

Toward a Shared Model for Wide Area Interoperability of Ubiquitous Computing Environments

Michael Blackstock¹, Rodger Lea², and Charles Krasic¹

¹ Department of Computer Science, University of British Columbia
201-2366 Main Mall, Vancouver, B.C., Canada

² Media and Graphics Interdisciplinary Centre, University of British Columbia
FSC 3640 - 2424 Main Mall, Vancouver, B.C., Canada
{michael@cs, rodgerl@ece, krasic@cs}.ubc.ca

Abstract. Despite many years of ubiquitous computing (ubicom) middleware research, deployment of such systems has not been widespread. For such systems to become more widely adopted, we argue that a shared model for ubicom environments, and a system to bridge such environments between network and administrative domains is needed. This paper presents work in progress toward deriving a shared environment model based on an analysis of several existing ubicom systems. This model is designed for use in an integrated middleware platform used to adapt existing ubicom systems for wide area application interaction. Together the model and platform highlight the benefits of Web Services and Semantic Web technologies for exposing ubicom environments to applications outside administrative domains.

1 Introduction

Over the last several years, researchers have created middleware, toolkits and operating systems to support the development of integrated ubicom environments in homes, meeting rooms and other environments. While these contributions have matured to address many challenges in supporting ubicom application development, they have mostly been confined to lab prototypes. Many reasons have been cited for this lack of progress [1]. Designers may simply be reluctant to deploy systems designed for research prototypes in the real world, however, even when proposed systems have used open standards such as Web protocols, they still have not been widely adopted.

We believe there are two important and related impediments to wider deployment of ubicom environments. One is that, until recently, system designers have focused primarily on supporting applications and user access within single administrative or network domains. While several researchers have suggested the use of a common systems infrastructure to address this [2-4], none so far have been adopted as a standard. Heterogeneity at the interface, communications and abstraction levels should be expected since ubicom systems have evolved from a range of established middleware platforms, and each has its own strengths for building ubicom applications. It is likely too early to agree on a single approach for application development. That said,

until systems can support interaction across domains they will not be widely adopted in the “real world”. The second related issue is that meaningful cross-domain integration relies on a shared understanding of the interactions that may occur between applications and the supporting infrastructure. We need a solution that allows heterogeneity within a single environment while supporting a degree of interoperability across domains.

Our approach is to create a bridge between existing systems to a model for wide area applications interaction with the goal that this model may lead to an interoperable standard. To build this bridge, we can look toward progress in enterprise application integration (EAI). To support the development of distributed enterprise applications, conventional enterprise middleware technology has evolved rapidly. Until recently, a lack of standards at the middleware and component levels between domains has hindered inter-enterprise communications and interoperability. To address this, the enterprise software development community has turned toward the use of Web Services and associated standards efforts. Just as *internal middleware* for enterprise application development matured for single domains, followed by the use of *external middleware* making use of Web Services standards for enterprise integration, we propose that the wider deployment of ubicomp systems for access by applications in other domains will require appropriate *external middleware* for interoperability.

Toward addressing cross-domain ubicomp interoperability, this paper describes work in progress toward a shared model for ubiquitous computing environments and the design of a flexible external middleware platform based on Web Services to *adapt* existing ubicomp systems to this model. This shared model must lend itself to specialization for environment domains such as the home, the office and public places and adaptation to existing ubicomp system abstractions.

The remainder of this paper is organized as follows. In section 2 we present the results of an analysis of some representative ubicomp systems toward deriving a common set of abstractions. Our core model is then described in Section 3. Section 4 presents our design and initial prototype of our external-middleware system called the Ubicomp Integration Framework. Section 5 concludes with a discussion of future work.

2 Analysis

Our goal is to derive a model that finds the right balance between interoperability and suitability for cross domain interactions while maintaining some of the flexibility of a given underlying ubicomp system. To derive our model we began with an analysis of representative ubicomp systems [2-12] in an attempt to find some common ground. Through this analysis we have arrived at the following taxonomy of abstractions suitable for capturing many of the semantics exposed by these systems.

- **Environment Model** that encapsulates the current state of the environment including the components available, the types of context and services components provide and other static and dynamic aspects of the environment.

- **Entities** are base-level abstractions such as people, places, things, groups and activities that can be extended to environment-specific entities such as game players, living rooms, mobile phones and meetings.
- **Context** associated with entities. Context information can include values such as location, temperature, direction, or time. Some systems support the inference of higher level context information such as activities and goals.
- **Services** or functionality associated with entities.
- **Entity Relationships** such as geometric, social, and activity-related relationships between people, places and things.
- **Events** that entities can fire related to a change of state of an entity, for example a door closing or a light turning on.
- **Data or Content** related to an entity. For example, this could include a user's personal notes, or documents associated with a meeting.

At first glance, the notion of an *application* may be one of the abstractions that need to be considered. In Aura for example [2], an application or user *task* is represented as a collection of abstract services that can be transferred between environments to marshal resources for the user. Since our aim is to provide support for both user-centric and environment-centric applications we have not included the notion of an application explicitly in our model, but we intend to revisit this possible abstraction in the future. Table 1 summarizes the results of our analysis.

Table 1. Abstractions exposed by existing ubiquitous computing and context aware systems. ✓ = abstraction clearly exposed, X = not exposed, P = partially or implicitly exposed.

Systems	Env. Model	Entities	Context	Inferred Context	Services	Entity-Entity Rel.	Events	Content Data Storage
Cooltown	X	✓	P	X	P	✓	P	P
Context Toolkit	P	✓	✓	✓	✓	X	P	X
JCAF	X	✓	✓	✓	P	✓	✓	X
iROS	P	X	✓	X	✓	X	✓	✓
Gaia	✓	X	✓	✓	P	X	✓	✓
Active Campus	✓	✓	✓	P	✓	✓	P	P
Aura	P	P	✓	X	✓	P	P	✓
EasyLiving	✓	✓	✓	P	P	✓	P	X
CoBrA	P	P	✓	✓	P	P	P	X
Sentient Computing	✓	✓	✓	P	P	✓	P	X
Sentient Objects	P	P	✓	✓	P	P	✓	X

We have also noted that there are three related aspects to an environment model that must be considered. The *Environment State* aspect consists of entities, entity relationships, and the current state of those entities: current context values and content for example. Secondly, the *Environment Meta-State* consists of entity instances associated with the *types* and *quality* of events, services, context and content. This aspect is required to support introspection. Both the Environment State and Meta-State must be exposed to applications to query the capabilities of an environment and make use of

them. Finally the *Environment Implementation* links entity instances to the specific components that supply the services, context and events associated with entities. These aspects depend on one another and will change over time. The Meta-State depends on the current Implementation, and the current State depends on the Meta-State. In some cases, components associated with an entity like a mobile user can depend on the current situation; the Implementation can depend on the Environment State.

To conclude our analysis, we present the following summary of requirements for a common environment model suitable for wide area application interaction.

- The model should support **interoperability** between different environment types such as the home, the office and public places while also supporting **specialization** for these different environments.
- The model should lend itself to a relatively **straightforward mapping to existing ubicomp systems' abstractions**, finding the right tradeoff between being suitably generic across a wide range of systems but semantically close enough to specific systems for flexibility.
- A common model must support **introspection**, exposing not only the current environment state, but also its current capabilities such as service availability and the types of context supported.
- To support wide area interactions, the model should lend itself to a simple and **coarse-grained access interface** for applications to minimize the communications required over wide area networks.
- The model should **separate exposed abstractions from implementation abstractions**. This separation of concerns will allow dynamic binding of components to entities without application involvement.

In addition to these model requirements we claim that this model should be *executable*, that is, have the ability to be queried and reasoned with [13]. With integrated reasoning a supporting system can maintain the exposed model as its composition changes, simplify integration tasks, and provide missing general purpose capabilities such as context interpretation

3 Core Model

In this section we describe the three aspects of our core model in more detail followed by a simple example to show the relationship between them.

The Environment State aspect consists of entity instances, their relationships, and their current context and content values. A root *Environment* hosts other *Entities* and subclasses of *Entities* such as places, people and devices linked by entity-entity relationships such as *containedIn* or *friendsWith*. Entities have context values retrieved by requesting the value of a *contextAttribute*. The *contextAttribute* property and associated *ContextValue* object may be specialized to support different types of context. Entities may also have content associated with them linked by a sub-property of the *hasContent* property.

The Environment Meta-State aspect associates Entities with the types and quality of events, services, context and content to support introspection. For example, when an Entity has context associated with it, the *hasContext* property will refer to a *ContextType* object. Similarly, Entities may *expose* certain *ServiceInterfaces*. Clients of the model can then call these services as specified in a *ServiceDescription* object that can be specialized to support standard service descriptions such as Web Services Description Language WSDL [14]. The types of events that and content associated with an Entity may also be specified in this aspect.

The Implementation aspect consists of several component types that are typically used to wrap the plethora of sensors, actuators services, and software components in an environment. *EventSources* are components responsible for firing events related to one or more entities and correspond to the Event abstraction exposed by the iROS EventHeap or Gaia Event Manager. For example, an event may be related to a location change, or moving to the next slide of a presentation. *ContextSources* are components responsible for delivering context, low level sensor data or inferred higher level context. This component type corresponds to sensor input components such as Context Widgets in the Context Toolkit. *Services* are components responsible for delivering functionality associated with one or more entity abstractions and their exposed *ServiceInterfaces*. We consider storage and retrieval such as the iROS Data-Heap and the Gaia Context File System a specialized service component. *EntityHandlers* are components responsible for the complete manifestation of one or more entities. For example, an entity handler may be used to adapt a Context Toolkit Aggregator or the *web presence* software representing people, places or things in Cooltown.

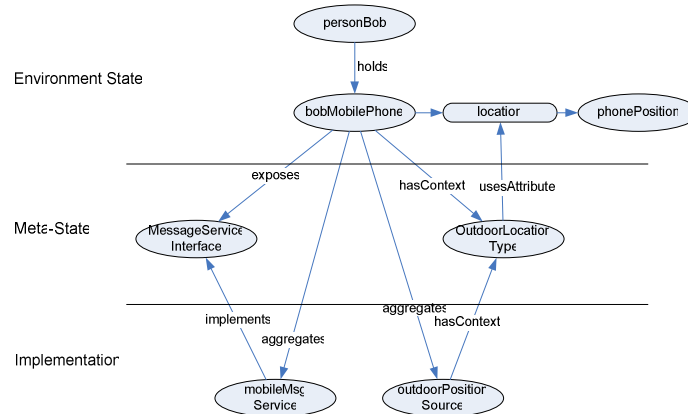


Fig. 1. Model example illustrating dependencies between Environment State, Meta-State and Implementation aspects.

The three aspects of the Environment model and how they relate to one other are illustrated by a simple example shown in Figure 2. A person Entity *personBob* is shown to hold a mobile phone *bobsMobilePhone*. In the current Environment State, this phone has a *location* called *phonePosition* containing the longitude and latitude of the device. In the current Meta-State, *bobsMobilePhone* is shown to *expose* a *MessageSer-*

viceInterface and has the *OutdoorLocationType* of context available for applications. The *OutdoorLocationType* uses the *location* attribute as shown. In the current implementation, *bobsMobilePhone* entity aggregates the *mobileMsgService* Service component and the *outdoorPositionSource* ContextSource component. The *mobileMsgService* implements the *MessageServiceInterface* type, and the *outdoorPositionSource* hasContext *OutdoorLocationType* as shown. This implies that the *bobsMobilePhone* exposes the same ServiceInterface and ContextType to applications.

When applications share a deeper semantic understanding of entities and their associated resources with the supporting infrastructure a much higher degree of interoperability is possible. To address this, we propose the use of *Environment Profiles* to specialize the core model for specific environments. A home profile, for example, can consist of entities such as kitchens, appliances, and living rooms, services for lighting and controlling home entertainment systems, context such as indoor location and temperature. Profiles may be extended further by an integrator to provide extensions specific to a deployment, at the possible expense of interoperability.

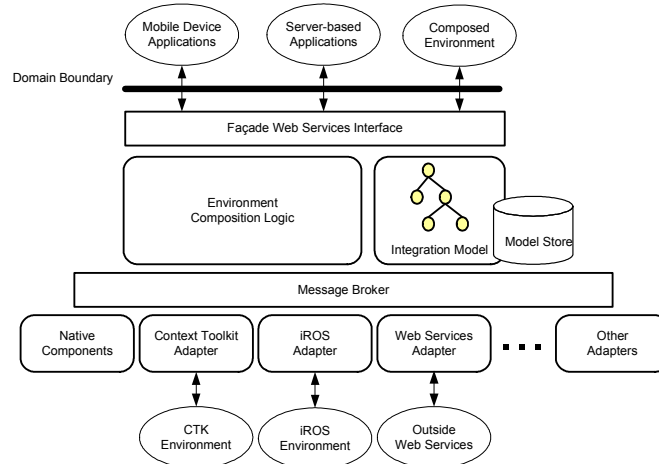


Fig. 2. Proposed high-level ubicomp integration framework architecture

4 Integration Framework

Toward addressing whether our wide area model finds the right balance between wide area interoperability and flexibility we have begun implementation of a middleware platform called Ubicomp Integration Framework (UIF). The high level design of our platform is shown in Figure 2. Unlike other systems, the UIF is designed specifically to adapt existing ubicomp environment middleware to a common wide area model. Our approach is to leverage Web Services for application-environment interaction (Façade Web Services Interface), and Semantic Web standards to describe our model ontologies and instance data, encapsulated in the Integration Model component shown. An integrated reasoning engine is used to maintain all three aspects of our

environment model and support flexible queries about the environment composition. The Environment Composition Logic is responsible for determining which Adapter and associated internal ubicomp system will handle a specific operation: to invoke a service, get context, or subscribe to events.

From our initial prototype we have found that semantic web technologies are useful as a modeling tool for expressing our environment model. The integrated reasoning engine can help maintain the environment model as it changes, maintain consistency, and perform validation checks. The same engine may be used to change component aggregations based the current situation, and as a general purpose context interpreter [7]. We have also found that wrapping an environment using web services has several advantages. Applications and user interfaces can be developed independently of the ubicomp environment, allowing integrators to focus on the resources their environment exposes without concern for user interface issues.

In an effort to assess the work involved in integrating existing ubicomp systems, we have begun work on adapters for the Context Toolkit [9], iROS [5] and the Equator Component Toolkit [15] for our system. So far we have found the difficulty of integrating a given ubicomp environment depends on how closely our component abstractions match those of our model. For example, it seems somewhat awkward to map a Service method calls to changing ECT component properties. The ECT component model, based on Java Beans, uses named properties to control components that manage physical sensors and actuators. An ECT Camera component defines the following properties:

- *configCaptureDevice* sets up the physical camera to use
- *triggerImageCapture* changing this property causes a picture to be taken
- *imageLocation* when the photo is ready, this property is changed to the URL of the image.

When a service method to take a picture is called by the UIF, this method call is routed to the ECT Adapter. The Adapter sets the *configCaptureDevice* property then changes the *trigger* property. The ECT camera component then reads the configuration property (service arguments), and takes the picture. The URL of the photo is written to the *imageLocation* property by the component which is read by the UIF Adapter and returned in the method call.

5 Related Work

Unlike the systems mentioned in our analysis [2-12] and integration work [15], our proposed model aims to unify the abstractions they have established as useful into a *coarse grained* model suitable for wide area application interaction. While other ontology-based context aware systems [7, 16] model context using ontologies, the UIF uses ontologies and reasoning to maintain relationships between abstractions such as entities, interfaces, context and the components of integrated ubicomp systems. Unlike the Context Fabric [17] that proposed an infrastructure for querying, and subscribing to context, the UIF aims to provide wide area abstractions for ubiquitous

computing environments, not only context, using standard wide area service protocols such as Web Services. This allows integrators to maintain the administrative and network boundaries of these environments while facilitating application interaction with hosted services and sensors.

6 Conclusions and Topics for Discussion

In this paper we have proposed a shared common model for ubicomp environments based on abstractions exposed by existing ubicomp systems and our design of *external middleware* to adapt existing ubicomp systems to this model for wide area access. In this workshop we hope to discuss the following open questions related to our work:

- **What are the most important properties of wide area ubicomp and does our model capture them?** Since our derived abstractions are based on an analysis of several existing systems, we expect that a model based on them will service a wide range of applications, but it is not yet clear that our model is ideal for *wide area* applications. Awareness of the most important properties of wide area ubicomp will help qualify the suitability of our model.
- **What are the key tradeoffs between flexibility, ease of programming and the constraints imposed by wide area networks and a shared environment model?** The introduction of our new level of abstraction will (necessarily) hide the details of an underlying ubicomp environment, but likely at the cost of flexibility in terms of the classes of applications that can be supported. Similarly the constraints of wide area networks such as latency and bandwidth constraints may affect the types of applications possible. Understanding those tradeoffs will be valuable in evaluating our solution.
- **What data needs to be exchanged between domains and how will these map to wide area interactions?** For example, how could we deliver mouse movement events to control a desktop PC or access high bandwidth multimedia content across domains? If we refer to other special purposed protocols for applications that can use them, will this lead to a proliferation of alternative interfaces?

In closing, we'd like to acknowledge the feedback Adrian Friday provided on this overall research direction.

References

1. Davies, N., Gellersen, H.-W.: Beyond Prototypes: Challenges in Deploying Ubiquitous Systems. *IEEE Pervasive Computing*, Vol. 1 (2002) 26-35
2. Sousa, J.P., Garlan, D.: Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. *Proceedings of the 3rd IEEE/IFIP Conference on Software Architecture*. Kluwer, B.V. (2002)
3. Roman, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K.: A Middleware Infrastructure for Active Spaces *IEEE Pervasive Computing* 1 (2002) 74-83
4. Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B.: People, places things: Web presence for the real world. *Third IEEE Workshop on Mobile Computing Systems and Applications* Monterey, California (2000)
5. Ponnekantia, S.R., Johanson, B., Kiciman, E., Fox, A.: Portability, extensibility and robustness in iROS. *Proceedings of IEEE International Conference on Pervasive Computing and Communications*, Dallas-Fort Worth (2003)
6. Brumitt, B., Meyers, B., Krumm, J., Kern, A., Shafer, S.A.: EasyLiving: Technologies for Intelligent Environments. *Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing*. Springer-Verlag, Bristol, UK (2000)
7. Chen, H., Finin, T., Joshi, A., Kagal, L., Perich, F., Chakraborty, D.: Intelligent Agents Meet the Semantic Web in Smart Spaces. *IEEE Internet Computing* 8 (2004) 69-79
8. Griswold, W., G., Boyer, R., Brown, S., W., Truong, T.M.: A component architecture for an extensible, highly integrated context-aware computing infrastructure. *Proceedings of the 25th International Conference on Software Engineering*. IEEE Computer Society, Portland, Oregon (2003)
9. Dey, A.K.: Providing Architectural Support for Building Context-Aware Applications. *College of Computing*. Georgia Institute of Technology (2000)
10. Biegel, G., Cahill, V.: A Framework for Developing Mobile, Context-aware Applications. *Second IEEE International Conference on Pervasive Computing and Communications* (2004) 361
11. Adlesee, M., Curwen, R., Hodges, S., Newman, J., Steggle, P., Ward, A., Hopper, A.: Implementing a sentient computing system. *IEEE Computer* 34 (2001) 50-56
12. Bardram, J.E.: The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications. *Pervasive Computing: Third International Conference*. Springer Berlin / Heidelberg, Munich, Germany (2005)
13. Tetlow, P., Pan, J.Z., Oberle, D., Wallace, E., Uschold, M., Kendall, E.: Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering. *W3C* (2006) <http://www.w3.org/2001/sw/BestPractices/SE/ODA/060211/>
14. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language. *W3C* (2001) <http://www.w3.org/TR/wsdl>
15. Greenhalgh, C., Izadi, S., Mathrick, J., Humble, J., Taylor, I.: ECT: a toolkit to support rapid construction of ubicomp environments. *Proceedings of UbiComp '04 (Demonstration)*. Springer, Nottingham (2004)
16. Gu, T., Pung, H.K., Zhang, D.Q.: Toward an OSGi-Based Infrastructure for Context-Aware Applications. *IEEE Pervasive Computing* 3 (2004) 66-74
17. Hong, J.I.: Context fabric: Infrastructure support for context aware systems. *CHI '02 extended abstracts on Human factors in computing systems* ACM Press, Minneapolis, Minnesota, USA (2001)