Deployment Experience Toward Core Abstractions for Context Aware Applications

Matthias Finke[1], Michael Blackstock[2], and Rodger Lea[1]

[1] Media and Graphics Interdisciplinary Centre, University of British Columbia
FSC 3640 - 2424 Main Mall, Vancouver, B.C., Canada
[2] Department of Computer Science, University of British Columbia
201-2366 Main Mall, Vancouver, B.C., Canada
{martinf@ece, michael@cs, rodgerl@ece}.ubc.ca

Abstract. Despite progress in the development of context aware applications and supporting systems, there is still significant diversity in the models and abstractions they expose. This work describes an effort to gain a better understanding of the situation and develop a core set of abstractions by deploying several context aware applications, using a rapid prototyping platform. From this experience we propose and demonstrate a set of abstractions shown to be useful for a range of context aware applications. Combined with a survey and analysis reported elsewhere [1] we then provide an analysis toward providing a core set of abstractions that we argue can be used as the basis for modeling many context aware systems, including not only context, but other aspects such as entities, their relationships and associated events, services and content. We then provide several practical lessons learned from the use of our model and abstractions during analysis and our iterative platform development process.

1 Introduction

Despite significant experimentation and deployment of context aware platforms and applications over the last 15 years, there is surprisingly little agreement on core abstractions and models for such systems. Individual research groups have developed abstractions suited to their application or research target [2-5] and often built bespoke systems to implement these [6-12]. While there is some overlap in the models and abstractions they have developed, there is also significant diversity. In an attempt to understand this situation, and in particular to try and develop a core set of common abstractions for context aware applications, we have, over the last two years, taken a dual research approach. First, we have surveyed and analyzed a set of key ubicomp systems with a goal of identifying abstractions and models in support of context aware applications. Secondly, and in parallel, we have implemented and deployed a set of context aware applications using a rapid prototyping platform with a goal of using practical experience to design, experiment with and validate core abstractions and models suitable for a range of context aware services. We have reported on the survey and analysis elsewhere [1]. This paper reports on our experiences developing and deploying four context aware applications and the underlying evolution of our platform as we improve our abstractions and model.

Our work has been carried out within the framework of the Mobile Multimedia urban shared experience (MUSE) project, a multi-disciplinary research project focused on exploring mobile multi-media services suitable for an urban environment. A key aspect of this project is its use of context aware services and its focus on real-world deployments [13]. To date Mobile MUSE has explored a variety of services deployed using traditional carrier networks as well as experimental WiFi based infrastructure and WiFi enabled cell phones. These include location aware games [14, 15] context aware tourist guides [16], tagging and folksonomy applications [17] as well as local event support services such as location based film festival services. Although Mobile MUSE has a strong technology and deployment focus it also includes significant research on business and sociological aspects of context aware services [18]. Within Mobile MUSE, the work of the MAGIC lab at the University of British Columbia (UBC) has primarily been to explore advanced services that exploit broadband wireless networks. As a basis for this research, we have developed the MUSE context aware platform (MUSEcap) and deployed a variety of services across the UBC WiFi network, one of the largest campus WiFi networks in North America with over 1700 access points.

1.1 Background and Motivation

While there have been many research applications and systems developed for places such as tourist destinations [19], campuses [20], meeting rooms [21], homes [9], and hospitals [22], there has been little consensus on the high level abstractions exposed to context aware applications from supporting platforms. With the wide variety of research and commercial systems available, using the same system for all context aware application domains is not realistic. One environment may differ significantly from another in terms of the entities (people, places and things) and the capabilities such as context, and services available. Researchers have justifiably proposed and built systems deemed important for different context aware application types and paradigms.

That said, when the same systems are used in different situations, practitioners have shown it is possible to seamlessly move user tasks between domains allowing them to make use of resources there [11]. However, given the variety of systems available and their specialization for different domains, using the same system in all places is not realistic. To address this, practitioners have demonstrated that data and control level interoperability can be achieved using various techniques. For example, the use of an intermediary such as the Patch Panel [23] to transform control messages as they flow through the iROS Event Heap [3] has been shown to be useful in addressing control flow interoperability. Component oriented systems like Obje/SpeakEasy [5] and the Equip Component Toolkit [24] have shown that the use of a small, standard set of component interfaces with mobile code or the use of component properties can allow users to configure components to interoperate. Friday et al [25] demonstrated the provision of an abstraction layer on top of heterogeneous service architectures from within the infrastructure, while the ReMMoC system shows that it is possible to provide a generic service abstraction in device-side middleware [26] to address service interoperability. Henrickson et al. designed a model for context [4] as the basis for a context aware application supporting infrastructure [10].

Based the experience outlined in this paper and previous analysis [1] we believe that it is possible to express the run time environment of *any* context aware application or supporting system using a set of common abstractions. We can use these abstractions for analysis and design, and to provide an abstraction layer for not only heterogeneous context producers or service infrastructures, but on the environment's computing resources as a whole. Our immediate aim is to develop a common model that we can use as a base set of abstractions for our MUSEcap platform for a variety of context aware applications.

This paper provides 3 key contributions: (1) it proposes a common model and abstractions we believe are suitable for a variety of context aware applications and services, (2) it validates this model and abstractions using deployed context aware applications and (3) it offers practical lessons learned from real world deployment and several iterations of the underlying systems platform.

This paper is organized as follows: In section 2 we discuss two initial context aware applications/services we have developed and explore the abstractions they needed. In particular, we explain the evolution of the abstractions to support increased application functionality and our experiences balancing abstractions against domain specific services. In addition, we briefly outline the implementation of our architecture and how we supported our core abstractions. In section 3 we present two further prototype deployments we used to validate our initial model and explore its ability to support a range of context aware services. In section 4 we combine the results of our practical experience with our parallel survey and analysis of existing context aware platforms to propose a more generic common model for context aware services. We then relate this common model to our practical experiences and discuss the mapping between our initial abstractions and those supported by the common model. In section 5 we discuss some of the lessons we have learned during this practical investigation with a particular focus on two issues; the tension between core system abstractions and domain specific services and secondly on the drawback of a purely practical approach to exploring and developing common abstractions and underlying system models. Finally in section 6 we conclude and discuss future work.

2 Context Aware Model and Abstractions: Evolution

We began our practical experimentation in late 2005 with the development of a simple location aware game played by teams on campus. Our goal was to develop an initial platform for context aware services driven out of the application needs. During early 2006 we deployed the service and used our experiences to refine our model and evolve our implementation. We deployed our second context aware service in summer of 2006 again using our experiences to refine the model and improve the implementation. We adopted a standard web services architecture for our underlying platform to ensure rapid prototyping and ease of development. Below we discuss the two deployments and the platform development.

2.1 CatchBob! to the Fugitive

The Fugitive [27] is a mobile multi-user location based game that extends the functionality of CatchBob! [28]. The game is played on a university campus by a team of three using Tablet PCs. The team attempts to locate a virtual character (the Fugitive) that is initially hidden on a digital map displayed on each participant's Tablet PC. The playing field (digital map) on the Tablet PC shows every player's present position while providing visual cues to signal one's proximity to the Fugitive.

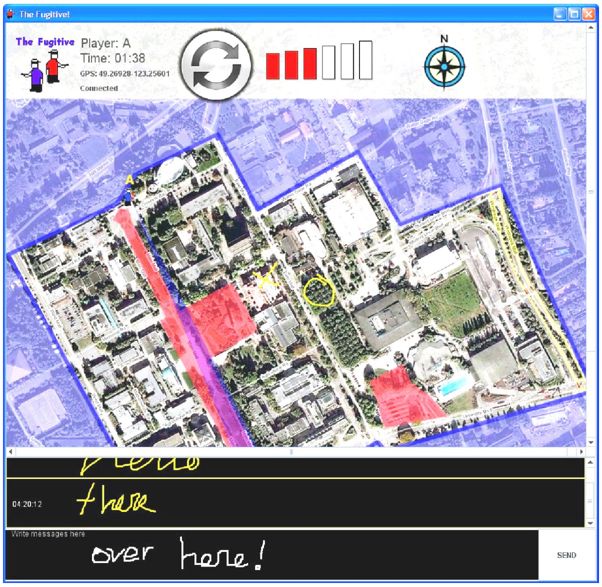


Fig. 1. The Fugitive user interface

The goal of the game involves two parts, a *catch* phase and a *chase* phase. In the catch phase, players physically move around the campus with their position being updated accordingly on their digital map. The objective is to trap the Fugitive by physically forming a triangle with the team members. When the triangle area has been reduced to a certain size by the team, the Fugitive becomes visible and starts to move to other locations on campus: the chase phase begins. In the chase phase, participants re-position themselves on the digital map to chase and trap the now visible, moving Fugitive by again forming an even smaller physical triangle than before. Map and ink messaging are available to enable communication in the game. Communications are augmented by auditory beeps to alert players of incoming messages from other teammates. Figure 1 shows a screenshot of the application installed on a Tablet PC.

2.1.1 Discussion

The Fugitive has a very simple set of system abstractions as shown in Figure 2. The core abstractions include the notion of users (with the Fugitive itself a special case) and context - primarily location information. A containment abstraction, the environment, was added to provide a framework for the overall application. Initially we considered associating communications, that is, ink messages with a particular user, but eventually placed this as a service associated with the environment since map annotations and ink messages are broadcast to all users in the environment.

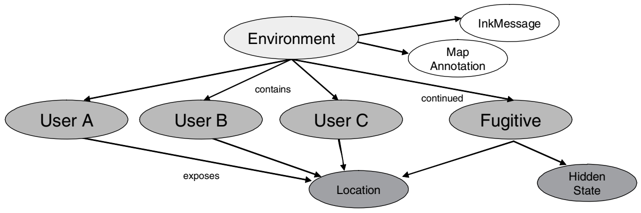


Fig. 2. The Fugitive environment model

2.2 PlaceMedia

PlaceMedia is a more ambitious system designed to explore how users can create and share media via the notion of location (or place). It allows users to define and maintain their personal profiles, to create and manage contact list of friends, to communicate with friends using instant messaging and to share media with friends. The physical location of all friends is visualized on a digital map as part of the user interface along with their profile and present status (i.e. online, busy, offline, etc.) as shown in Figure 3. Instant messaging enables friends to see each other's presence and to chat with each other while moving around campus.

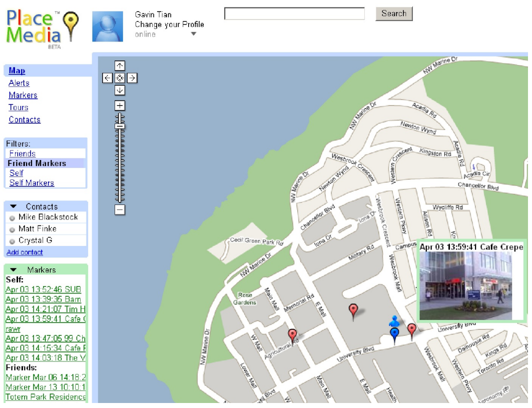


Fig. 3. PlaceMedia Tablet PC user interface

Users can create multimedia artifacts that are combined with context data such as location, creation time etc. This content can then be uploaded on a server to share with friends. When media is shared, icons representing the content on the digital map indicate the place where artifacts were created. One key feature of the PlaceMedia application is support for *Context Sensitive Alerts*. These alerts are triggered by a context variable, e.g. person or place, and can be used to build dynamic context aware applications. For example, with our prototype users can create an alert containing a multimedia message combined with a specific place (e.g. a coffee bar) and a proximity variable. Other users will get the alert when they approach the predefined location (i.e. within the proximity distance). Our location tracking subsystem supported either GPS or WiFi triangulation using Intel's PlaceLab [29].

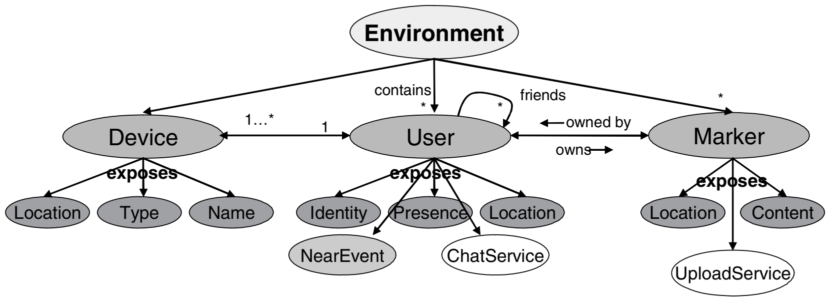


Fig. 4. PlaceMedia environment model

The abstractions exposed by the PlaceMedia prototype extended the simple Fugitive model as shown in Figure 4. The Fugitive game was designed for a specific device, the Tablet PC, and so there was an implicit mapping between the device and the user. In the more generic PlaceMedia system, we aimed to support multiple devices for a given user, and so we introduced *Devices* as a core abstraction in our model and provided support for a one-to-many relationship between *User* and *Device* entities. *Device*s now expose their location context rather than the end user. Where the Fugitive game structure defined a fixed number of users, in PlaceMedia we support an unlimited number of users. The PlaceMedia application required several new types of context such as device location, type and name, user identity and presence. In addition to these simple context types, each user in PlaceMedia has a friends list or a *roster* to relate friends to one another. Communications in PlaceMedia is now supported through a *Chat* service associated with a *User* entity. Messages are directed to a single user, rather than broadcast to all users in the environment as in the Fugitive.

The notion of *place* was another important abstraction introduced into PlaceMedia so that content like photos, videos and text could be left by users at particular places. To support these content-enhanced places, we introduced a *Marker* abstraction that not only exposes its location context but also allows users to associate multimedia artifacts such as pictures or video recordings with these Markers. To support context-sensitive alerts, an event abstraction called a *NearEvent* was added to the model. This event, associated with a user, supports the subscription of events that are fired when a user is within a specified range of another user, or a Marker. Finally, we added two services to facilitate direct user communications and to upload marker content. To reflect that Chat is between two users, the Chat services are associated with users involved in communications. A user can *retrieve* messages exchanged with another user, and *send* messages to another user. To send messages to many users, they must be sent individually in contrast to the broadcast model in the Fugitive.

2.2.1 Discussion

As can be seen from Figure 4, the core abstractions of PlaceMedia have been expanded and are more generic than the original Fugitive model. The types of entities and our support for context has also been expanded to include device, user and marker, position, identity, type, and content. The Marker entity is interesting in that it elevates location to a first class object. Typically in context aware systems, location is a key context attribute associated with users or other objects. However, the PlaceMedia application forced us to rethink location – in some cases it is simply a context attribute like a longitude and latitude, but in others it constitutes a first class object in its own right. We found that the PlaceMedia application required friendship and ownership relationships between registered users and markers to more easily find other relevant entities in the system for display. Another key abstraction that PlaceMedia required was the notion of context sensitive alerts which we generalized through an event system.

2.3 MUSE Context Aware Platform Implementation (MUSEcap)

MUSEcap was developed using the JBoss [30] Java 2 Platform Enterprise Edition (J2EE) [31] application server following a classic three tier system architecture; a simplified system diagram is shown in Figure 5. As is typical in three tier architectures, the top tier is for presentation and user interface, the middle tier for functional processing logic, and the bottom tier for data access. This architecture allows the user interface, application logic and database to change independently without affecting the other tiers.

To map the PlaceMedia system model (see section 2.2) to this architecture, we used standard entity-relationship modeling techniques[1] for the persistent data in our system. This included database tables for *Markers*, *Users*, *Devices*, and the relationships between them such as *friendship* and *ownership* (see Data Tier in Figure 5). We then refactored the single application-specific PlaceMedia interface into several, more general purpose interfaces in the logic tier, with each corresponding to an entity in the system such as Users, Device and Markers for easier reuse in subsequent applications. These interfaces provide functions to *find*, *add* and *remove* the entities they handle (e.g. Users and Devices). Methods are also provided to *get* and *set* context values such as location and presence, to call the associated service methods such as sending and retrieving messages, and to subscribe to events. The SessionBean logic accesses objects in the data tier wrapped using J2EE EntityBean interfaces. In the presentation tier we provided servlets, HTML and Java Server Pages (JSP) and a J2ME based application for access using a Tablet PCs, handheld PCs, and mobile phones.

[1] http://en.wikipedia.org/wiki/Entity-relationship_diagram

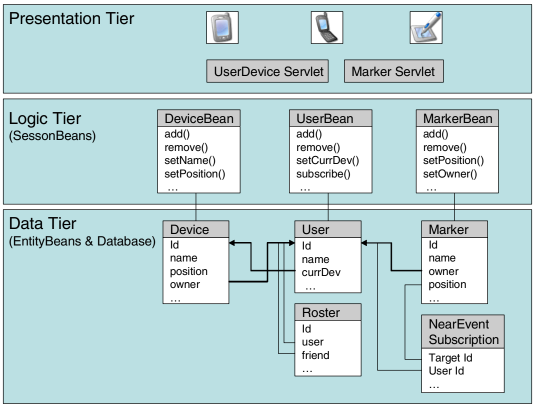


Fig. 5. Simplified MUSEcap architecture

3 Validating the Model Through Additional Prototypes

To evaluate MUSEcap and its associated abstractions and model we developed two additional applications. The first application developed and deployed in late 2006 focused on the notion of a *Tour* while the second application called *MoCoVo* supports mobile social networking and was deployed in spring of 2007.

3.1 Tour Prototype

The Tour prototype supports the notion of tours: a set of connected locations, organized around a theme and employing media and context data to guide people from location to location. This prototype was created as a logical extension of the Place-Media application and as another instantiation of the canonical 'tour guide' context aware application [19].

We created a mobile phone application using Java Micro Edition (J2ME) [32] that automatically records location and content captured by the end user over time. The idea is to allow friends to create tours for each other using mobile phones. Users can see these tours on the screen, download them to their mobile phones, and then "play back" these downloaded tours when walking around later. This could be extended to support hikers, cyclists or tourists who carry a mobile device. The application tracks their location and allows them to annotate places with media such as photographs, video clips and audio commentaries.

When the trip is captured, the user can upload the tour data to the platform. A digital map visualization is created on a web page that shows the locations and content recorded during the tour. Icons are placed on the map along the tour path to indicate where multimedia artifacts were created. A simple web-based editing tool allows tour creators to edit, add locations and media after the basic location data has been captured. Finally, any tour can be published and can be downloaded to a mobile device and followed by others. Friends can visualize tours, and download them to their mobile phone to follow a tour. During tour playback on a mobile phone, location is again tracked to trigger play back of media when a user is within a defined distance of a tour location. In addition users can add media to an existing tour thus building up a richer tour narrative over time.

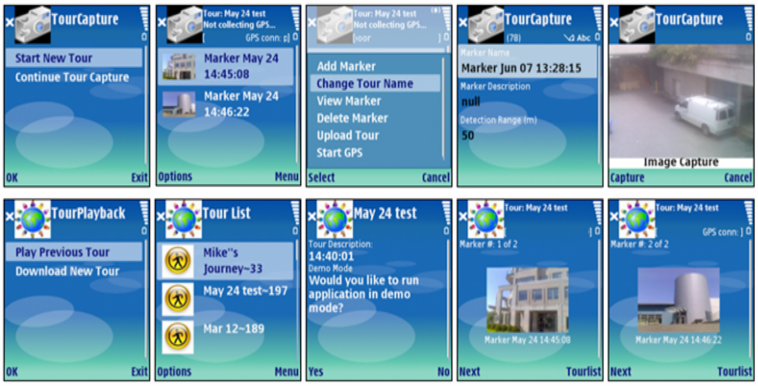


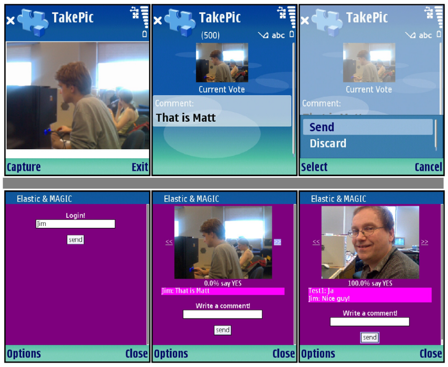
Fig. 6. Mobile phone tour prototype user interface. The tour capture interface is shown along the top row, and the tour playback along the bottom row.

3.1.1 Discussion

The purpose of our Tour prototype is to attempt to reuse the core abstractions developed for the Fugitive and PlaceMedia and implemented in MUSEcap in an effort to assess how useful these abstractions were in building additional applications. Despite the tour prototype's additional requirements, we found that the abstractions provided by PlaceMedia did facilitate reuse. The tour application made use of users, markers, and associated context including location and content provided by the MUSEcap platform, requiring no changes to the underlying model or abstractions. However, the Tour application did require new application-specific code for grouping Markers into tours, adding support for paths between markers, and the association of content with these paths, and services for uploading and downloading tours.

3.2 Mobile Comments and Voting (MoCoVo) Prototype

MoCoVo is a social networking application that allows groups of users to share media via their phones and to network based on the media. One of the major objectives of the application idea is the creation and sharing of new multimedia artifacts paired with group-based communication support. Users can easily create a new group and invite friends to join it. Such a group has a more of dynamic character than our friends roster in PlaceMedia and is designed to support dynamic ad-hoc groupings such as special interest groups organized around events or locations. For instance, a group of people visiting "New York" could create a group that shares pictures (including location context) and comments about the city – obviously the grouping is dynamic and time limited to the visit.



That is Matt

Jim: That is Matt

Test1: Ja
Jim: Nice guy!

Fig. 7. Mobile phone interface for the MoCoVo application. The first row illustrates taking and commenting on a picture, the second row browsing, and voting on a picture.

Once part of a group, users are able to create media, such as pictures, using their mobile phone and write comments or tag them before uploading to the server for sharing within the group. Once shared on the server any group member can access the media and download it back on their phones. While browsing pictures or other media, group members have the ability to comment and vote on each media clip. Current vote results are presented with media while browsing. Furthermore, each member of the group can provide comments/tags that will be associated with the pictures and can be accessed by others.

3.2.1 Discussion

Despite the different application domain, we were able to reuse several facilities of MUSEcap - specifically user management, and content uploading, in the development of MoCoVo. We decided to refactor the platform to support Content as a separate entity in itself, associated with either Users or Markers, recognizing that content may be associated with other entities in the environment, not just Markers. Group entities were also added to the application to contain multiple users. We added support for ranking, comments and tags context types associated with Content. While there were some changes made to the platform to support MoCoVo, most of the development work of this prototype was spent on the client application and only limited work on the server side implementation was needed.

While the Tour and MoCoVo application highlighted the fact that our core model and abstractions were adaptable to a range of context aware applications we realized that to go beyond this class of applications, we would need to extend and generalize our model further. We must be able to easily add new reusable entity types such as social groups, and new capabilities such as a "ranking" context, voting services and text comments that can be reused by new classes of social networking applications. To accomplish this we found the need to generalize our context aware application models further. We describe our analysis toward these general abstractions in the following section.

4 Analysis

During the development of our context aware applications, we found that we could generalize our abstractions further for greater reuse during analysis and in future platform iterations. The Fugitive, for example, can be modeled as a game hosting a set of players and the Fugitive itself. If we generalize a game to an *environment*, and the users and fugitive as *entities*, the Fugitive environment hosts several entities: the players of the game (users), and the Fugitive, a virtual agent. To generalize further, an *environment* hosts *entities*.

The user entities in the Fugitive expose context: their location. The agent entity has two types of context: its hiding state, and current location. In general, entities like users *expose* different forms of *context*. Finally, the Fugitive platform provides several *services*. In the case of the Fugitive, these are most obviously associated with the environment as a whole: a map annotation service, and an ink chat service to facilitate group communications. To generalize, the *environment* entity *exposes* these *services*.

Placemedia extended the capabilities of the Fugitive. Like the Fugitive it included mobile user entities in a campus environment model, but we also introduced Device entities as shown in Figure 4. PlaceMedia users expose not only location, but a presence-state context associated to indicate their instant messaging status (on-line, away, busy), and information about registered users' identity. Marker entities were added to the system to mark places of interest on the campus. Markers have static location context and content associated with them. In PlaceMedia, Devices, and Markers entities are owned by specific users. That is, there is an ownership *entity relationship* between PlaceMedia users, and their Devices and Markers. Similarly, users can be friends with one another, in this case a social entity relationship using the roster functionality. PlaceMedia also introduced support for event notifications that are signaled when one user is near another, or near a Marker on the campus. Event support has been found to be useful in supporting follow-on applications such as the tour to notify the user when a tour marker is nearby. In general events are also a capability associated with an entity.

While the Tour application was able to reuse specific entities and exposed capabilities in MUSEcap, the MoCoVo application required more changes to the platform to satisfy its new requirements. For a platform to support MoCoVo, the Tour application, PlaceMedia and the Fugitive without changes to the system's API, we need to move toward a more general set of abstractions.

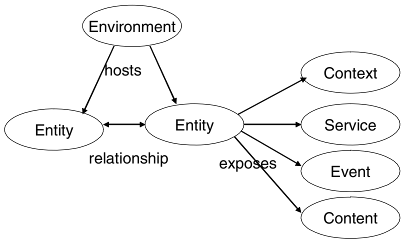


Fig. 8. Core abstractions of the Ubicomp Common Model

Based on the experience outlined in this paper and on an analysis of other ubiquitous computing systems [1] we have identified the abstractions common not only to PlaceMedia and follow on applications, but to a significant number of infrastructure-based context aware applications independent of the environment domain and infrastructure used. These abstractions represent a more generic model than those supported in MUSEcap and can be summarized as follows:

- **Environment Model** that encapsulates the current state of the environment including entities in the environment such as users and devices, the types of context and services components provide, and other aspects of the environment as a whole.
- **Entities** are base-level abstractions such as people, places, computing devices and other things, groups and activities. They can be specialized to environment-specific entities such as game players, living rooms, class rooms, mobile phones and meetings.
- **Entity Relationships** such as location, social, ownership, and activity-related relationships between people, places and things.
- **Context** associated with entities. Context information can include values such as location, status.
- **Services** or functionality associated with entities such as users, places and devices. Ubiquitous systems either provide their own service infrastructure or build on existing middleware systems.
- **Events** that can signal a change of state such as a person entering a room, a light turning on or a presence state change (online to offline).
- **Data or Content** related to an entity such as a place, user, or activity. This could include a user's personal photo, or an audio description of a location.

To summarize, all of the context aware applications outlined in this paper expose an environment model to the user, hosting the entities (people, places and things) that are relevant to the application. These entities have capabilities associated with them, where capabilities are the types of context, relationships, events, content or services they expose to the application. The abstractions and how they relate to one another in an environment model are illustrated in Figure 8.

The deployment of these applications and a survey of existing systems has informed the design of a general model for all ubiquitous computing environments called the Ubicomp Common Model [1]. The UCM not only describes entities and their capabilities as in our analysis, but three related aspects of a context aware environment. The first aspect, called the *Environment State* contains entity instances, their types, and current context values. The *Environment Meta-State* aspect contains entity types and their *capabilities*: the context, events, relationships and services they expose. The *Environment Implementation* aspect contains information about the components of an underlying system that implement the capabilities described in the Meta State. For example, if a location context is supported by User entities, a SessionBean in the PlaceMedia system is the component that will implement this context.

With the UCM and a supporting platform called the Ubicomp Integration Framework (UIF) we have begun to integrate the MUSEcap capabilities described here with other context aware systems [6, 24, 33] into a single environment model in an effort to evaluate our model's suitability for addressing interoperability and portability across smart spaces. The UIF has also been designed to support native context aware development, where the environment model and implementation components are hosted by the UIF itself, rather than an integrated system. Due to space limitations, for more information please refer to [1].

5 Lessons Learned

Based on our experience developing our applications and platform, and our subsequent analysis, we have derived some useful guidelines for context aware application and systems designers related to the use of our abstractions, and platform deployment experience summarized here.

Abstractions: Model applications and systems as environments. We have found that it is possible and useful to model all of our systems as an environment that hosts entities and their capabilities. We have found this technique to be valuable in determining how the requirements of applications supporting systems are similar or different since the model highlights the new entities and capabilities required by a context aware application. We have also found it to be valuable in designing an API for context aware applications that is independent of the underlying sensor and service implementations [1].

Abstractions: Associate capabilities with entities. We have found that it is possible to not only associate context with an entity as suggested by Dey et al. [34], but also the other capabilities of an underlying system, such as the services, events, and content. While the location of a user is obviously related to the user entity, the chat service and “near event” should also be associated with the users they serve. When a service is not obviously associated with one entity in particular, such as a user, place or device, it can be associated with the environment entity as a whole, as is in the case for the map annotation and ink chat facilities of the Fugitive.

Abstractions and Platform: Clearly differentiate application-specific services from reusable platform abstractions. During our iterative development of MUSEcap we initially implemented a requirement as part of the PlaceMedia application, only later realizing that it was more appropriate as a reusable MUSEcap platform service. For example, the Marker entities were developed as part of the PlaceMedia abstraction, and then were found to be a useful abstraction for the Tour application. We recognized that the supporting platform, such as the enterprise application platform we used, should provide a general purpose and extensible facility for providing a range of reusable capabilities. This way, application-specific services can be migrated into the general purpose platform when they have been found to be reusable by more than one application.

Platform: Traditional three tier enterprise application environments are excellent rapid prototype platforms. There is a tendency in the context aware and ubiquitous computing (ubicomp) systems community to develop bespoke platforms in support of experimental context aware applications. Our experience has shown us that by using existing web services and enterprise application infrastructures (e.g. J2EE and JBoss) and standard 3 tier architectures, we are able to rapidly develop and more importantly, evolve and change, our underlying platform, MUSEcap. While we understand the eventual need for more sophisticated middleware systems, we were surprised by how far a more standard systems environment could carry us. Given their ease of development, reliability, sophisticated tool chains and rapid deployment support we believe that such development environments have a stronger role to play in the systems and middleware community.

Platform: Development of architectural models through practice needs to be augmented with survey and analysis. Our exploration of a context aware platform and common abstractions in support of context aware applications has reinforced the lesson that prototyping and real world deployment is necessary but not sufficient to develop generic system abstractions. The breadth and scope of the ubicomp space dictates that any group will struggle to prototype context aware applications which cover more than a small subset of the possible application domains. It is crucial therefore that when considering common models and abstractions, a comprehensive survey and analysis of existing systems and applications is carried out. While this lesson may be perhaps obvious it is surprising how many ubicomp systems are developed that provide little evidence of wide applicability.

6 Conclusions

From the deployment experience described here we have found that several context aware applications and their supporting infrastructure can be described by an environment model consisting of related entities (people, places, things), and their associated capabilities as described in our analysis. Furthermore, we have implemented this model using a standard three tier architecture. We have described the development of an evolving platform to deploy four context aware applications, and how the features of these systems can be categorized into one of seven high level abstractions: an environment model, entities, entity relationships, context, events, services and content.

With this work in parallel with an analysis and survey of other ubicomp systems [1] we have designed the Ubicomp Common Model (UCM), a comprehensive model for context aware computing environments. We have provided some guidelines into how to use this model for context aware systems analysis, and lessons learned based on our deployment experience. In future work we aim to evaluate the model and integration system further by integrating several existing ubicomp systems into a composite environment model.

References

1. Blackstock, M., Lea, R., Krasic, C.: Toward Wide Area Interaction with Ubiquitous Computing Environments. In: Havinga, P., Lijding, M., Meratnia, N., Wegdam, M. (eds.) EuroSSC 2006. LNCS, vol. 4272, Springer, Heidelberg (2006)
2. Dey, A.K.: Understanding and Using Context. Personal Ubiquitous Comput. 5, 4–7 (2001)
3. Johanson, B., Fox, A.: The Event Heap: A Coordination Infrastructure for Interactive Workspaces. In: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications, IEEE Computer Society, Los Alamitos (2002)
4. Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling Context Information in Pervasive Computing Systems. In: Proceedings of the First International Conference on Pervasive Computing, Springer, Heidelberg (2002)

5. Newman, M.W., Sedivy, J.Z., Neuwirth, C.M., Edwards, W.K., Hong, J.I., Izadi, S., Marcelo, K., Smith, T.F.: Challenge: Recombinant Computing and the Speakeasy Approach. In: Proceedings of Mobicom 2002 (2002)
6. Ponnekantia, S.R., Johanson, B., Kiciman, E., Fox, A.: Portability, extensibility and robustness in iROS. In: Proceedings of IEEE International Conference on Pervasive Computing and Communications, Dallas-Fort Wirth (2003)
7. Dey, A.K.: Providing Architectural Support for Building Context-Aware Applications. PhD Thesis. College of Computing, Georgia Institute of Technology (2000)
8. Román, M., Hess, C.K., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K.: Gaia: A Middleware Infrastructure to Enable Active Spaces. IEEE Pervasive Computing, 74–83 (2002)
9. Brumitt, B., Meyers, B., Krumm, J., Kern, A., Shafer, S.A.: EasyLiving: Technologies for Intelligent Environments. In: Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing, Springer, Bristol (2000)
10. Henricksen, K., Indulska, J.: A Software Engineering Framework for Context-Aware Pervasive Computing. In: PerCom 2004. Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications, IEEE Computer Society, Los Alamitos (2004)
11. Sousa, J.P., Garlan, D.: Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. In: Proceedings of the 3rd IEEE/IFIP Conference on Software Architecture, Kluwer, B.V (2002)
12. Bardram, J.E.: The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications. In: Pervasive Computing: Third International Conference, Springer, Berlin (2005)
13. Mobile MUSE, http://www.mobilemuse.ca/
14. The Digital Dragon Boat Race (DDBR). Mobile MUSE (2005), http://www.mobilemuse.ca/projects/digital-dragon-boat-race
15. Jeffrey, P., Blackstock, M., Deutscher, M., Lea, R.: Creating Shared Experiences and Cultural Engagement through Location-Based Play. In: Computer Games and CSCW workshop at ECSCW 2005, Paris, France (2005)
16. The Re:call Project. Mobile MUSE (2005), http://www.mobilemuse.ca/projects/re-call-project
17. Metrocode. Mobile MUSE (2007), http://www.mobilemuse.ca/projects/metrocode
18. Smith, R.: Cell in the city: Is cellular phone use eroding the distinction between public and private space? In: Greenberg, J., Elliott, C. (eds.) Communications in question: Canadian perspectives on controversial issues in communication studies, Thomson-Nelson, Toronto, Canada (2007)
19. Cheverst, K., Davies, N., Friday, A., Mitchell, K.: Experiences of Developing and Deploying a Context-Aware Tourist Guide: The Lancaster GUIDE Project. In: Mobicom 2000, Boston, USA (2000)
20. Griswold, W.G., Shanahan, P., Brown, S.W., Boyer, R., Ratto, M., Shapiro, R.B., Truong, T.M.: ActiveCampus: Experiments in Community-Oriented Ubiquitous Computing, vol. 37. IEEE Computer Society Press, Los Alamitos (2004)
21. Johanson, B., Fox, A., Winograd, T.: The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. IEEE Pervasive Computing 1, 67–74 (2002)
22. Bardram, J.E., Hansen, T.R., Mogensen, M., Soegaard, M.: Experiences from Real-World Deployment of Context-Aware Technologies in a Hospital Environment. In: Dourish, P., Friday, A. (eds.) UbiComp 2006. LNCS, vol. 4206, pp. 369–386. Springer, Heidelberg (2006)

23. Ballagas, R., Szybalski, A., Fox, A.: Patch Panel: Enabling Control-Flow Interoperability in Ubicomp Environments. In: PerCom 2004 Second IEEE International Conference on Pervasive Computing and Communications, Orlando, Florida, USA (2004)
24. Greenhalgh, C., Izadi, S., Mathrick, J., Humble, J., Taylor, I.: ECT: a toolkit to support rapid construction of ubicomp environments. In: Davies, N., Mynatt, E.D., Siio, I. (eds.) UbiComp 2004. LNCS, vol. 3205, Springer, Heidelberg (2004)
25. Friday, A., Davies, N., Wallbank, N., Catterall, E., Pink, S.: Supporting service discovery, querying and interaction in ubiquitous computing environments. Wirel. Netw. 10, 631–641 (2004)
26. Grace, P., Blair, G.S., Samuel, S.: A reflective framework for discovery and interaction in heterogeneous mobile environments. SIGMOBILE Mob. Comput. Commun. Rev. 9, 2–14 (2005)
27. Jeffrey, P., Blackstock, M., Finke, M., Tang, T., Lea, R., Deutscher, M., Miyaoku, K.: Chasing the Fugitive on Campus: Designing a Location-based Game for Collaborative Play. Loading..Journal 1(1). Special Issue from Canadian Games Studies Association (CGSA) Workshop, vol. 1 (2006)
28. Nova, N., Girardin, F., Dillenbourg, P.: 'Location is not enough!': an Empirical Study of Location-Awareness in Mobile Collaboration. In: IEEE International Workshop on Wireless and Mobile Technologies in Education, Tokushima, Japan (2005)
29. Place Lab: A Privacy-observant location system. Intel Research Seattle, http://www.placelab.org/
30. JBoss Home Page (2006), http://www.jboss.com/
31. Java 2 Platform, Enterprise Edition (J2EE) Overview. Sun Microsystems, http://java.sun.com/j2ee/overview.html
32. Java Platform Micro Edition at a Glance, http://java.sun.com/javame/index.jsp
33. Salber, D., Dey, A.K., Abowd, G.D.: The context toolkit: aiding the development of context-enabled applications. In: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM Press, Pittsburgh, Pennsylvania (1999)
34. Dey, A.K., Abowd, G.D.: Toward a Better Understanding of Context and Context-Awareness. Georgia Institute of Technology, College of Computing (1999)