A Context Lifecycle for Web-Based Context Management Services

Gearoid Hynes, Vinny Reynolds, and Manfred Hauswirth

Digital Enterprise Research Institute,
National University of Ireland, Galway
firstname.lastname@deri.org

Abstract. During the development of context aware applications a context management component must traditionally be created. This task requires specialist context lifecycle management expertise and hence can be a significant deterrent to application development. It also removes the developers focus from differentiation of their application to an oft repeated development task. This issue can be addressed by encapsulating the context management lifecycle within a web-service, thus providing applications with a low-overhead alternative to managing their context data. The adoption of a web-based approach maximizes the potential number of interacting applications, including smart spaces, web and mobile applications, due to ease of access and widespread support of web technologies. The contribution of this paper is the development of a lifecycle, based on existing work on enterprise data and context aware lifecycles, which is optimized for web-based context management services (WCXMS) and the provision of a web-service implementation of the lifecycle.

Keywords: context data, lifecycle, webtechnologies, context management.

1 Introduction

Development of context-aware applications is no longer limited to the domain of ubiquitous computing, mobile and web-based context-aware applications are also being developed. Recently several location broker applications, including FireEagle by Yahoo![3] and Google Latitude [1], have been launched and their growing popularity illustrates the demand for real world location data within the virtual world. However, location alone is only a small portion of a user's context, the remainder of which has thus far been ignored. In order to provide truly context-aware applications we must move beyond serving location data alone by providing a service capable of managing multiple types of context in a time-sensitive manner. By providing *context as a service* (CXaaS) we can provide application developers with a foundation upon which they can build their context-aware applications. This enables developers to focus on their application functionality rather than the development of context management components.

The use of a web-based approach facilitates interaction with many different types of applications, including traditional context consumers such as smart-spaces

P. Barnaghi et al. (Eds.): EuroSSC 2009, LNCS 5741, pp. 51–65, 2009.

and the more recent consumers of context data such as web-applications and mobile applications. Additionally, the use of a centralized approach facilitates the sharing of context data between applications. This means applications can access context data from sources which they would not ordinarily have access to.

The realization of CXaaS requires as much of the context lifecycle as possible to be encapsulated within the web-service. In order for this to occur a formal model for the lifecycle of context data within a WCXMS must be developed. Current web-based location management services do not provide a reference lifecycle which can be extended to include other types of context data, therefore the lifecycle presented in this paper is based upon related lifecycle work from other domains, in particular enterprise data lifecycles and ubiquitous system context lifecycles. In this paper we will present a lifecycle for web-based context management services, we begin with the requirements on the lifecycle, followed by development of the lifecycle itself and how it was influenced by existing lifecycles. The paper continues by introducing ConServ, which is an implementation of the lifecycle, and then describing two applications which have been developed using ConServ. The paper finishes with some related work and conclusions.

2 Lifecycle Requirements

In order to provide CXaaS there are several requirements which the lifecycle must fulfill in addition to the *standard lifecycle requirements such as acquiring, storing and delivering the context data.* As outlined in the introduction, CxaaS has two primary goals, firstly *to enable the rapid development of context aware applications* and secondly *to facilitate the sharing of context data in a controlled manner between context applications.* Stemming from these two key requirements are some additional criteria without which the service would not succeed, of which the main ones are described below.

The development of a generic context lifecycle which would be applicable to all types of context data is not feasible, therefore, the lifecycle should be optimized for personal context data as it is the most common category of context data used by existing context aware applications. When dealing with personal context data privacy is crucial. Users must have complete control over who has access to what parts of their context data and at what level of granularity, at what time, from what location etc. Therefore, *expressive privacy policies are necessary to ensure the safety of users' context data.*

Context data can come from a variety of sources such as GPS sensors, accelerometers, calendars and online profile data. For some of this data it is appropriate for it to be pushed to the service and for other data types it is more appropriate for it to be pulled. As a result *it is necessary that the lifecycle account for context data being both pushed and pulled into the system.*

The lifecycle must allow for interaction with third parties, both in the acquisition and provision of context data. Additionally, *An expressive, extensible and commonly understood context data model is necessary* in order to ease importing/exporting of data from/to third parties. Extensibility allows developers to model any concepts which have not been previously modeled within the system.

3 Lifecycles

A substantial amount of related lifecycle work exists which can be leveraged in the development of the WCXMS lifecycle. This section analyses the related work, which is divided into two classifications of lifecycles:

- Enterprise Lifecycle Approaches: Lifecycles which are developed for enterprise applications but which do not explicitly target context data. These lifecycles are robust and well-established industry standard strategies for data management.
- Context Lifecycle Approaches: Lifecycles for managing context data, however they have not undergone the same level of testing as the enterprise approaches.

3.1 Enterprise Lifecycle Approaches

There are several well established data lifecycles approaches in use within enterprise which can influence the WCXMS lifecycle. Enterprise lifecycles are formulated to control the lifecycle of sensitive data in enterprise applications. In comparison to context data, the data stored within the enterprise applications is relatively static. For instance an enterprise application which stores users' contact details is not going to receive as many updates as a context aware application which stores users' location data. However, thanks to commercial support from some of the large software vendors, such as Oracle and Microsoft, enterprise lifecycle approaches are robust and well tested and for those reasons should be analysed to identify any potential relevance to the WCXMS lifecycle. Two of the more popular enterprise lifecycle approaches are Information Lifecycle Management (ILM) [17] and Enterprise Content Management (ECM) [4]. Both ILM and ECM consist of more than just the lifecycle, but also the tools, practices and methods which surround the lifecycle.

ILM is defined by the SNIA Information Lifecycle Management Initiative to consist of "*the policies, processes, practices, services and tools used to align the business value of information with the most appropriate and cost-effective infrastructure from the time information is created through its final disposition*"[17]. The five steps in the ILM are show on the left of Figure 1. ILM is primarily focused on data lifecycles which occur within the same organization; while data may initially be received from a third party, each of the ensuing steps occur within the same organization. As a result the lifecycle is not suitable for the WCXMS application where interaction with third parties is of great importance. However the steps following the data use are relevant as context data must also be maintained and disposed of.

ECM is said to be "*the strategies, methods and tools used to capture, manage, store, preserve, and deliver content and documents related to organizational processes*"[4]. The ECM lifecycle, seen in Figure 1, is not applicable to context data as it is overly focused on the steps prior to data dissemination, whereas ILM is more concerned with the steps after data distribution. The steps within

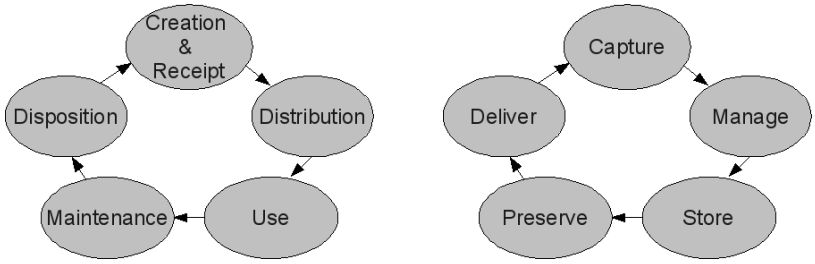

Fig. 1. ILM and ECM Lifecycles

the ECM strategy are prepossessed towards data storage with two entire steps, *Store* and *Preserve*, dedicated to it and a portion of the *Manage* step is also storage related.

Hayden[11] provides an extensive data lifecycle management strategy containing ten steps. Haydens steps, shown in Figure 2, are focused on a distributed enterprise setup which is subject to Federal data retention rules therefore not all of the steps, such as *Transmission & Transportation* and *Retention*, are applicable to a WCXMS. Hayden's lifecycle includes steps to mitigate the impact of users altering data in an unauthorized manner during the lifecycle (*Manipulation, Conversion/Alteration*) this is not an issue for a WCXMS as only the data owner can alter their data. Hayden strikes a balance between ICM and ECM by having a better distribution of steps before and after the *Release* of the data. A *Classification* stage was not present in the previous strategies and is of interest to the WCXMS lifecycle as classification of acquired context data is important due to the variety of context types and the importance of knowing the relationship between different pieces of context data.

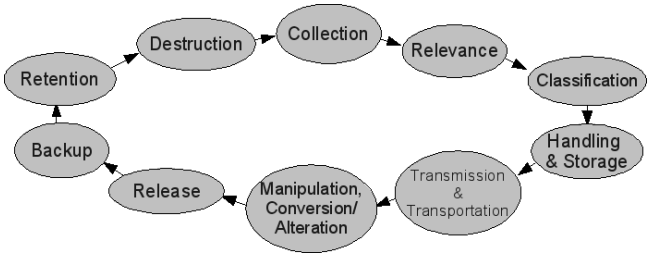

Fig. 2. Hayden's Data Lifecycle

3.2 Context Lifecycle Approaches

Different approaches have been taken to the area of context lifecycle management depending on the application requirements. Chantzara and Anagnostou [7] have identified four separate stages in the lifecycle, shown on the left of Figure 3, and the stages are distributed across three types of actors; *Context Providers*, a

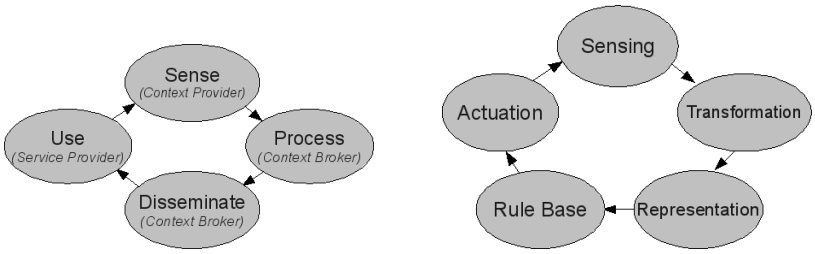

Fig. 3. Chantzara and Ferscha's Lifecycles

Context Broker and context aware *Service Providers*. This approach is of great relevance to the WCXMS lifecycle as the interacting actors in both systems are quite similar. However, the model is overly simplistic, only two stages are handled by the context broker, and would require further elaboration in order for its adoption within the WCXMS lifecycle.

Ferscha et. al. [9] provide details of their lifecycle for context information in wireless networks. They begin with the *Context Sensing* stage, which can be a time triggered or event triggered acquisition of low level context data, followed by the *Context Transformation* stage which involves the aggregation and interpretation of the sensed information, the next stage is *Context Representation*, whereby the transformed context is applied to an ontology to create high level contextual information, *Context Rules* are then applied to the semantic context data which then sends commands to the *Actuation* stage. This lifecycle has more stages than the previous one and it makes use of both ontologies and rules which makes it relevant to the WCXMS lifecycle. The functionality of the *Transformation*, *Representation* and *Rule Base* stages are of particular interest. Unfortunately there is no distinction between different actors within the lifecycle and therefore it is only applicable for adoption within a controlled environment. Additionally, after the *Actuation* stage the lifecycle restarts and as there are no stages for data storage or controlled disposal of the data, as a result the strategy is incomplete.

As part of the MOSQUITO project Wrona et. al. [21] developed a simple three stage context life cycle which was made up of "*Context Information Discovery*", "*Context Information Acquisition*", "*Context Information Reasoning*". This scheme is incomplete as there is no mention of a stage which uses the context data or a stage which disposes of the data. But the *Context Information Acquisition* stage is applicable to the WCXMS lifecycle as it is more generic than the *Context Sensing* stages used by the other lifecycles and hence applies to pushing/pulling of context data from many different sources, not only sensors.

4 WCXMS Lifecycle

The WCXMS context lifecycle, shown in Figure 4, borrows aspects from many of the above strategies. The lifecyle has been separated between three actors

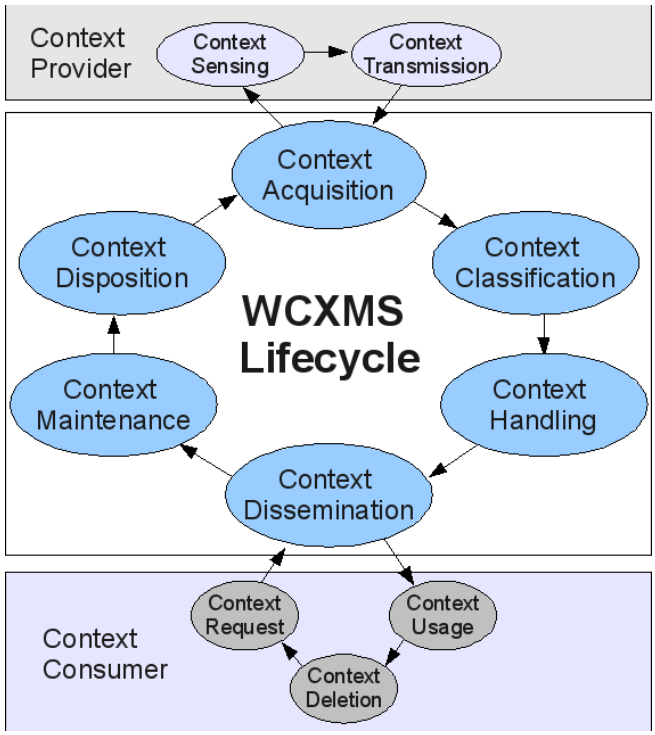

Fig. 4. The WCXMS Lifecycle

in a similar manner to Chantzara and Anagnostou. The *Context Provider* and *Context Consumer* are third party applications, for example, web-applications, smart-spaces, mobile applications which either provide and/or consume context data. The *Context Acquisition* stage, which is inspired by MOSQUITO, provides mechanisms for the pulling and pushing of context data. Two stages overlap with stages in Hayden's approach, *Classification*, which classifies the acquired data using a set of context ontologies, and *Handling*, which expands on the classified data to create associated data with different levels of granularity. Hayden's *Relevance* stage is also integrated into the *Classification* stage as the relevance of WCXMS context data is decided once it once it has been classified.

Context is disseminated in accordance with user defined privacy policies. Users have complete control over their data, i.e. who has access to that data, at what level of granularity and under what conditions. Therefore, once a Context Consumer makes a Context Request for a user's data the user's privacy policy is examined in order to establish what permissions they have been given for that data.The *Maintenance* and *Disposition* stages are borrowed from ILM. The result is a formal context management lifecycle optimized for deployment in a

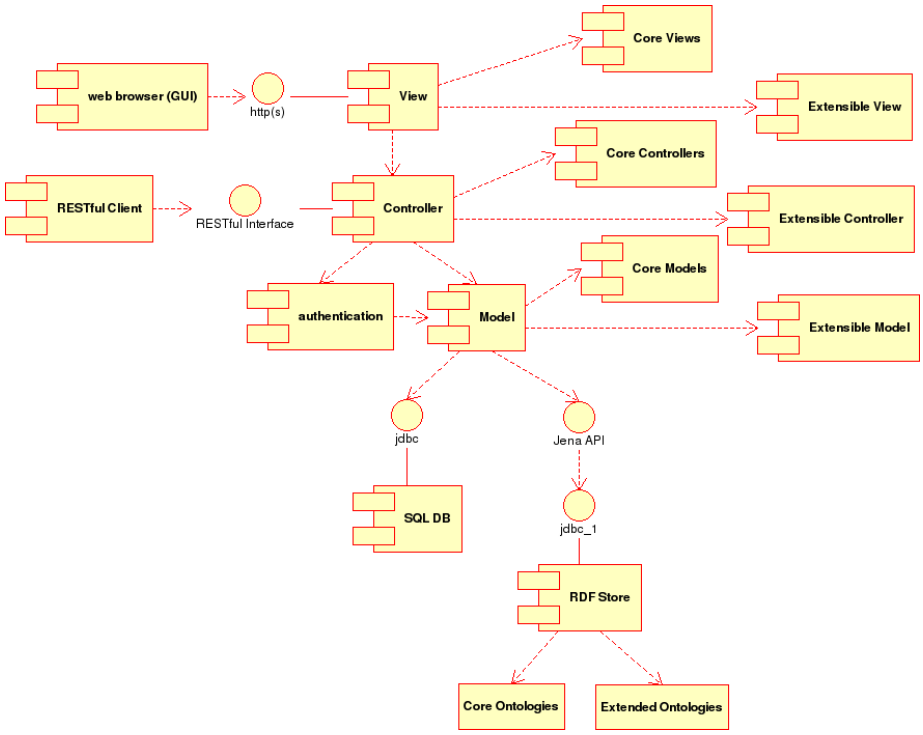

Fig. 5. ConServ Component Diagram

web-based environment. The remainder of this section provides details of a WCXMS lifecycle implementation called ConServ and describes the implementation details of each of the lifecycle steps.

ConServ Overview

ConServ is a REST [10] based web service developed using the model-view-controller (MVC) architectural pattern. ConServ has a core set of ontologies which are tightly coupled to the web-service implementation in order to provide a fast and reliable service for managing popular context data types. The ability to extend the types of context data managed is also provided. By allowing developers to upload additional ontologies to ConServ they can extend the core set of ontologies with any additional concepts which have not been modeled. The component diagram in Figure 5 shows the differentiation between the core and the extensible components, along with the MVC structure of the web-service.

4.1 Context Acquisition

As per the WCXMS lifecycle requirements ConServ can receive context data by pulling or by data sources pushing context data. The *Controller* component

handles the *Context Acquisition* stage of the lifecycle. Pushed data can either be manually entered via the web-interface or pushed to the RESTful interface. Context data can also be acquired by pulling data from a context source. Acquiring calendar data is a good example of this; the user provides the URL of their iCalendar file, this XML file is then imported by ConServ and the data in the iCal file is then provided to the classification stage of the lifecycle.

4.2 Context Classification

Context classification occurs in two different ways depending on whether the context data was pulled or pushed. If the context data was pushed then the URL which it was pushed to is decoded into a class and a property for a particular ontology and then the data is converted into RDF before being passed to the *Context Handling* stage of the lifecycle. If the context data is pulled, it is converted to RDF if necessary, then the individual classes are separated and forwarded to the *Context Handling* stage.

Context Ontology Description. A set of core ontologies and related ontological extensions are used to classify the context data acquired by ConServ. The core ontologies model users, devices, policies, locations, events and services. Based on the definition that ontologies are *shared models of a domain that encode a view which is common to a set of different parties* [5] we have opted to use existing popular ontologies where possible in order to ease integration with ConServ and to leverage existing context data rather than create a completely new set of ontologies. The structure of the context ontology is illustrated in Figure 6 and an overview of the key classes is provided below.

Location: The GeoNames location ontology [19] and web service is used as the core of the ConServ location system. The GeoNames ontology is extended to provide a method of modeling buildings, floors, rooms and more. This extension is based on the MIBO [16] building ontology which has been adapted to interoperate with the GeoNames ontology which uses the SKOS [14] ontology.

Person: The FOAF ontology [6] is one of the most commonly used ontologies on the web and therefore was an obvious choice for modeling users within ConServ. FOAF provides many useful of properties for modeling such things as name, date of birth, email address, acquaintances, publications and group memberships to name but a few.

Events: Events are modeled by extending the icaltzd format [20]. Event data can either be imported from an iCal file or manually created using the RESTful interface or the web-interface.

Detection: The detection of a person results in the creation of a detection event. A detection event is associated with a time, location and device; the device is

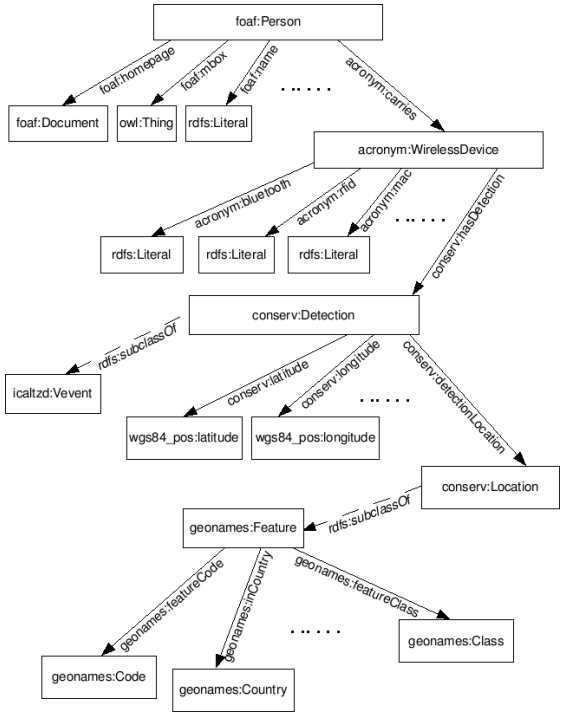

Fig. 6. Structure of the ConServ context ontology

in turn associated with a person. The time at which a detection occurs can be instantaneous or it can be over a period of time, for instance if a person is sitting at their desk for 2 hours, rather than create a new detection every time the person is detected a single detection event can be created with a 2 hour duration.

Devices: Due to the vast array of available personal mobile devices it is difficult to model each individual mobile device and their capabilities and keep the model updated. We have taken the approach of modeling mobile devices based on their connectivity options (Bluetooth, WiFi, RFID). The ACRONYM ontology [15] is used for modeling the devices and it has been extended with the *conserv#hasDetection* property in order to connect devices to detection events.

4.3 Context Handling

The primary function of the context handling step, which occurs in the *Model* component, is to facilitate ConServ's total forward-chained reasoning. ConServ materializes all appropriate location and event data in order to provide access to those concepts at different levels of granularity. Location has thirteen levels of granularity which directly map to levels used in both the GeoNames and MIBO ontologies.

For instance if ConServ is informed that a user, John Smith, is located in the DERI building it then materializes that John Smith is located in Lower Dangan, Galway City, Galway County and Ireland. This allows John Smith to specify at which level he wishes particular context consumers to access his location data. The context granularities are modeled using the SKOS Ontology [14] with the granularities being defined within a *skos#OrderedCollection* which contains a *skos#Concept* for each level of granularity for a particular context type.

There are 3 levels of granularity in event context data. These levels are directly related to the access levels given in online calendar applications. Level 0 does not allow any access to the calendar data, Level 1 allows services to see when the user is either free or busy but it does not provide any details about the events in questions, the final level provides all the available details about the events.

4.4 Context Dissemination

The dissemination of context data to the context consumer is controlled by user defined access policies. The ConServ Policy Ontology, shown in Figure 7, is inspired by the SOUPA Policy Ontology [8] which is in turn based on the Rei policy ontology[12]. Each user has one policy and this policy contains two types of objects, *conserv#AccessRule* and *conserv#AccessRequest*.

Instantiations of *conserv#AccessRequest* are created by services and are used to request access to a particular type of context data. These requests can then

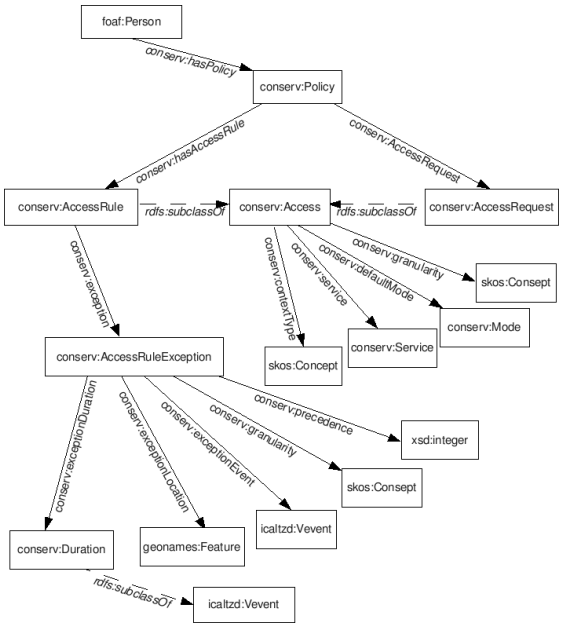

Fig. 7. An overview of the structure of the ConServ Policy

be approved or deleted by the user if they do not wish to allow any access. If the request is approved it is converted into a *conserv#AccessRule* where the user can apply conditions, modeled as *conserv#AccessRuleException*, to when the service can access the data. The *conserv#AccessRuleException* class has a granularity property which sets the granularity at which the context data in question will be accessible to the requesting service.

4.5 Context Maintenance

The context maintenance stage consists of archiving all context data which is more than one month old at the end of each month. This archived data is still accessible to context consumers, however queries on this data are not as fast as the non-archived data which have priority. It is envisaged that a feedback mechanism may also be included within the context maintenance stage in order for the user to provide accuracy ratings for portions of the managed context data.

4.6 Context Disposition

Context disposition is entirely handled by the user. Within ConServ the user is viewed as the owner of all context data which concerns them, therefore they alone have the ability to delete context data. At the moment this is handled via the web-interface, whereby the user selects individual or groups of context data and manually deletes them. In future the users will be able to create rules which would allow for the automatic disposal of their context data according to certain conditions such as time elapsed, location, context source.

5 Proof of Concept

To illustrate how ConServ supports the management of context data throughout the context lifecycle we developed two independent applications; DERI Bluetooth Location System (DBLS) is a *Context Provider* which pushes location data to ConServ and Colleague Finder is a *Context Consumer* which uses the location data provided by DBLS. These two applications illustrate some of the lifecycle requirements implemented within ConServ, in particular the sharing of context data between applications, the enabling of rapid context aware application development, the ability to interact with third parties and the use of privacy policies to protect the user's context data. The evaluation of the remaining lifecycle requirements will be dealt as part of the future work.

The DBLS infrastructure consists of a bluetooth based location scanner deployed in each floor of each wing of the DERI building, as illustrated in Figure 8. The bluetooth scanners are bluetooth enabled Linksys NSLU2s running a modified version of Debian. DBLS monitors the movement of personal bluetooth devices in the DERI building and informs ConServ of any updates. This system enables the locating of a person carrying a bluetooth enabled device to a room within the DERI Building.

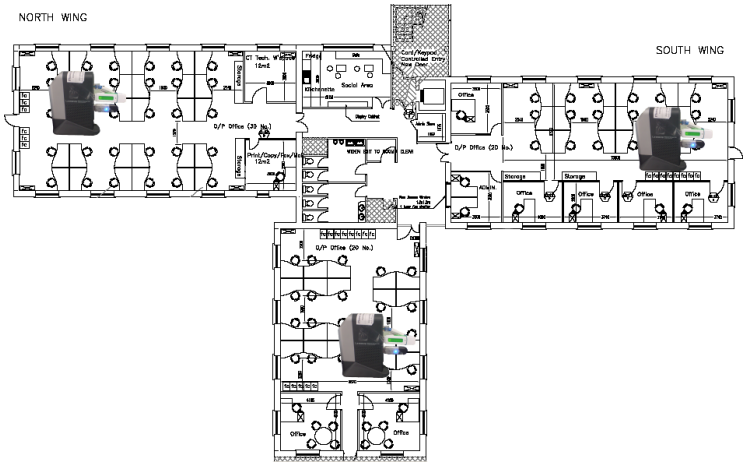

Fig. 8. Linksys NSLU2 Deployment on a Floor of the DERI Building

Fig. 9. User Mark Jones searches for a colleague using "Colleague Finder"

Colleague Finder is a Ruby on Rails web application which allows users to locate a colleague within the DERI building. In the screenshot shown in Figure 9 the user, John Smith, has registered an account with Colleague Finder and added the ConServ URI of his colleague, Mark Jones, to his list of friends. Provided Mark Jones has given Colleague Finder permission to access his location data then John Smith can use Colleague Finder to locate Mark Jones within the DERI building. When Colleague Finder requests location data, ConServ first checks Mark Jones' policy to see what level of granularity was specified for Colleague

Finder. The granularity details are appended to the query, the query is executed and any results are then returned to Colleague Finder. Colleague Finder simply issues a HTTP GET request to ConServ in order to retrieve the context data and the results are returned in XML. Only the HTTP GET request is required to make Colleague Finder context aware as ConServ manages the context lifecycle functionality. At the end of the following month the location data is archived as part of the *Context Maintenance* step and at anytime Mark can *dispose* of his location data within ConServ.

6 Related Work

Recently there have been several efforts to provide web-based context management with the focus primarily being on location data, some examples are FireEagle, Google Latitude, IYOUIT [2] and the applications associated with the OSLO Alliance (Open Sharing of Location Objects) [18]. We will discuss FireEagle due to its early entry into the field, Google Latitude because of its potentially large user base and IYOUIT as a result of its focus on other types of context data rather than just location. Unfortunately none of the three services provide a formal lifecycle which we can directly compare with the WCXMS lifecycle, therefore they will be compared based on the lifecycle requirements outlined at the beginning of this paper.

FireEagle is a location broker which enables users to share information about their location with applications or services in a controlled manner. FireEagle's functionality is limited due to its sole focus on user location data and hence does not fulfill the *extensibility requirement*. It also does not have the ability to *pull location data from third party sources*. However it does provide a strong reference point against which new solutions must be compared and its popularity proves that users have a demand for location aware applications and that users are prepared to provide their location data to web services.

Google Latitude is another example of a location broker which, due to its inclusion in Google's Mobile Maps application, has a large potential user-base. A basic API is provided which allows developers access to a JSON or XML feed containing details of a user's location. Unfortunately, it also suffers from the *extensibility* and *context acquisition* deficiencies of FireEagle and its *privacy policies* are extremely limited as users can only specify full public access or no access at all to their location data.

IYOUIT, developed from the ContextWatcher project [13], is comprised of a mobile phone application which captures as much context data as possible, both automatically and manually, and a web-based context broker which aggregates the data. IYOUIT supports a far greater number of context types than FireEagle and Google Latitude and it represents a significant step towards the creation of a comprehensive personal context data repository. However, IYOUIT currently does not provide access to a developer API, therefore it is not possible to build applications using their data and as a result there is no *interaction with third parties*. An ontology is used to model the context data, however as this ontology

had not been made available IYOUIT cannot be said to provide a *commonly understood data model* and thus is not available for reuse or *extension*.

7 Conclusions and Future Work

In this paper we have presented a lifecycle for web-based context management services (WCXMS) and provided details of an example implementation of that lifecycle called ConServ. A set of requirements for a WCXMS were outlined and existing lifecycles were analyzed to establish their applicability, based on this the WCXMS lifecycle was developed. The details of each stage in the lifecycle were then described by giving details of how they were implemented in ConServ.

The purpose of a WCXMS is to encapsulate the context management lifecycle within a web-service which developers can access. This facilitates the rapid prototyping of context-aware applications by removing the necessity for developers to create a context management component for their applications. By providing Context as a Service (CXaaS), WCXMS allows applications access to context data from a wide variety of sources by enabling the sharing of context data between applications. This allows context-aware applications to move beyond the current situation whereby each application is an island of information with little or no interconnecting bridges.

WCXMS lifecycle implementations, such as ConServ, have the potential to increase the number of context-aware applications available to users while still providing the user with authority over their context data. There is a fine balance which must be maintained between supporting the rapid development of context-aware applications and persevering user privacy. It is our belief that the WCXMS lifecycle maintains this balance through the use of, amongst other things, RESTful interfaces and an extensible context models to aid development and the use of detailed privacy policies and providing complete data control from a user's perspective.

At present ConServ's implementation of some of the lifecycle requirements has not been demonstrated and the development of these proof of concept applications is a key part of the future work. The addition of event-driven functionality is also a priority as at present applications must query ConServ frequently to check for new data. Finally the scalability of the web-service must be evaluated.

Acknowledgment

The work presented in this paper has been co-funded by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2) and by the European FP7 project PECES (ICT-224342).

References

1. Google latitude, `http://www.google.com/latitude/`
2. Iyouit, `http://www.iyouit.eu/`

3. Yahoo! fireeagle, http://fireeagle.yahoo.net/
4. AIIM. What is ecm? http://www.aiim.org/What-is-ECM-Enterprise-Content-Management.aspx
5. Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., Stuckenschmidt, H.: Contextualizing ontologies. Web Semantics: Science, Services and Agents on the World Wide Web 10, 325–343 (2004)
6. Brickley, D., Miller, L.: Foaf vocabulary specification, http://xmlns.com/foaf/spec/
7. Chantzara, M., Anagnostou, M.: Evaluation and selection of context information. In: Second International Workshop on Modeling and Retrieval of Context, Edinburgh (July 2005)
8. Chen, H., Finin, T., Joshi, A. (eds.): The SOUPA Ontology for Pervasive Computing, July 2005. Whitestein Series in Software Agent Technologies. Springer, Heidelberg (2005)
9. Ferscha, A., Vogl, S., Beer, W.: Context sensing, aggregation, representation and exploitation in wireless networks. In: Scalable Computing: Practice and Experience (2005)
10. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Trans. Internet Technol. 2(2), 115–150 (2002)
11. Hayden, E.: Data lifecycle management model shows risks and integrated data flow. Information Security Magazine (July 2008)
12. Kagal, L., Paoucci, M., Srinivasan, N., Denker, G., Finin, T., Sycara, K.: Authorization and Privacy for Semantic Web Services. IEEE Intelligent Systems (Special Issue on Semantic Web Services) 19(4), 50–56 (2004)
13. Koolwaaij, J., Tarlano, A., Luther, M., Nurmi, P., Mrohs, B., Battestini, A., Vaidya, R.: Context watcher: Sharing context information in everyday life. In: Proceedings of the IASTED conference on Web Technologies, Applications and Services (WTAS), pp. 12–21 (2006)
14. Miles, A., Matthews, B., Wilson, M., Brickley, D.: Skos core: simple knowledge organisation for the web. In: DCMI 2005: Proceedings of the 2005 international conference on Dublin Core and metadata applications. Dublin Core Metadata Initiative, pp. 1–9 (2005)
15. Monaghan, F., O'Sullivan, D.: Leveraging ontologies, context and social networks to automate photo annotation. In: Falcidieno, B., Spagnuolo, M., Avrithis, Y., Kompatsiaris, I., Buitelaar, P. (eds.) SAMT 2007. LNCS, vol. 4816, pp. 252–255. Springer, Heidelberg (2007)
16. Niu, W.T., Kay, J.: Location conflict resolution with an ontology. In: Indulska, J., Patterson, D.J., Rodden, T., Ott, M. (eds.) PERVASIVE 2008. LNCS, vol. 5013, pp. 162–179. Springer, Heidelberg (2008)
17. Peterson, M., St. Pierre, E.: Snias vision for ilm. In: Computerworld: Storage Networking World (2004)
18. Siegler, M.: Oslo alliance wants to share location across networks (February 2009), http://www.webcitation.org/5i7Kqzj6d
19. Vatant, B.: Geonames ontology, http://www.geonames.org/ontology/
20. W3C. ical schema, http://www.w3.org/2002/12/cal/icaltzd
21. Wrona, K., Gomez, L.: Context-aware security and secure context-awareness in ubiquitous computing environments. In: XXI Autumn Meeting of Polish Information Processing Society (2005)