7], Common Tag[8] and the Online Presence Ontology[9] (OPO). The ontology is based around the concept of *Resource*. A *Resource* is a specific class that holds a user generated content. Currently different class exists. Particularly, the *Document* and *Event* classes enable the combination of virtual events with a real one.

Besides the SIOC and FOAF concepts, CURIO defines 36 additional concepts and 18 new relations that enable the connection of users with resources and the storage of personal information. The ontology is divided in four different namespaces:

- **http://purl.org/net/curio/ns#** - CURIO Core Vocabulary Namespace
- **http://purl.org/net/curio/annotations#** - CURIO Annotations Vocabulary Namespace;
- **http://purl.org/net/curio/documents#** - CURIO Documents Vocabulary Namespace;
- **http://purl.org/net/curio/resources#** - CURIO General Resources Vocabulary Namespace.

Each namespace provides additional features to the core ontology such as specific type of resources.

CURIO connects *Users* and *Resources* together in order to model the interaction patterns between the information contributed by a user, their characteristics and their relationships. All these relations are based on the basic SIOC and FOAF classes such as *User*, *Item* and *Container*. The following paragraph describes the User, Resource and Event classes. For more information about the other concepts of the ontology, you can read the annex of this document or the online documentation of the ontology (**http://purl.org/net/curio/ns#**).

User

In CURIO, users are defined as a subclass of FOAF and SIOC users. This model allows the usage of the entire user properties defined in FOAF (as

[6] DCTerms, http://dublincore.org/documents/dcmi-terms/

[7] GeoOWL, http://www.w3.org/2005/Incubator/geo/XGR-geo-20071023/

[8] Common Tag, http://www.commontag.org/Specification

[9] OPO, http://online-presence.net/ontology.php

mentioned previously). The CURIO user model is also extended with additional concepts. A user can participate in an event (for instance, an Incident). A user can contributes to information within the system; such type of relation is a creation relation associating a *Resource* with a *User*. A user can also subscribe to information feeds that describe the evolution of a Resource (for example, the user wants to know when a sub event is occurring). A user can also be identified as connected (or not) or available thanks to the integration of the Online Presence Ontology (such type of information is useful in the case of the ER scenario). Finally, users can be associated to access policies for restricting the access to a particular resource.

For more information concerning the User class, you can refer to the annex of this document (Figure 16: The CURIO Ontology).

Resource

A *Resource* can be defined as "anything that is added to the system by an individual". In general, a *Resource* is created by a *User* and owns a set of general properties. Besides the ownership relation, creation date and so on. A Resource can be annotated (Figure 2: CURIO Resources Annotations) and associated with events (Figure 4: CURIO Event Model) as evidences. Such pattern enables users to discuss, modify, rate and tag any information within the system. Because of the nature of WeKnowIt, CURIO has a set of default Resources such as audio information, textual documents and images (Figure 3: CURIO Documents Types).

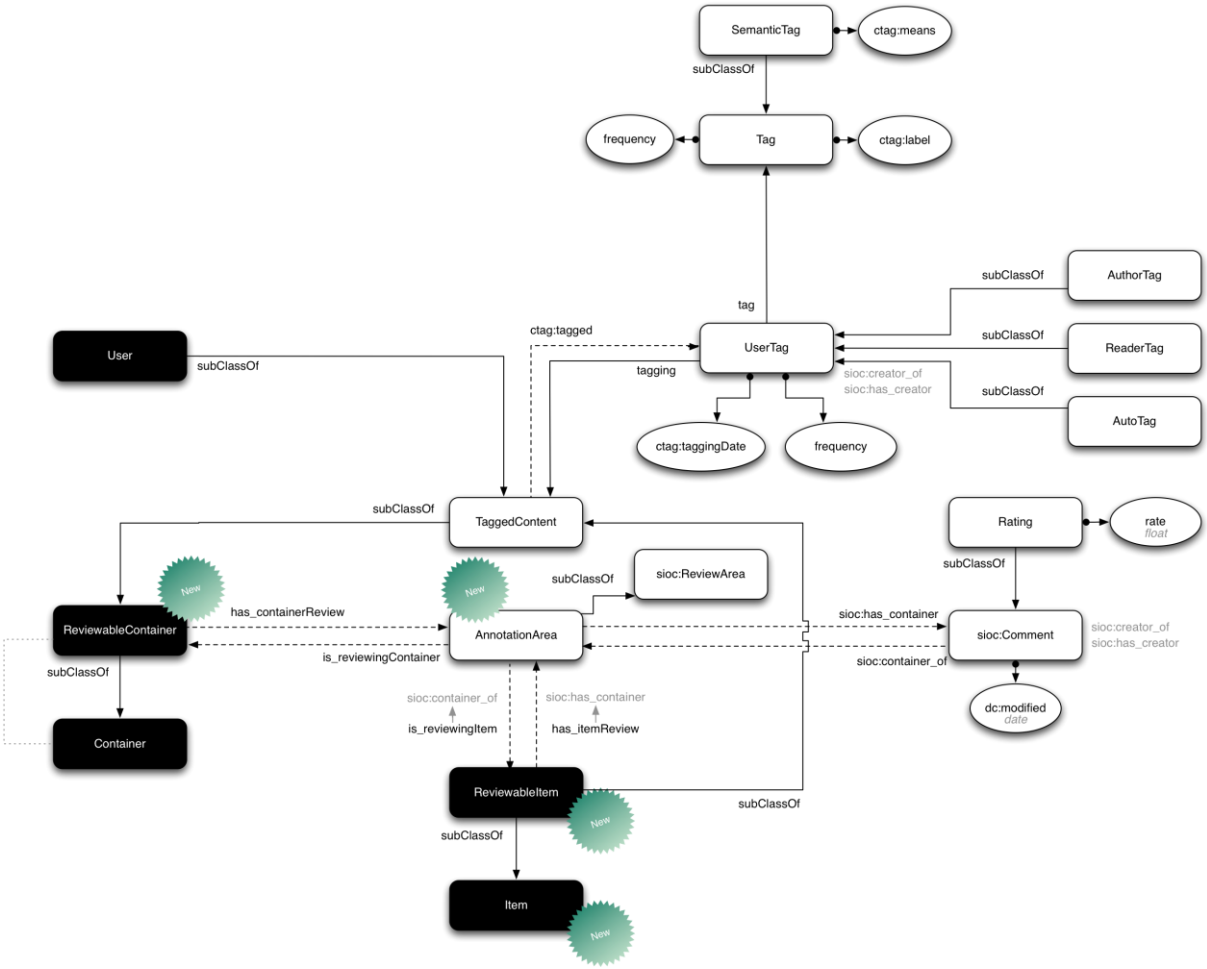

Figure 2: CURIO Resources Annotations

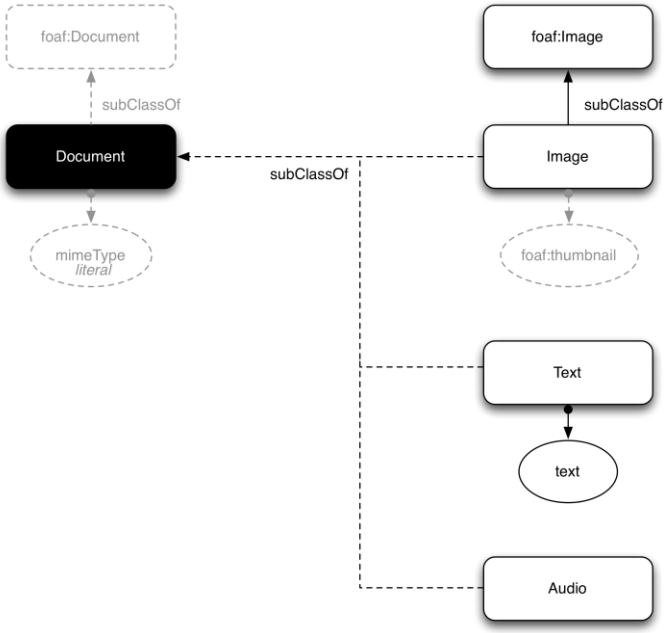

Figure 3: CURIO Documents Types

Events

In order to build a user and context model that relates to the whole WeKnowIt system, it is also important to be clear about the different kinds of *events* that are modelled by WeKnowIt and how this model connects to these events (Figure 4: CURIO Event Model). There is a generic event itself that occurs at some fixed point and unfolds over a period of time. For example, in the ER case study an event could be a tree falling over and blocking a road. The event starts at the point when the tree falls over and continues over time until the emergency is over.

There are then other, more specific, events that are connected to this generic event in the context of WeKnowIt. These are such events as an individual making a record of the emergency, for example taking a photo of the tree, uploading the image to WeKnowIt and then annotating the photo. Each of these actions could also be considered to be an event but these events are inherently associated with the generic tree falling event. Additionally it should be noted that the generic event effectively occurs outside of WeKnowIt (since the tree falling over is unrelated to WeKnowIt, unlike the specific events of uploading information to the system).

As reported in D1.2 and D5.2.1 given this structure of events, one approach to modelling the interaction between users and the WeKnowIt system would have been to extend the core Event Model (Scherp et al. 2009) to account for the hierarchical nature of the events. The Event Model is, however, a generic model for representing real world events and, as highlighted above, makes no reference to the specific nature of WeKnowIt. Therefore, a problem with extending this model to account for the types of interaction found in WeKnowIt would be that the model would get too detailed and would therefore lose its expressive power.

The approach taken, therefore, is to integrate a simple event model to the CURIO ontology. This approach means that the interaction between the user and the information processed and stored, by WeKnowIt can be modelled outside of the core Event Model.

In this context, the integration of a simple Event model within the CURIO ontology facilitates the annotation of events with participation pattern or, more generally, the CURIO annotation modules shared between each *Resource* of the system. By making an *Event* a particular type of *Resource*, it is possible to manage the personal relationships that a *User* has with an *Event*.

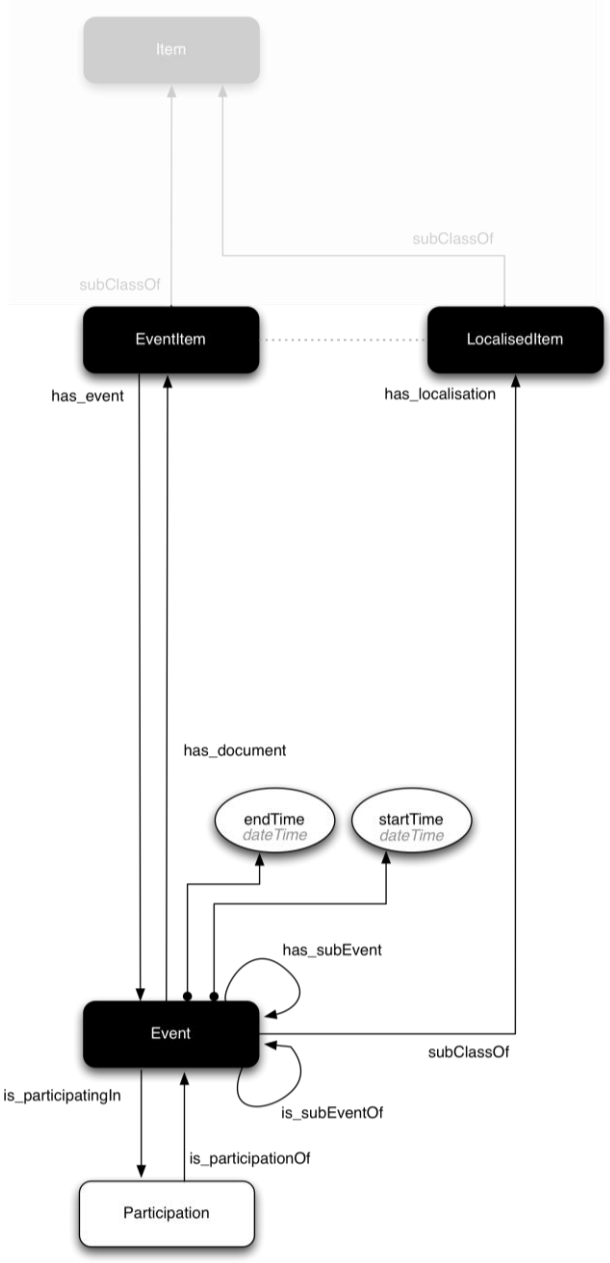

Figure 4: CURIO Event Model

The *Event* model also supports the concept of evidence. This concept means that any *Resource* can be connected to an *Event* as evidences. For example, a picture of a fire can serve as evidence during a fire incident.

1.1.3. Implementation

CURIO is written in OWL/RDF and divided in different namespaces. The full ontology is available online at the following address: **http://purl.org/net/curio/ns#**. CURIO is integrated in the WeKnowIt Common Model. Such implementation is described in D6.3.

1.1.4. Conclusions

The WeKnowIt user model is integrated within the CURIO ontology. It supports a large set of personal attributes and also allows the identification of the resources a user is contributing to. CURIO represents the knowledge model of WeKnowIt and it is a continuous evolving work that can adapts to the requirements and progress of the system. It is modularity allows the integration of additional personal information by reusing standard ontology.

2. User Context Modelling

Modelling the user context allows the WKI applications (more specifically the mobile applications) to fit themselves to the users working environment. This has clear applications to a personal intelligence in general and towards the two use cases in particular. Clearly, for personal intelligence to act effectively it must not only be able to represent each individual user but also, to some extent, model the situation in which the user finds themselves. Thus the personal intelligence layer acts to understand the user and their current environment in order to allow that user to act efficiently and effectively.

In terms of the demonstrators the context modelling in the personal intelligence layer as well as how asking the question of how context should be modelled there is a further question of how to make use of that context model. Thus modelling context is important but of equal importance is how the implications of that context model should be included within the interface and application as a whole.

The following sections describe the contributions made as part of the personal intelligence work package in the domain of context modelling. We start by outlining the state of the art, describe how context was modelled in the WKI system, highlight examples of how this context was used in the two demonstrators, describe future applications of the context model and conclude.

2.1.1. State of the Art

Before considering the state of the art of context modelling and context aware applications it is useful to define what context is. Clearly "context" is a necessarily vague construct and can overlap with user modelling to some degree – the context of the user may account for or take account of the user profile when making contextual decisions. Thus for example the user may be in a newcomer context when using an application for the first time.

Schilit (1994) defines context across three axes:

- Computing Context: Network Connectivity, Communication Costs, Bandwidth etc.
- User Context: User profile, location, social situation etc.
- Physical Context: Lighting, Noise Levels, Traffic Conditions etc.

Chen and Kotz (2000) extend this definition by including an additional area:

- Time: Time of day, season of the year etc.

Furthermore, they include the notion of context history – which the context model should additionally account for past contexts of the user. Thus it can be seen that the context model must account for a large number of elements when building a model of the current context of the user.

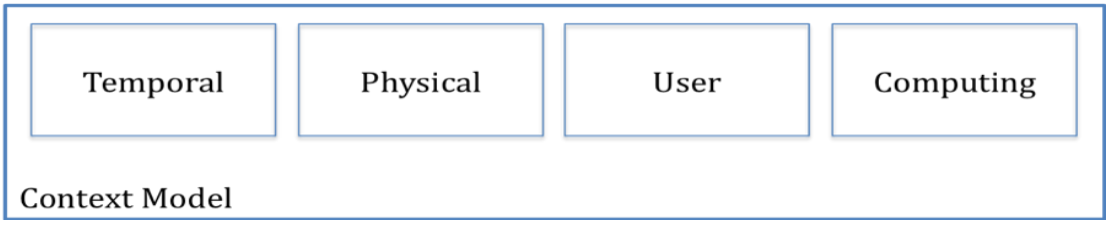

Figure 5: Combinatorial Context Model

Typically information is collected from a variety of sensors – the type of sensors used and information gathered depending on the type and level of context being sought.

Thus the process of modelling context can be broken down into three sections. Firstly how to extract sensor information and from which devices these should be extracted. Secondly this raw sensor information must be analysed and transformed in order to provide input data for a modelling process. Finally the input data must be reasoned upon in order to derive the resulting contextual information. We deal with each of the these processes in turn below

Sensor Information

Numerous sensors can be used to derive contextual information. In a classic study (Bennett et al. 1994) installed infrared receivers around an office building and built a small transmitter into each workers name badge. The badge then transmitted the user id at regular time intervals. This process allowed the users location to a) be identified at all times and b) recorded so that the users could retrieve a diary of their locations over the course of a day or a week. This process allowed the company to implement a context aware phone routing system which routed incoming calls to whichever phone was closest to the intended recipient.

A similar location appropriate approach is taken by (Marmasse & Schmandt 2000). Here the user creates contextual reminders – a note for

a given place and time. When the system detects that the current context is met it delivers the appropriate reminder. Further work built a device that had a built in clock and GPS sensor in order to determine if the time and place is right for the contextual reminder.

The majority of work has focused on locational context but context should account for more than just location (Schmidt et al. 1999). GPS or GPSa is a method for determining the location of the user when outdoors but modern smartphones, as well as offering GPS sensors, have a multitude of other sensors which can combine to define a more rounded picture of the user context.

Transformation of Sensor Information

Once the raw sensor information has been collected it must be transformed into a format that is usable within context models. For example a sensor can measure your location but this information must be transformed in order to determine if you are in the vicinity of a given location or your overall distance from a given location. Depending on the complexity or the type of transformation required the calculations can be done on the device or in a server environment. Furthermore some sensor information may only make sense as part of a series, so again some integration over a range of data may be required in order to fully interpret the sensor information.

Modelling Transformed Sensor Information

The majority of context modelling described in the literature focuses on a single aspect of context and is, therefore, easier to model. For example there is no need to explicitly model for different locational situations in the active badge system (Roy Want et al. 1992) since the sensor information fully determines the resultant contextual decisions.

Schilt (1994), however, describes a simple language for building context models which integrates over a number of sensors to produce a resulting context.

```
Coffee Kitchen arriving "play -v 50 ~
schilit *□/sounds/rooster.au" attention "emacs -display $NEARESTHOST:0.0"
```

Figure 6: Rule Based Context Model Approach

Thus in this example the system will play a certain sound if the user is in a given location. In this way application developers are able to manipulate context in a straightforward way and incorporate context within systems and applications without dealing with the process of understanding how to compute the context itself.

This approach is appropriate in situations where context can be easily specified but there may also be situations where it is not straightforward

for application developers to define context using this rule based notation. Thus whilst rule based approaches may be useful when application developers have clear idea how context should be defined, they are of less use if developers are unsure how best to combine sensor information into an overall picture of the user context.

2.1.2. Implementation

Given the above analysis we implemented a light weight context model that was centred on the notion of receiving both continuous and discrete sensor information as required and allowing the user of the service to derive the context as required from these sensor inputs. Our implementation consists of two parts. The sensor information is provided by either a mobile or desktop client which passes information either at regular intervals or at the time of requirements to the WKI system in order to derive the correct measure of context. The context model is then implemented as a simple machine learning process which can integrate the information coming from the sensors and deliver the appropriate measure of context.

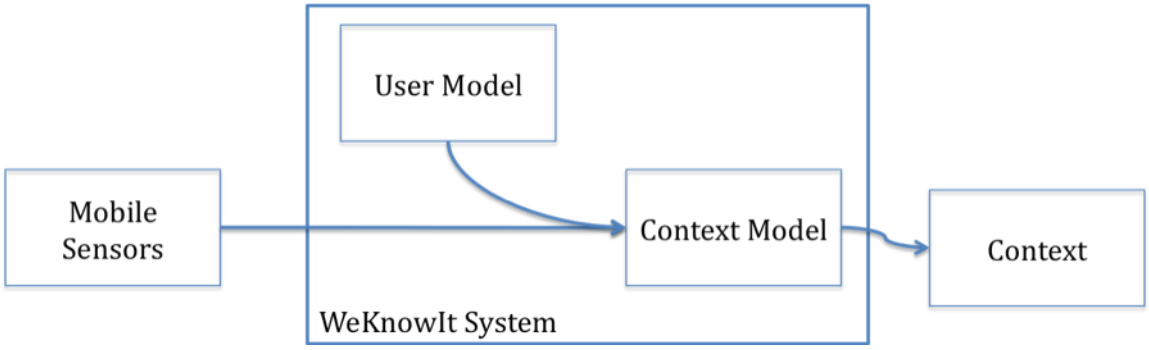

Figure 7: Context Model

Thus the sensor data is passed via the internet to the WKI system which then records the latest value of the sensor information in an internal database. The sensor information consists of a sensor name and a given value – there is no interpretation of the value or restriction on the range of values that a sensor can collect at this stage.

The user then acquires context by defining the sensor information required (and potentially updating the system by defining the current values of these sensors) and the selecting the context model to be used. This allows the system to load the model which is stored within the WKI system, determine the appropriate sensor values from the ones provided at the request time, the last values stored in the database or from default values if neither are provided. The model is then passed each sensor value and from these values determines the relevant context.

The model itself is represented as a decision tree. Each node in the tree corresponds to an interpretation of sensor data. Thus the model may process all or some of the sensor data in order to determine the relevant context, depending on the construction of the model.

Context models can therefore be constructed in two ways. Firstly, for complex connections between sensor data and context the user can provide some training data and the intended context and have the model built for them. Alternatively models can be constructed by hand by joining together nodes and their appropriate outcomes. Once constructed in either fashion the models can be saved to disk and recalled at a later date in order to interpret the context on the basis of the contextual model.

2.1.3. Example

Example 1: Limited Access

During times of emergency it is desirable to ensure that information from citizens or emergency workers at the scene are able to have functional access to the WKI system. It would be undesirable in such situations for the user to have to spend a lot of time or use a lot of bandwidth when providing information to the WKI system. Thus it would be appropriate for the system to ensure a pathway whereby users provide information with a minimal impact on the amount of time taken to complete the information upload process.

To account for this we created a "low-bandwidth" context. This model determines whether the user is in a situation where the preservation of their bandwidth is critical. If this context is detected then when providing information the user has a simplified path which focuses on getting the most out of the information they send as quickly and with less bandwidth as possible. Thus the interface shown to the user in this circumstance does not contain many images and the process is shortened to focus on annotation and provision of information. The context model to detect such situations is shown in Figure 8: Low Bandwidth Context Model.

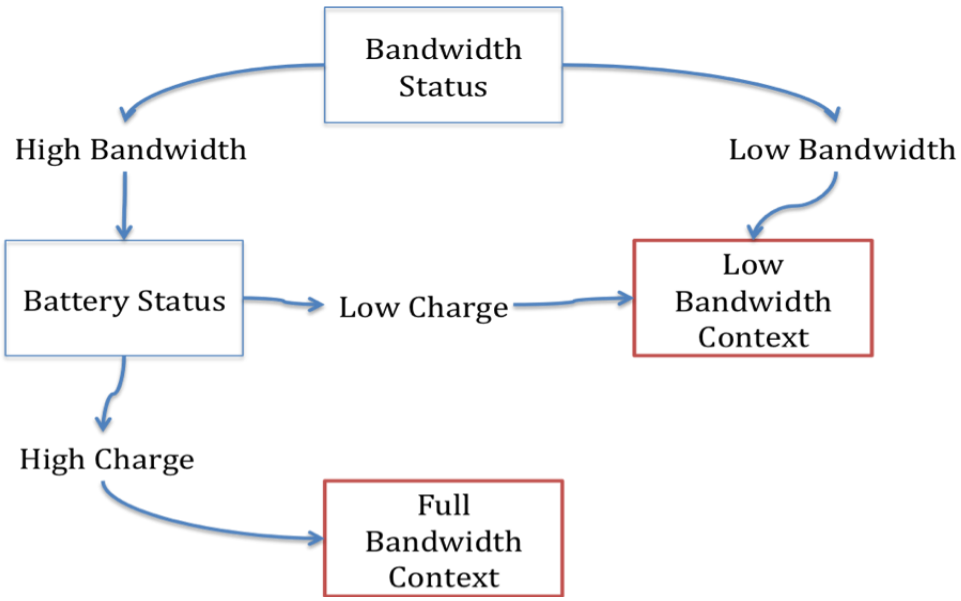

Figure 8: Low Bandwidth Context Model

Thus the model has two principle nodes. One receives information about the battery level of the upload device – if the battery level is low then the

context is activated. If the battery is fine but the system detects that the users current bandwidth is lower than normal then again the context is activated. If neither of these conditions is met then a normal context is detected and no change is made to the user experience.

Clearly this is a relatively simple context model which can be specified manually through the combination of different contexts. If, for example, a more fine-grained measure of battery life was required the continuous sensor mode could be used to assess the interaction between the level of battery life remaining and the actual time left for use of the device. This could then be incorporated into a multi-platform node within the context model with a default setting to allow for a range of different devices.

Example 2: In Vicinity Context

The vicinity scene context combines a node that understands the role of the user and their current location. The model is shown in Figure 9: In Vicinity Context.

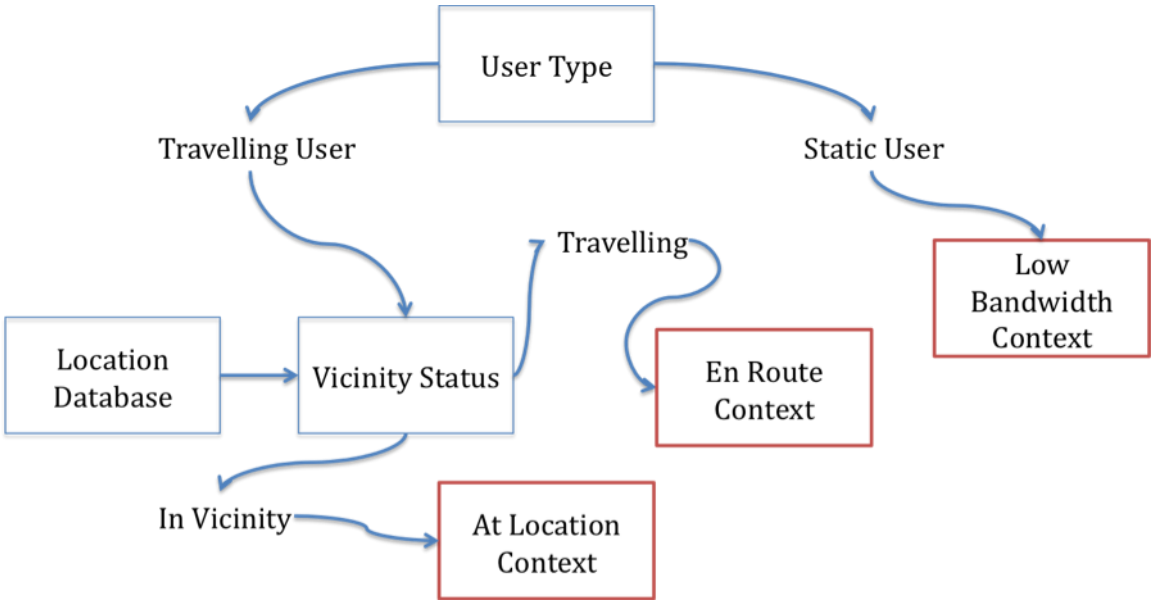

Figure 9: In Vicinity Context

This model configures the mobile interface to support the most likely action of the user when using the mobile WKI application. It examines the role of the user and determines if they are of a user type that operates differently when in the vicinity of a given location. If the user does operate in such a capacity then the context model asserts if the user is in the vicinity of a given location and then alerts the user interface to this and so the user interface can be tailored accordingly.

For example, the consumer social group application could offer a different set of options when the user is travelling between points of interest to that when the user is at a point of interest. So when travelling between the interface could offer a view which allows the user to navigate to the point of interest and then offers a view which focuses on taking images, or information about the point when the user arrives at the point.

In the emergency response scenario the user interface can highlight the map functionality when the user (ideally a Forward Liason Officer or emergency response worker) is travelling to the scene of an incident. When the user arrives at the scene of the incident the interface can highlight the information upload option.

Thus it can be seen that the context model allows the interface designer to abstract away from having to interpret the sensor information and resolve the context of the user and focus on how that context should be interpreted within the user interface.

2.1.4. Future Work

Future work will explore how context can be incorporated dynamically within the user interface. Currently the use of context is a one-time operation – the interface measures the context and configures itself towards that context. An alternate approach is that the interface should dynamically alter itself to account for user context. A key example of this approach seen in both of the demonstrators is navigation. When the user is in transit between two locations they may want to use a map functionality to guide their movements between the two locations.

Traditional map interfaces focus on the users location however, rather than considering their context (i.e. that they are travelling to a given location).

An approach for clarifying this would be to warp the map view so that the user has fine detail around their current location with the level of detail diminishing outwards towards their intended location. Thus the user is able to see the context of their journey – i.e. their current location in the context of their destination – which allows them to make more informed decisions as to what routes to take and whether they need to move away from the intended path (Figure 10: Distorted Map Example).

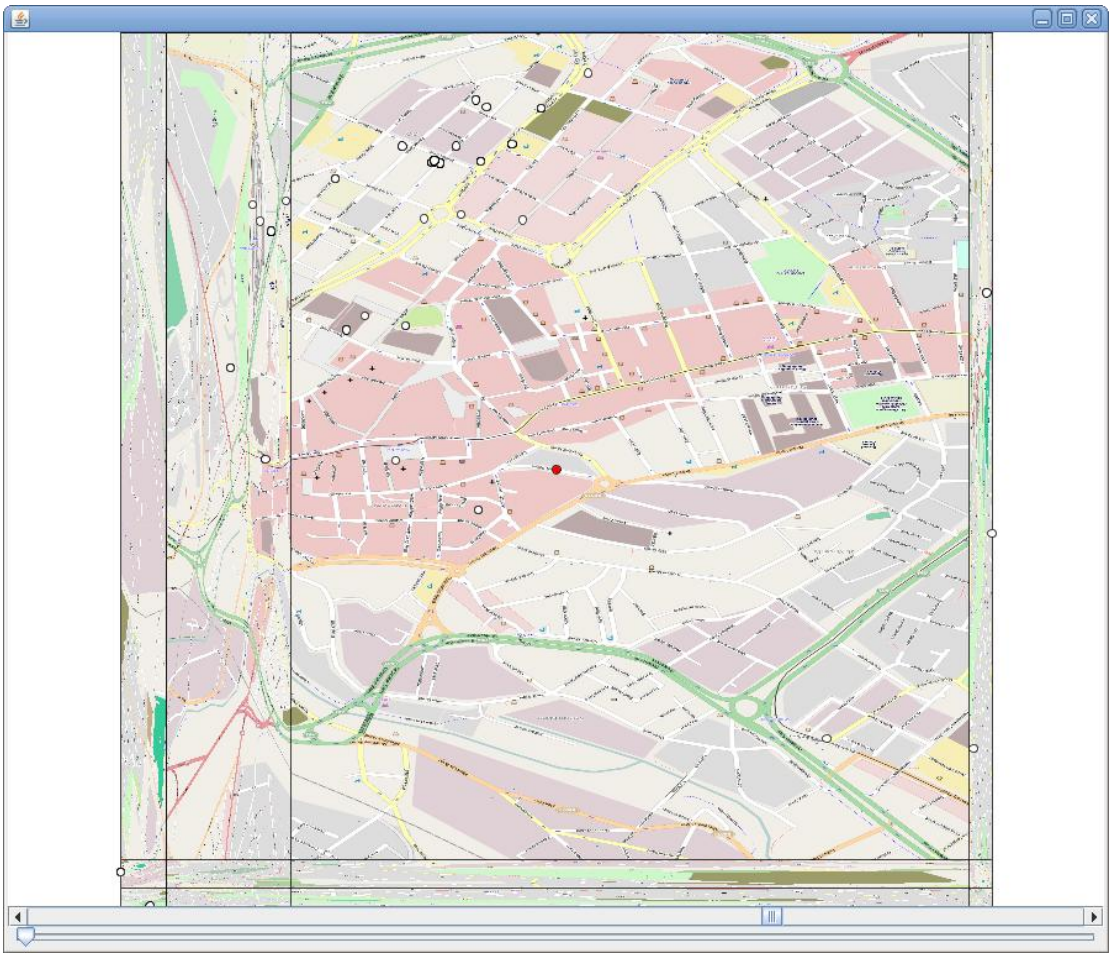

Figure 10: Distorted Map Example

2.1.5. Conclusion

We have described how context is used with the WeKnowIt system. Context is a complex, multi-faceted concept and as such is difficult to model. In WKI we use a simple decision tree model to infer context over the input from sensors on the mobile device and from information held within the WKI knowledge store. Context can then be utilised in the applications by tailoring the interface so that is of increased relevance to the user context. This, in turn, increases the effectiveness of the personal intelligence layer as the user interaction is focused and simplified thereby encouraging the provision and use of information within the system.

3. User Attention Modelling – Attention-Streams

Contrary to most of the existing online communities, WeKnowIt is not a general-purpose community. This particularity reduces the efficiency of tracking user interests within the system since users use the system only when they need it. Such behaviour is particularly significant in the case of the Emergency Response scenario where users only use the system once in a while, when an emergency occurs. For instance, a typical use would be to first read news on local news websites before going on the

WeKnowIt application for checking additional information. In this context, the external user behaviour gives information on what they are the most likely to be interested in the WKI application. Relying just on internal user behaviour would make impossible to understand the users real needs.

As a consequence, a methodology called Attention-Streams has been developed for tracking user interests. Attention-Streams is different to existing approach as it profiles users within and outside WeKnowIt thanks to a simple script that the user can install in his web browser. The proposed model is based on the Attention theory (Treisman & Gelade 1980; Neumann 1996) and the APML ontology[10]. The approach uses the indirect analysis of the user activity for understanding the information he is attending to. Then, such information can be used for representing the user interests and consumed in a recommender system.

The following section presents the different approaches existing for representing user interest on the Web and introduces some key concepts behind attention theory. Next, the Attention-Streams model is presented and a recommendation system is introduced. Finally, before concluding, we present the current implementation and the application usage.

3.1.1. Related Work

A lot of research has investigated the problem of user interest modelling (Burke 2002; Daoud et al. 2007; Adomavicius & Tuzhilin 2005). However little research has focused on the creation of evolving interests using implicit and cross-domain user analysis.

User Interests Modelling

One key area which WKI addresses is the problem of cross-domain user profiling. Most of the existing systems have been focused on analysing user within their own platform. A recent exception to such behaviour is Facebook[11] with the publication of it Social Graph API [12] which enable Facebook to track users "likes" across domains. However, such system needs the explicit interaction of the user in order to register an interest. Some research has tried to perform similar task using search history (Daoud et al. 2007) but remains limited to very particular use cases.

Some ontologies have been developed for modelling the user interests. For instance, FOAF (Brickley & Miller 2005) supports a very simple model of user interests. However FOAF doesn't have any way to record the weight of such type of interest and is not suitable for modelling evolving interests. A more recent initiative created the Weighted Interests Vocabulary[13]

[10] APML, **http://apml.areyoupayingattention.com/**

[11] Facebook, **http://facebook.com/**

[12] The Facebook Social Graph API, **http://developers.facebook.com/docs/reference/api/**

[13] The Weighted Interests Vocabulary, **http://smiy.sourceforge.net/wi/spec/weightedinterests.html**

(**http://smiy.sourceforge.net/wi/spec/weightedinterests.html**) for modelling user interests along different dimensions such as temporality and importance.

Attention Modelling

As explained, WeKnowIt needs to know the users interests as they occur within and outside the platform. Unfortunately, the existing techniques try to model the interests directly from the user input (such as Facebook) or through a controlled environment (closed community). In this context, it is necessary to understand where the users interests are focused, or more precisely where is their attention. Such activity is the process of selecting particular information within an environment while ignoring other. In other words, when a user looks at a document, it is because he has selected to ignore other information and decided to concentrate on a particular one. Such selection can be used for determining what the user wants to look in the future and in a higher degree to what are his current interests.

Contrary to interests analysis, attention analysis focuses on the analysis of the user interaction instead of the relation between the user and the current information itself. In this context, a model of the attention represents generic features of the user interests that can be followed between different environments easily since this information are more like interests stimulus rather than interests themselves (Lavie et al. 2004). Such characteristic can be differentiated from interest models since Attention is by definition limited in time and remains present only if the stimulus is still present over time.

The Attention Profiling Language[14] (Figure 11: The APML Ontology) is a rare attempt to represent attention in a standardised way. APML represents user attention on the form of weighted concepts and profiles that holds a set of such concepts given a particular situation. APML is technically very similar to the Weighted Interests Vocabulary but it is an XML format and it is targeted as a representation of attention. The corresponding ontology is called APML RDF. When we designed Attention-Streams, the Weighted Interests Vocabulary did not exist so Attention-Streams is modelled on the top of APML RDF.

[14] APML, **http://apml.areyoupayingattention.com/**

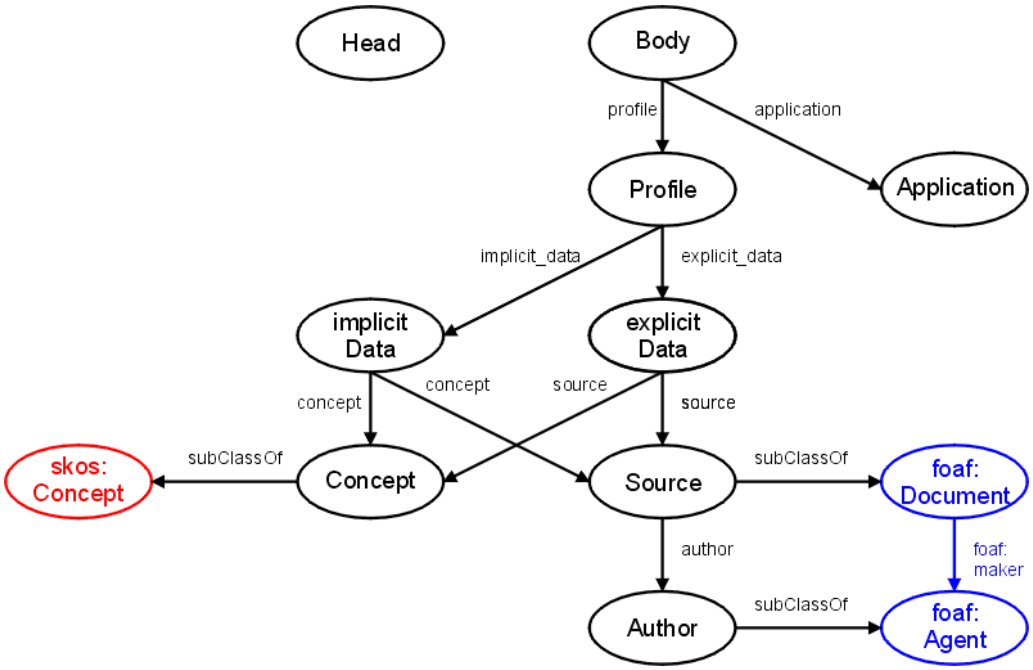

Figure 11: The APML Ontology

3.1.2. Attention-Streams

Attention-Streams can be divided in three different components: A model, a tracker and a recommender. The first component represents the data model used by the profiling system. The tracker generates the model from the user activity. Finally, the recommender consumes the model for recommending and personalising the user interface given the current user activity.

In the rest of this document we define an Attention-Stream as the continuous flow of interrelated information accessed through the Attention of an Agent (or User).

The Tracker is a small application that generates the APML model in real-time. Basically, the system analyses each pages the user is visiting by creating a concept model of each pages. Then, this model is submitted to a server and merged with the current user model for generating a current representation of the user attention. Finally, the created model is stored using the Attention-Streams model and ready to be consumed by a recommender system.

Before presenting the implementation, we need to introduce the model behind Attention-Stream and explains how the attention is extracted and then consumed within the WeKnowIt system.

Attention Model

As introduced in the previous section, a simple model of attention can be used for describing the user interests using tags. The APML ontology has

been designed for modelling user interests along activity profiles using weighted tags. Accordingly, APML is used as our main model for Attention-Streams. In our model, we refer to two different concepts: Attention Tags and Attention Space.

Attention Tags (or an APML Concept) represents the concepts that specify the degree of attention of a given user. Each tag has an associated weight that defines how much the concept is relevant in the current situation. Each document that is accessed by a user has an associated concept that can be used as the user's Attention Tags. The tags are automatically weighted as the user is browsing documents (the weight is set to 0 when the user is not interested in a given concept and 1 when is completely committed to such concept at a given time).

An Attention Space (or an APML Profile) is a set of Attention Tags that represents the user attention at a current time. All the attention tags with their weight above zero represent a tag cloud that defines the user attention. It is the projection of cross-domain concepts within an Attention (Figure 12: An Attention Space).

Figure 12: An Attention Space

Attention Extraction

In order to create the user Attention Space, it is required to extract the concepts of the web documents accessed by the user and to merge the current concepts with the previous attention tags of the user. The tags need to be weighted given the previous user attention and the current document.

The first step is to extract the concept of the document. Such task requires the identification of the main concepts of the documents through automatic techniques. For instance, machine-learning techniques can be applied for extracting categories from textual document. For example,

OpenCalais[15] applies such type of techniques for extracting tags from web documents.

The second step is to weight the tags given the previous weight of the concept within the user Attention Space. This step requires analysing how much a tag is relevant given the previous attention (Does it match the current attention or is it new?) and what should be its weight. The first step is called Attention Tag Similarity while the weight is calculated using Attention-Range Calculation.

The Attention Tag Similarity is the relatedness between two tags. The current method is done using the Google Similarity Distance (Cilibrasi & Vitanyi 2004). This method is preferred over other techniques such as WordNet since concepts may not exist within WordNet. The current similarity is calculated using an index of 3M articles from the English Wikipedia. Given the similarity value between two tags, it is possible to calculate the affinity of a new concept with an Attention Space: the correlation between a given tag and a set of tags can be described as the affinity between the relatedness of a tag with each tag of a tagset.

For each tag, a tagset affinity is calculated; such affinity can be combined with the Exponential Moving Average (EMA) of the new tag with its previous value for calculating the weight of the tag ($\dot{\nabla} val$). Accordingly, for a new tag *t* and a tagset *T*, the weight (val) of *t* is given by the following formula:

$$val(t,T) = affinity(t,T) + \dot{\nabla} val(t,T)$$

By calculating each weight of each concept, it is possible to determine the value of the associated attention and update the status of the user Attention Space thus generating a real-time model of the user interest.

By updating the Attention Space of the user each time she accesses a different web document, it becomes possible to create a real-time model of the user interest (Figure 13: The Evolution of Attention Over Time for different Tags).

[15] OpenCalais, http://www.opencalais.com/

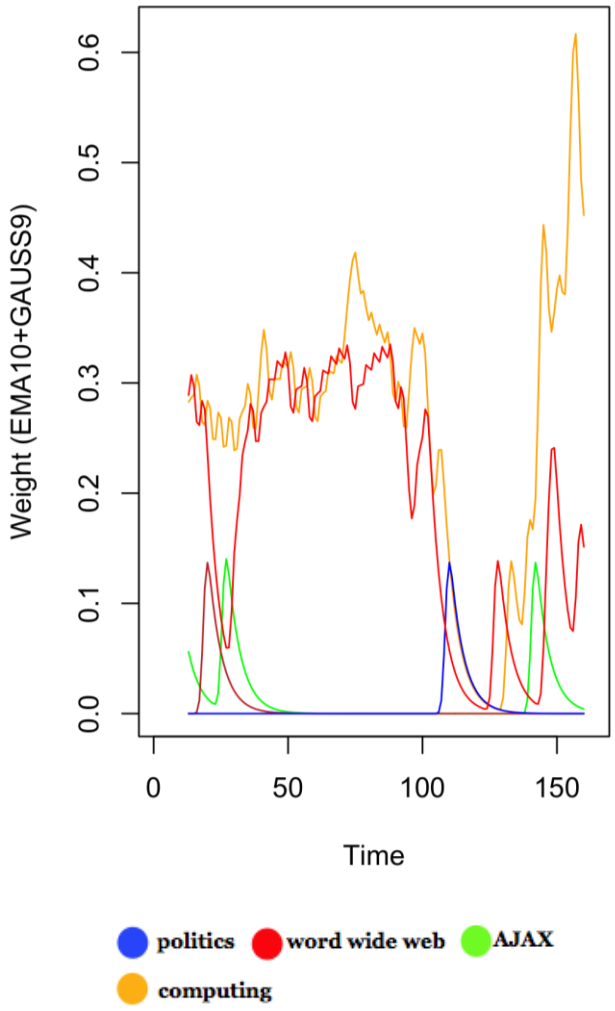

Figure 13: The Evolution of Attention Over Time for different Tags

Recommender System

After creating the model of the current user interests, it is important to use it for presenting content to the user in a more personalised way or to help him to discover new relevant knowledge. Thanks to the simplicity of the model, any application can consume the attention model of the user for presenting interesting information. Such recommendation depends on the targeted system (ER or CSG scenario). The following section presents an implementation of a recommender system on the top of Attention-Streams.

3.1.3. Implementation

The implementation of the Attention model is divided in three steps (Figure 14: Attention-Streams Architecture). First, a script collects the attention tags of the user given the current document he is browsing (1,2,3). Then, the collected information is submitted to a server that

analyses the tags and merges them in the user attention model (4). Finally, the attention information is used for recommending interesting information to the user (5,6).

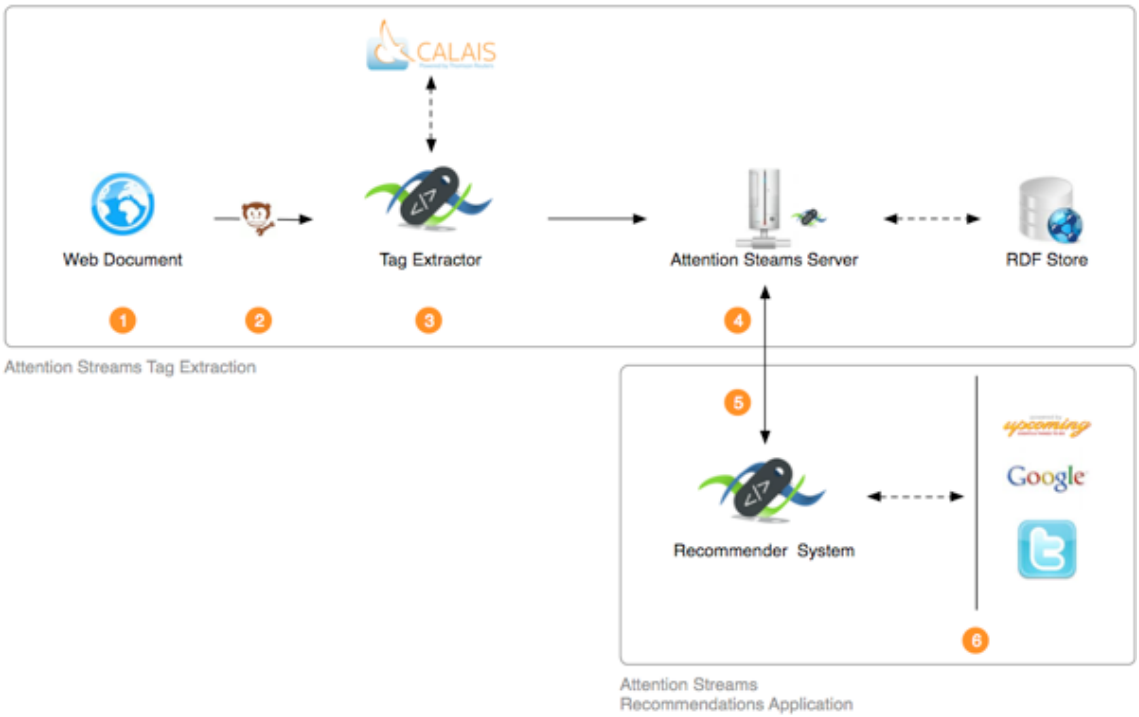

Figure 14: Attention-Streams Architecture

The first step is currently implemented as a Greasemonkey[16] script. A Greasemonkey script is a piece of JavaScript code that is executed automatically on the top of a set of specified pages. Such type of script can be compared to a bookmarklet except that it is executed automatically. The Greasemonkey script extracts the main concepts (using OpenCalais) from the pages the user visits and merges back the information to the page itself using RDFa. Then, this information can be used for applying filtering techniques within the page given the user attention tags.

In the second step, the extracted tags are merged to the user's FOAF profile hosted on the application server using the APML RDF Ontology. As the user updates his profile, the attention tags values increase or decrease for matching the increases and decreases in his interests. Currently, there are two tag lists differentiating the long-term tags and short-term tags. The retrieved tags enable the adaptation of a user interface by matching the user's tag and the tags within the interface. For instance, it becomes possible to hide irrelevant tagged information or highlight/extend the

[16] Greasemonkey, http://userscripts.org/

information that seems to be interesting for the user. In a similar fashion, it is feasible to give real-time information recommendations based on the user's activity tags. For example, it is possible to retrieve bookmarks, or news that might interest the user right now.

The last step of the system involves a small client-side recommender system application that provides real-time recommendations. These recommendations are performed using the attention tags of the user generated during the previous step. The application analyses in real-time the attention tags provided by the Greasemonkey script and the current user location for providing relevant real-time information suggestions from streaming sources such as twitter. The recommendation system enables the user to "star" items within its current context. While staring an item, the user stores a bookmark within the HTML5 Storage facility (a future version will enable remote bookmark storage). The bookmark appears in the headline section of the user interface when the user experiences the same attention later on.

The Attention Streams Architecture and applications are written using the Sparks Framework for the client side (for more information, see D1.2) and Ruby on Rails for the server side. The semantic models are hosted on a dedicated server using RDF. These models are currently using the APML and FOAF ontologies.

As mentioned, FOAF models the user profile while APML describes his current attention. After the creation of the profile, a user can install a Greasemonkey script that operates on any public web document (1,2). The script fetch the user attention model using tags retrieved from Open Calais (3). Then, the script sends the extracted tags to the Attention Streams Server for processing. When the server receives new Attention Tags, it performs some internal tasks in order to increase or decrease the user Attention Tags using heuristics (section 3.1.2). The recommendation application fetches the user Attention Model periodically from the server (5) and his current location using HTML5 geolocation features. After getting the user model, the system performs different tasks (6):

1) The most important tags and previous bookmarks (bookmarks are currently stored within the HTML5 database) are extracted from the current Attention Model and displayed accordingly;
2) RSS suggestion are fetched from Google and displayed;
3) If the user location is available, the location is accessed and local events displayed using the Upcoming API[17];
4) Tweets and events are extracted periodically for displaying useful tweets updates according to his Attention Model.

[17] Upcoming API, http://upcoming.yahoo.com/services/api/

3.1.4. Application Usage

As mentioned, the application is divided in two sections. The first application is in charge of collecting and creating the user attention model while the second one provides recommendations. It is important to note that the first application is independent from the final system as it only creates the user model. However, the recommender depends on the application. For instance, the recommender could provide recommendations that are more specific to emergencies (**Error! Reference source not found.**).

As illustrated in the previous picture, the recommender system can be used on mobile devices and on a standard computer. If the user is using the application on a desktop computer, he can transform the application to a desktop one using Mozilla Prism[18]. The interface of the application is created using jQTouch[19] so it can be transformed to a native iPhone application (the application works on other mobile platforms, such as Android).

Before starting using the recommendation system the user must add some attention tags to his APML profile. In order to do it, the Greasemonkey script must be installed and Greasmonkey activated. By simply going on a public web page, the Attention Stream script will extract the attention tags from the current page (the tags are extracted 20 seconds after the page loading process). It is important to note that some pages (such as Google) do not allow tag extraction. The user may also access the tags of the external link of a page by clicking on the small character situated after each external link. If the user clicked on it, the character turns green and the user can access the tags of the linked page and decides to add them directly to his profile by clicking on the plus button.

[18] Mozilla Prism, http://prism.mozillalabs.com/

[19] jQTouch, http://www.jqtouch.com/

Figure 15: The Recommendations Application User Interface

The user can start using recommendations by starting his personal recommendation application (Figure 15: The Recommendations Application User Interface). The application returns real-time tweets (4), local events (6) and RSS recommendations (3) based on his current attention (1) and location (5). Each tweet suggestion may be bookmarked using the star button (4). The bookmarked items will be displayed later on when a similar attention occurs (2).

For more information and a demo, the webpage of the system can be accessed (**http://nebula.dcs.shef.ac.uk/sparks/astreams**).

3.1.5. Conclusions and Future Work

The Attention-Streams model enables WeKnowIt to gather attention information from its users within and outside the system. Such information is particularly relevant in the context of the ER Scenario where the citizen use the system occasionally. In this context, it is possible to understand what users are interested in real-time and provide recommendations accordingly.

The current implementation is able to analyse almost any web page and give generic recommendations within an external application. The

recommendations rely on different external web APIs and use OpenCalais for generic tag recognition.

The ongoing and future work will explore the creation of a tag recognition service that is able to recognise domain specific tags (ER and CSG specific tags) and the integration of WeKnowIt APIs. For instance, the integration of the contextual model is envisaged. Such integration could adapt the recommendations depending on the physical situation of the user (for instance, if the bandwidth is low, the images present in the recommendation could be turned off automatically). At the moment, an updated version of the algorithm is being implemented. Such algorithm will provide better analysis of the attention.

3. Services and Integration

The different technologies discussed in the previous sections needs to be integrated in the WeKnowIt platform. Such integration is necessary in order to make accessible these different technologies to the whole platform.

The CURIO User Model, the Context Model and Attention-Streams have been integrated to the WeKnowIt system using different patterns. For instance, CURIO is integrated within the Common Model (D6.4.1 and D5.3.1) while the Contextual User Model services are part of the WeKnowIt core services and the Attention-Streams services are implemented as external services that can be called by any component of the system.

In addition to these models, the other technologies supporting user modelling and personal information storage developed for T1.1 and T1.2 have been integrated within the system using the built-in service model of the WeKnowIt platform. These services have been described in more detailed in D6.2.1 and their description is annexed to this document (Annex B.).

As stated in D6.3, the CURIO User model is integrated with the Common Model, such integration makes it possible to create and access user information using the WeKnowIt Data Storage and the services that have been implemented by WP1 (Annex B.).

The context modelling services have been integrated into the WKI architecture. Two principal methods are used to access these services (Annex B.). First, the "setSensorValue" method enables a client to send a sensor value for a given user and sensor. This value is then considered to be the current value of the sensor, thus clients need to ensure that the values provided are reasonably up to date if they are time critical. The client or internal service can then request a given user context using the "getContext" procedure. This request needs to be called with the necessary sensor model and user id for being effective. Thus both clients and internal services are able to integrate user context into their interfaces and processing.

The Attention-Stream model is integrated using REST APIs. The services can be called using external APIs from any WeKnowIt services. These APIs are considered "external" because they can be accessed outside the WeKnowIt system.

The Attention-Streams services are external because they need to work even if the user is not currently using the WeKnowIt platform. Attention-Streams provides different services that can be categorised in three different categories: Attention Update, Attention Retrieval and Recommendations. These services use REST APIs and are accessible

through its own server hosted at the University of Sheffield (**http://nebula.dcs.shef.ac.uk/sparks/astreams**).

The Attention-Streams services allow the creation of a new user profile, the addition of attention tags, the retrieval of an attention profile. The services also support the retrieval of recommendations using the JSON format. The different services are described in more details in the appendix (Annex C.).

4. Conclusions

This document discusses the user modelling techniques that have been applied and developed in WeKnowIt. In order to be successful, the WeKnowIt infrastructure needs a user model integrated with the other information stored in the system. In such context a system wide ontology called CURIO has been developed for tying together all the collective knowledge stored in the system. This ontology is centred on the concepts of users, resources and events.

As presented, it is also necessary to provide means for personalising the user interaction with the system. In such setting, we developed two different methodologies that aim to help in two different situations. On one hand, user context modelling give a method for adapting user interface and interaction patterns such as interaction simplification. On the other hand, attention profiling provide a general framework for tracking user attention in real-time wherever it happens thus providing methods for retrieving relevant user content or filtering information that is not needed.

Future work suggests a tighter integration of the cited technologies within the two demonstrators. Particularly, in the case of the CSG scenario, the developed recommender system could suggest information related to the local environment by feeding its information directly from other user-generated content. In the case of the user context, the benefits would be largely in the ER case where, for instance, bandwidth management is critical.

5. References

Adomavicius, G. & Tuzhilin, A., 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 734–749.

Alexander, B., 2006. Web 2.0: A new wave of innovation for teaching and learning. *Learning*, 41(2), 32–44.

Bennett, F., Richardson, T. & Harter, A., 1994. Teleporting - Making Applications Mobile. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*. Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on. pp. 82-84.

Bojars, U. et al., 2008. Using the Semantic Web for linking and reusing data across Web 2.0 communities. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1), 21-28.

Brickley, D. & Miller, L., 2005. *FOAF vocabulary specification*,

Burke, R., 2002. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4), 331–370.

Chen, G. & Kotz, D., 2000. *A survey of context-aware mobile computing research*, Citeseer.

Cilibrasi, R. & Vitanyi, P., 2004. Automatic meaning discovery using Google. *Manuscript, CWI*.

Daoud, M. et al., 2007. Learning implicit user interests using ontology and search history for personalization. In *Web Information Systems Engineering–WISE 2007 Workshops*. pp. 325–336.

Ireson, N. & Burel, G., 2010. Knowledge Sharing in E-Collaboration. *Electronic Government*, 351–362.

Kobsa, A., 2007. Generic user modeling systems. In *The adaptive web*. pp. 136–154.

Lavie, N. et al., 2004. Load theory of selective attention and cognitive control. *Journal of Experimental Psychology-General*, 133(3), 339–353.

Marmasse, N. & Schmandt, C., 2000. Location-aware information delivery with commotion. In *Handheld and Ubiquitous Computing*. pp. 361–370.

Neumann, O., 1996. Theories of attention. *Handbook of perception and action*, 3, 389–446.

Oreilly, T., 2007. What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. *SSRN eLibrary*. Available at: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1008839&download=yes%22 [Accessed January 10, 2011].

Sauermann, L. et al., 2006. Semantic desktop 2.0: The gnowsis experience. *The Semantic Web-ISWC 2006*, 887–900.

Sauermann, L., Van Elst, L. & Dengel, A., 2007. Pimo-a framework for representing personal information models. *Proceedings of I-Semantics*, 7, 270–277.

Scherp, A. et al., 2009. F–a model of events based on the foundational ontology dolce+ DnS ultralight. In *Proceedings of the fifth international conference on Knowledge capture*. pp. 137–144.

Schilit, B., Adams, N. & Want, R., 1994. Context-Aware Computing Applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*. Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on. pp. 85-90.

Schmidt, A., Beigl, M. & Gellersen, H., 1999. There is more to context than location. *Computers & Graphics*, 23(6), 893-901.

Treisman, A.M. & Gelade, G., 1980. A feature-integration theory of attention. *Cognitive psychology*, 12(1), 97–136.

Want, R. et al., 1992. The active badge location system. *ACM Transactions on Information Systems*, 10(1), 91-102.

A. The CURIO Ontology

This section present the different parts of the CURIO ontology as a diagram (Figure 16: The CURIO Ontology). This section completes the CURIO presentation of the user model description and the online documentation of CURIO (**http://purl.org/net/curio/ns#**).

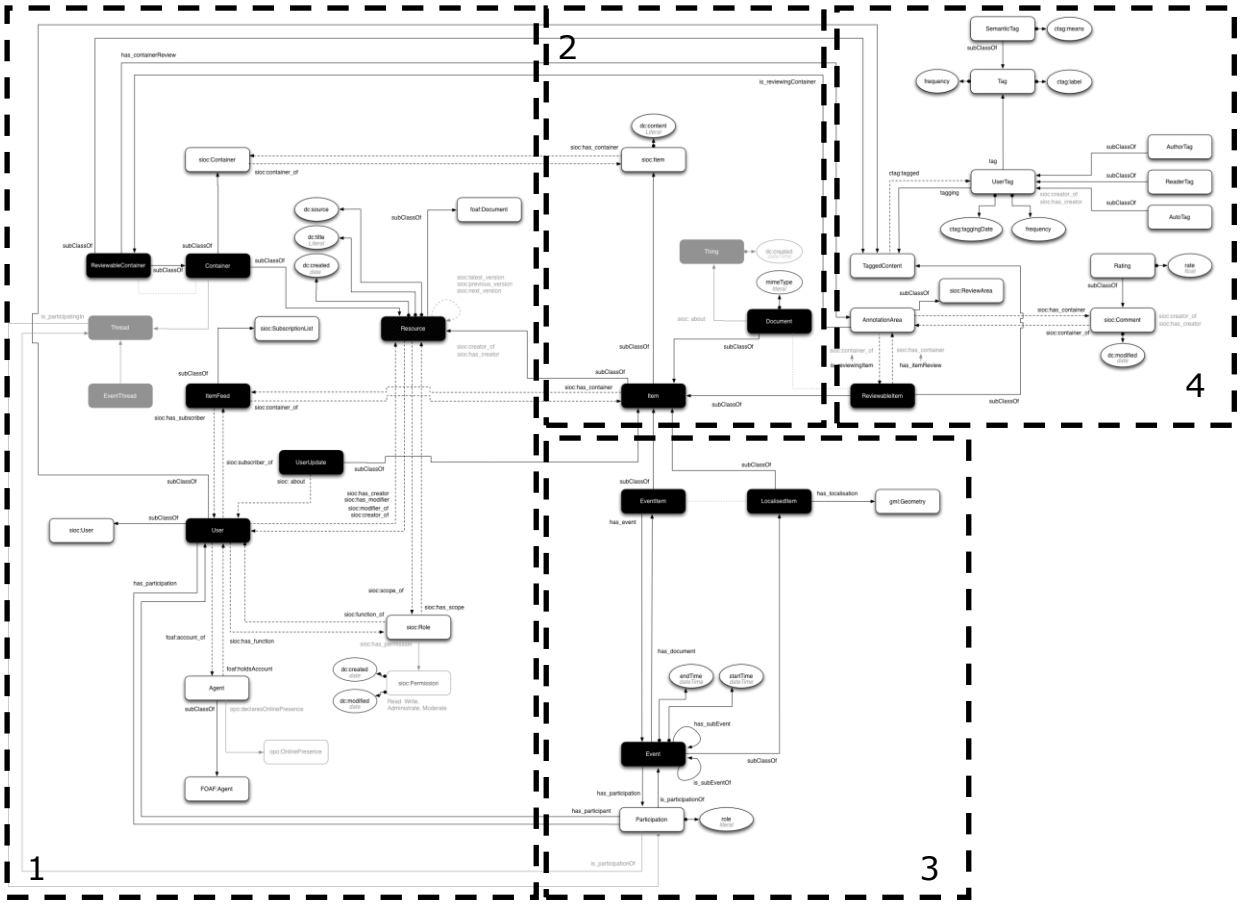

Figure 16: The CURIO Ontology

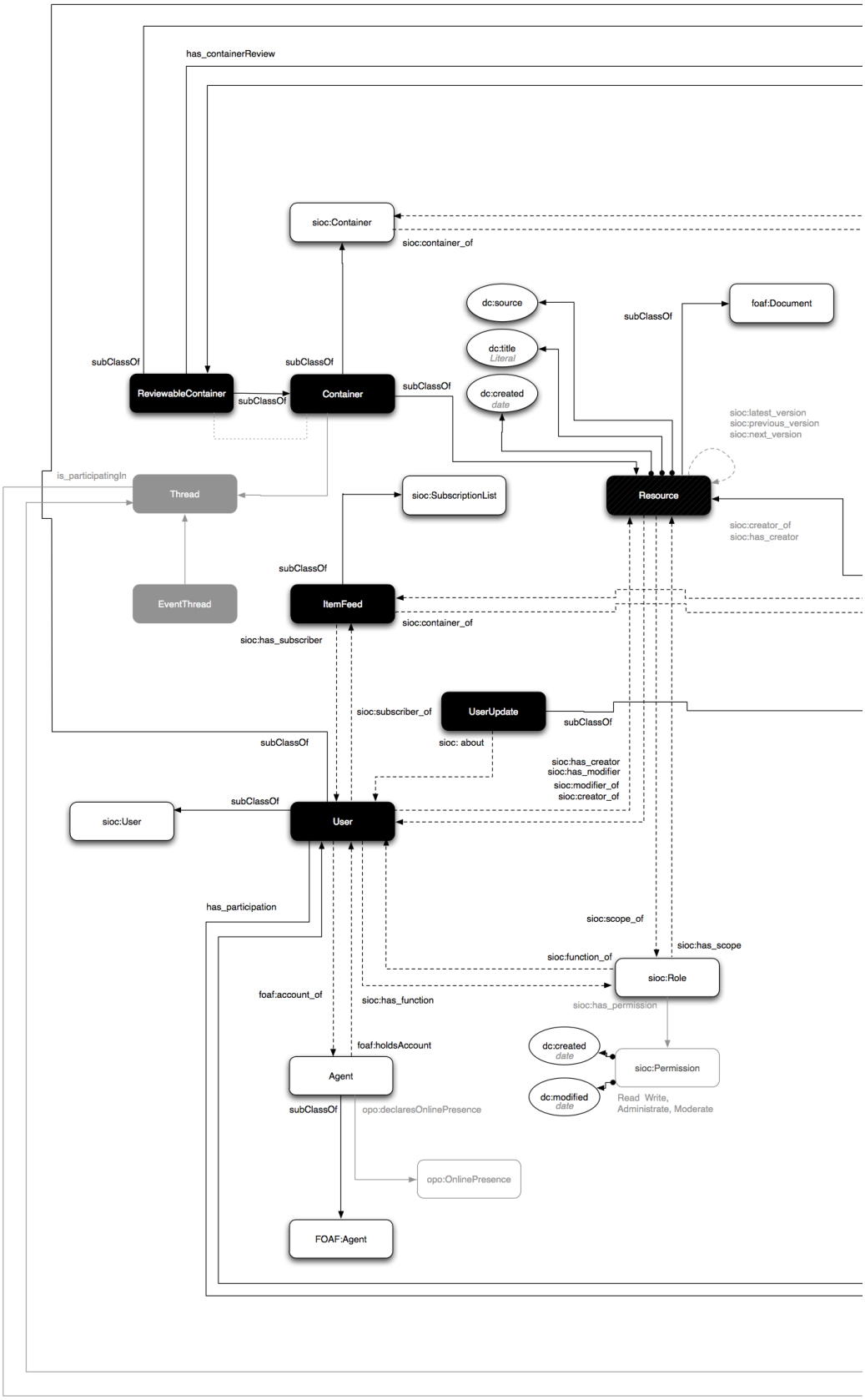

Figure 17: The CURIO Ontology: Part 1

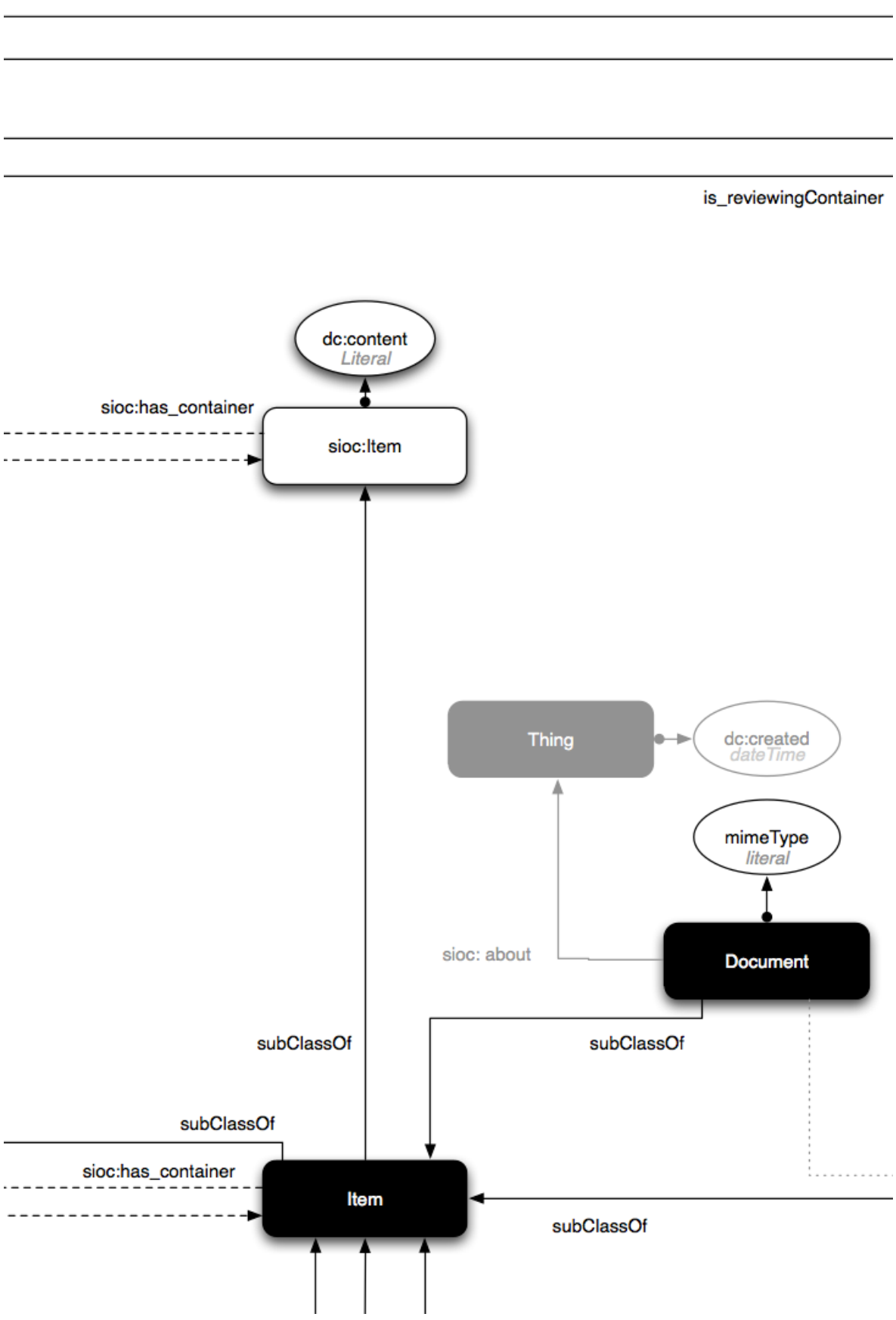

Figure 18: The CURIO Ontology: Part 2

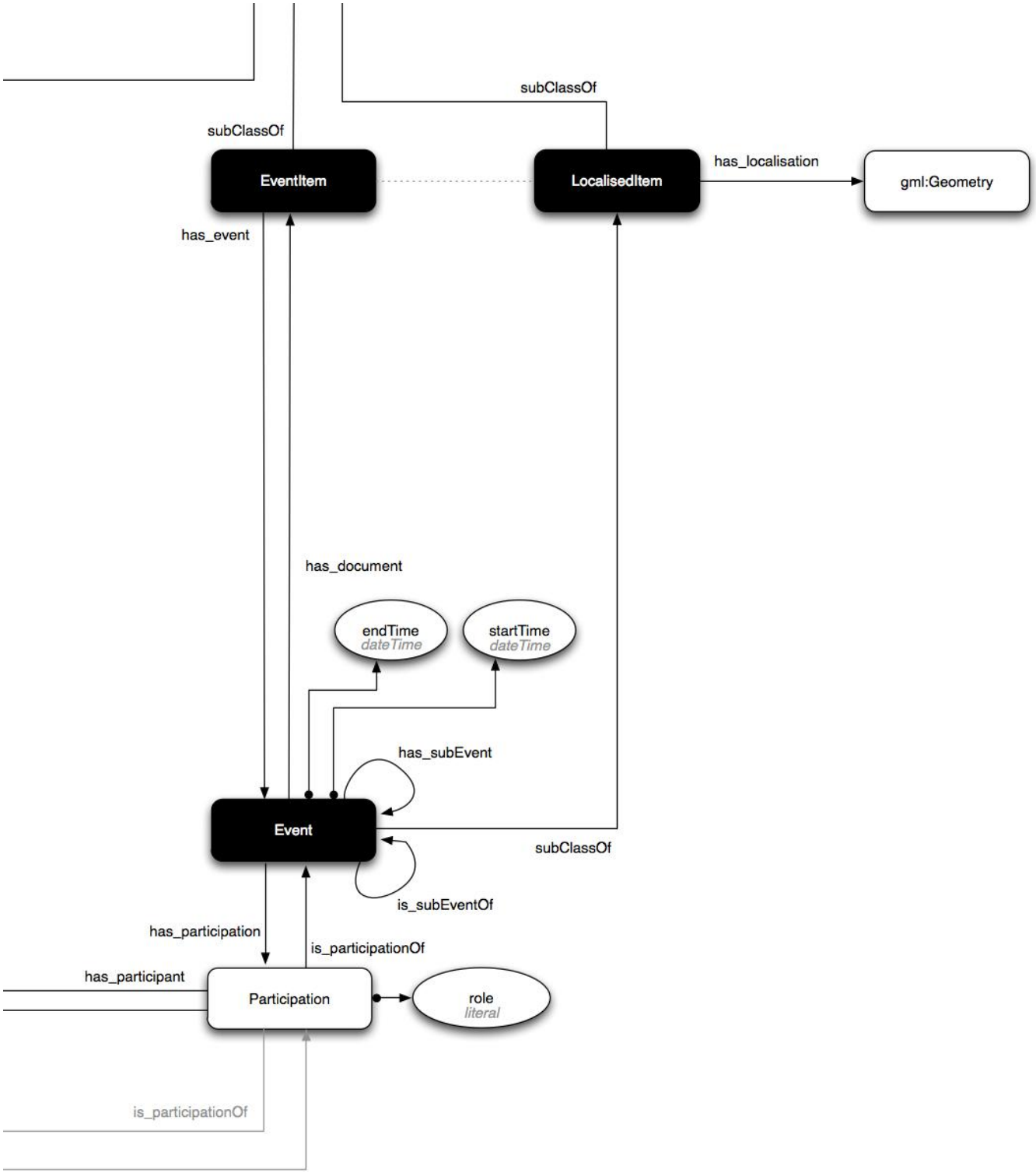

Figure 19: The CURIO Ontology: Part 3

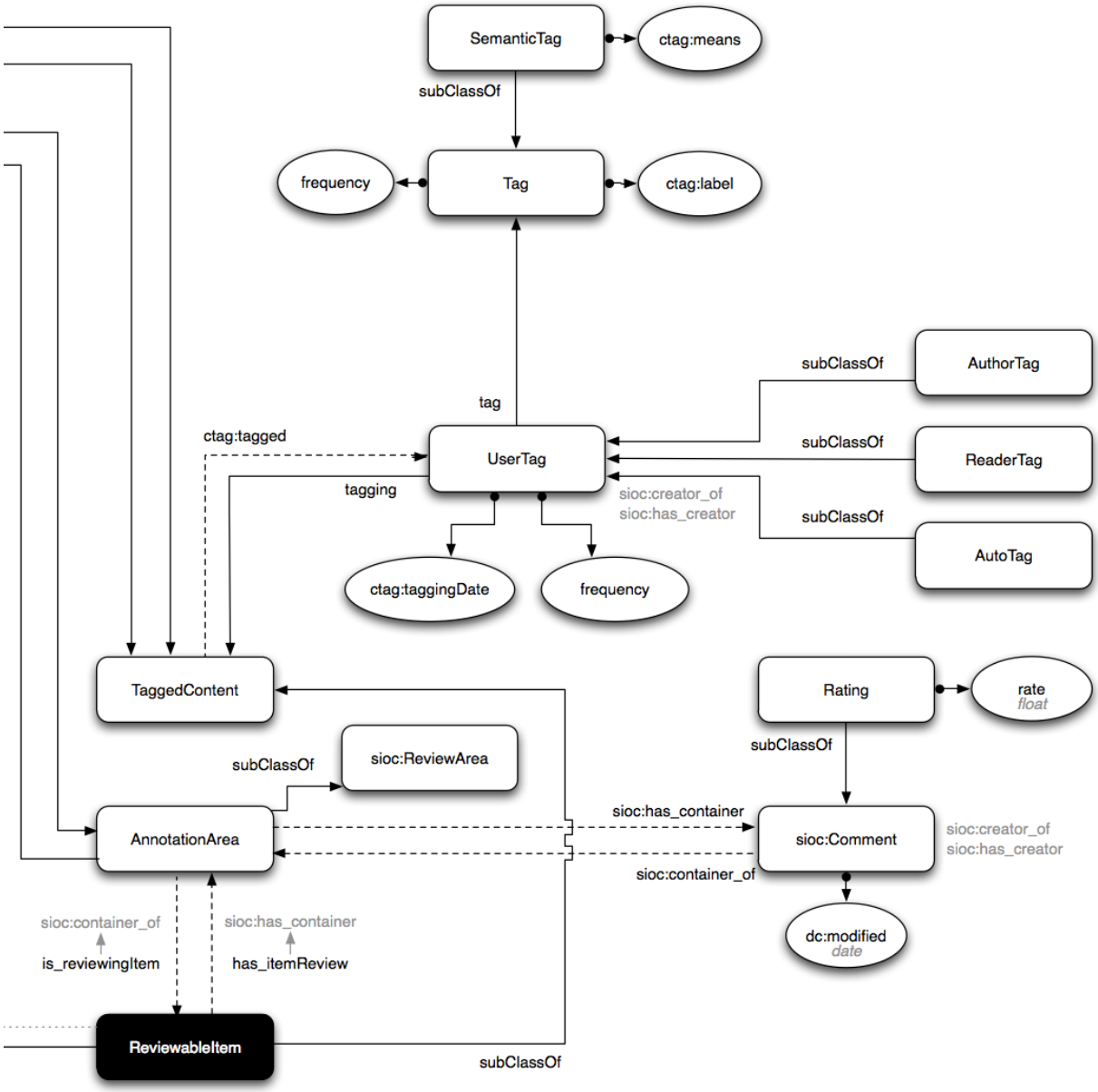

Figure 20: The CURIO Ontology: Part 4

B. Integrated Personal Intelligence Management Services

This section describes the details of the specific services implemented for the management of Personal Intelligence.

B.1. ManageItems

Method	Inputs	Effect
uploadItem	The URI of the item The relevant permissions	This method stores the relevant item alongside the given permissions level in the Knowledge Base
deleteItem	The ID number of the item	Removes the item from the Knowledge Base

B.2. Tag

Method	Inputs	Effect
addTag	The ID of the item to add Tags to A list of tags to be added to the item	The tags are stored in the Knowledge Base and are linked to the given document
getItemTags	The ID of the item to retrieve tags for.	Returns the tags appropriate to this item from the Knowledge Base
modifyTag	The ID of the item The ID of the tag for the given item The modified Tag	Modifies the given tag associated with the relevant item in the knowledge base
deleteTag	The ID of the item The ID of the tag for the given item	Removes the specified tag from the specified item.

B.3. Comment

Method	Inputs	Effect
addComment	The ID of the item to comment The text of the given comment	Attaches the provided comment to the given item within the Knowledge Base
modifyComment	The ID of the item The ID of the comment The new text of the comment	Modifies the specified comment with the new text in the Knowledge Base
deleteComment	The ID of the item The ID of the comment	Removes the item from the Knowledge Base

B.4. Rate

Method	Inputs	Effect
addRating	The ID of the item to rate The value of the rating	Stores the provided rating alongside the item in the Knowledge Base
modifyRating	The ID of the item to modify The ID of the rating to modify The new value of the rating	Updates the given rating within the Knowledge Base

B.5. Account Manager

Method	Inputs	Effect
createUser	The OpenID reference for this user	Creates this user account within the Knowledge Base
createUser	The username of this user The password for this user	Creates this user account within the Knowledge Base and attaches the given user name and password to the user
updateUser	The User to update	Updates the Knowledge Base to reflect any changes to the given User
deleteUser	The User to be deleted	Removes the provided user from the Knowledge Base

B.6. Log In

Method	Inputs	Effect
userLogin	The OpenID of the user	Logs the user in to the system
userLogout	The OpenID of the user	Logs the user out of the system
userLogin	The ID of the user The password for this user	If the credentials match the User is logged in to the system
userLogout	The ID of the user	The User is logged out of the system

B.7. Search Knowledge Base

Method	Inputs	Effects
searchQuery	The ID of the user The keywords required for searching	Builds a list of results from the information contained in the Knowledge Base

B.8. ContextModelling

Method	Inputs	Effects
getContext	The ID of the user The model with which to generate the context	Uses the specified context model to determine the current context of the user
setSensorValue	The ID of the user The name of the sensor The value of the sensor	Stores the current value of the sensor for the given user. Can be called prior to requesting the user context or continuously.

C. External Personal Intelligence Management Services

This section describes the details of the external services implemented for the management of Personal Intelligence. These services can be accessed using REST APIs.

C.1. Attention Update

Method	Inputs	Effect
create (**http://nebula.dcs.shef.ac.uk/sparks/services/foaf/user/create**)	The user name. The content of a FOAF profile.	Create a new Attention-Streams profile and return a profile identifier.
update **(http://nebula.dcs.shef.ac.uk/sparks/services/foaf/user/<UserID>/apml/update)**	The user ID. The list of concepts to integrate to the APML profile. The URI of the document where the concepts where found.	Update the Attention-Space of the user.

C.2. Attention Retrieval

Method	Inputs	Effect
now (**http://nebula.dcs.shef.ac.uk/sparks/services/foaf/user/<UserID>/apml/now**)	The user ID. The callback function The size of the Attention Space.	Retrieve the current APML profile of a user in JSON+RDF.

C.3. Recommendations

Method	Inputs	Effect
events (**http://nebula.dcs.shef.ac.uk/sparks/services/foaf/user/<UserID>/aml/now/events**)	The user ID. The longitude and latitude of the user.	Retrieve a set of local events given the user location and attention.
titles (**http://nebula.dcs.shef.ac.uk/sparks/services/foaf/user/<UserID>/aml/now/rss/titles**)	The user ID.	Retrieve a set of interesting news from RSS given the current user attention.