The Location Stack: A Layered Model for Location in Ubiquitous Computing

Jeffrey Hightower
University of Washington
Computer Science & Engineering
Campus Box 352350
Seattle, WA 98195
jeffro@cs.washington.edu

Barry Brumitt
Microsoft Research
One Microsoft Way
Redmond, WA 98052
barry@microsoft.com

Gaetano Borriello
University of Washington/
Intel Research
1100 NE 45th Street
Seattle, WA 98105
gaetano.borriello@intel.com

Abstract

Based on five design principles extracted from a survey of location systems, we present the Location Stack, a layered software engineering model for location in ubiquitous computing. Our model is similar in spirit to the seven-layer Open System Interconnect (OSI) model for computer networks. We map two existing ubiquitous computing systems to the model to illustrate the leverage the Location Stack provides. By encouraging system designers to think of their applications in this way, we hope to drive location-based computing toward a common vocabulary and standard infrastructure, permitting members of the ubiquitous computing community to easily evaluate and build on each other's work.

1 Introduction

Location is often essential information for ubiquitous computing systems: We want our home to learn and respond to inhabitants' movements. We want to capture activity in a biological laboratory so that we can record and reproduce experiments. We need directions from one place to another. We want to interact naturally with I/O devices encountered in our environment. Yet despite the critical need for location information, each ubiquitous computing system typically treats such data in its own idiosyncratic manner rather than relying on any standard infrastructure components.

This is not surprising. The substantial amount of engineering and capital investment required to build even a minimal location system dictated that it only be done in support of a limited set of carefully chosen applications. For example, the ParcTab system [19] required the design, deployment, and maintenance of infrared access points throughout the offices and corridors of PARC to support a limited set of applications interested in room-level proximity information. The cost of providing more general capabilities was prohibitive given the expense of the location system itself.

As a consequence, many existing location-aware systems either build the entire system (including sensing, representation, and application logic) as a monolithic structure [12, 14] or they simply surrender to simulating the needed location data in order to remain focused on research questions at the application level [5]. Monolithic systems are easier to build than componentized ones. Tying the sensors to a particular application avoids any mismatch (such as spatial accuracy or update rate) between the data provided by the location sensors and that needed by the application, but at the cost of losing flexibility and scalability. Simulated systems are flexible but often fail to account for the critical complexities inherent to uncertain perception data. Both approaches are difficult to generalize and re-target to new applications.

Fortunately, as a survey of the techniques and issues in location systems for ubiquitous computing [9] shows, location technology trends finally allow the creation of a standard software architecture. Section 2 discusses these trends. In Section 3, we present 5 design principles and the *Location Stack*, a general layered model for location-aware ubiquitous computing systems. Section 4 illustrates the value of the Location Stack through its use in EasyLiving and Labscape, two mature ubiquitous computing systems. Finally, Section 5 outlines opportunities for future work and Section 6 concludes.

2 Location Sensing Trends

Three trends in location technology make now the right time to adopt a standard software architecture for location in

ubiquitous computing. First, hardware and location sensors are becoming more widely available and no longer must be homegrown. Systems like Microsoft Research's RADAR [3], Bluesoft [13], and the United States' E-911 initiatives [6] take location measurements using existing wireless networking infrastructure. Many commercial companies now sell "Active Badge"-like indoor infrared proximity systems. A full-featured GPS chip-set can cost under $100 US and occupies less than $4cm^3$.

Second, though many systems still only support a single technology and sensing modality, reusable software abstractions allowing multiple applications to leverage a single technology are appearing in the research literature. For example, the ActiveBat system at AT&T Cambridge [1] uses a dense grid of ultrasound sensors installed in a dropped interior ceiling to provide precise localization of badges that contain small ultrasonic emitters. These measurements are placed into a database serving as a generic abstraction layer between the sensing and the applications. The robust world model and programming interface allow multiple applications such as location browsing, follow-me resources, and virtual buttons to simultaneously share Bat location information.

Finally, composite location systems and sensor fusion techniques are starting to enable multiplicity in the sensor technologies [8, 16]. Usually no single location technology possesses adequate capabilities (e.g. precision, spatial update rate, ability to locate certain types of objects) to satisfy the broad spectrum of ubiquitous computing applications, particularly when location is to be aggregated with other contextual information through a system such as the Context Toolkit [7]. A composite location system, therefore, uses independent and redundant sensors provide aggregate capabilities and flexibility beyond what any lone technology affords.

From these trends we can conclude that the location problem will be solved either *A)* in rigid, vendor-integrated systems for specific applications or *B)* with robust software abstractions connecting multiple sensing technologies and multiple applications.

3   The Location Stack

Network-based applications are easily implemented in modern computing systems largely because of the Open Systems Interconnection (OSI) layered model to networking. A web browser need not worry about protocols such as Ethernet versus Token Ring or 100-baseT versus 10Base-T versus 802.11. As long as it speaks HTTP, the browser can leverage all lower layers post and retrieve data from remote servers. On the bottom, network hardware needs no knowledge of the applications using it and must only be able to support a specific physical layer. In the middle providing
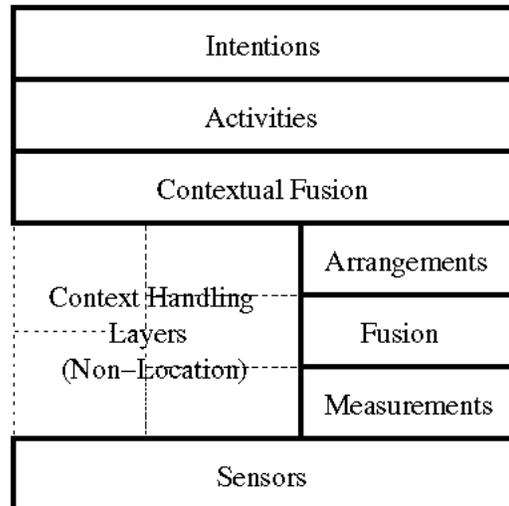

Figure 1. The seven-layer Location Stack, a design abstraction for location-aware ubiquitous computing systems.

a standard transport layer in most systems is IP. Much the same way, the Location Stack is multiple sensing technologies supporting multiple applications with reusable middle layers. Figure 1 shows the Location Stack.

3.1   Design Principles

The Location Stack is based on 5 design principles for location-aware systems extracted from our survey of location systems [9]:

1. **There are fundamental measurement types.** Data reported from location sensors can be classified as distance, angle, proximity, asserted position, or non-geometric features. Each measurement has uncertainty derived from a model of the sensor that created it. Non-geometric features such light level or electromagnetic characteristics of a room may be used to compute location in some systems but often such data is non-location context information handled separately. An example of such a location system is Microsoft Research's RADAR which learns the electromagnetic characteristics correlated with the locations of mobile 802.11 network devices [3].

2. **There are standard ways to combine measurements.** To locate objects, we combine various types of measurements. For example, combining distance measurements with asserted positions (of sensors) creates a multilateration system like AT&T's Bats. Combining proximity measurements with asserted position

yields an Active Badge-style system. Using only distance measurements results in an ad hoc lateration system like SpotON [11]. Any of these systems could be augmented to compute object orientation by adding angle measurements from an on-board digital compass or perhaps simply by inferring that an infrared badge only points forward. In general , every object's location consists of a position, orientation, and time.

3. **There are standard object relationship queries.** Two or more objects can be related by proximity, containment in a region, or geometric formations.

4. **Uncertainty must be preserved.** Applications are concerned with location uncertainty. An application routing telephone calls to a handset near the intended recipient may take a message if uncertainty about the user's location is too high. Performing contextual fusion and activity inference often depend on machine learning algorithms that need to know the fidelity and uncertainty of the underlying location measurements. A study of real-time sensor error [10] has shown that measurement uncertainty in the sensors should be preserved in order to provide the correct uncertainty information at higher abstraction levels.

5. **Applications are usually concerned with activities.** The reason for capturing location and other context data is typically not for direct use in applications but to enable reasoning at the level of user activities. For example, applications want to react when dinner is in progress, a presentation is going on in Conference Room A, or Alice is dispensing a 50% solution of ethylene-glycol into beaker #45039.

3.2 The Layers

We shall now describe each layer of the Location Stack. Since a reference implementation is currently in progress, it is too early to fully specify the interfaces between layers (this is the area on which our research is currently focused). Nonetheless, we believe the Location Stack captures the robust abstractions needed in any location-aware ubiquitous computing system. By encouraging system designers to think of their applications in this space, our hope is to drive location-based computing toward a standard infrastructure that can be easily leveraged by a wide range of sensors and applications. In this section, we adopt the convention of specifying what the layer **contains** (data processing, algorithms, uncertainty representation) and what it **exports** (the data and interface presented to the layer above).

3.2.1 Sensors

- **Contains:** Sensor hardware and software drivers for detecting a variety of physical and logical phenomena such as GPS psuedorange measurements, blob pixels from a camera, time of flight of an ultrasonic emission, a computer login event, or a proximity beacon.

- **Exports:** Raw data values in a variety of formats.

3.2.2 Measurements

- **Contains:** Algorithms to transcribe raw sensor data into the canonical measurement types along with an uncertainty representation based on a model of the sensor that created it. Measurements may be of anonymous objects such as those reported by motion sensors or pressure sensing floor mats.

- **Exports:** A stream of distance, angle, proximity, asserted position, or non-geometric feature measurements.

3.2.3 Fusion

- **Contains:** A general method of continually merging streams of measurements into a time-stamped probabilistic representation of the positions and orientations of objects. Through measurement fusion, differing capabilities, redundancies, and contradictions are exploited to reduce uncertainty. If necessary, the Fusion layer is also responsible for assigning unique identification to objects. No particular coordinate system is implied so stack implementations may choose to use a standard map datum (e.g. the NAD27 or WGS84 commonly used in topographic maps and GPS), implement a custom coordinate system, or implement converters allowing free representation in multiple coordinate frames.

- **Exports:** An query or event interface providing, at minimum, the immediate locations individual objects and the uncertainty of those locations. More complex information may also be available including derivatives (speed, acceleration), positional histories, and object names or unique IDs.

3.2.4 Arrangements

- **Contains:** An engine for probabilistically reasoning about the relationships (e.g. proximity, containment, geometric formations) between two or more objects. This engine is also able to convert location information between absolute and relative coordinates using any of the coordinate systems implemented in the Fusion layer.

- **Exports:** A query or event interface to the relationships between two or more objects which are individually locatable using the Fusion layer interface.

3.2.5   Contextual Fusion

- **Contains:** A system for merging location data with other non-location contextual information such as personal data (calendars, email, contact lists, To-Do lists), color, temperature, light level, galvanic skin response, and so forth.

- **Exports:** An interface allowing applications to recognize interesting states, sequences, or situations presumably to take predictive or responsive action on behalf of users.

3.2.6   Activities

- **Contains:** A system, such as a machine learning system, for categorizing all available context information including location into activities. Activities are semantic states defined by a given ubiquitous computing application. Activities are different from context in that activities are an application's interpretation of the state of the world given both location information and other state information. For example, a home energy management system may wish to conclude that dinner is about to occur or that the residents are all asleep in order to take specific action.

- **Exports:** An application specific interface such as a system of rule-based triggers driving particular application scenarios.

3.2.7   Intentions

- **Contains:** The cognitive desires of users as they relate to what a ubiquitous computing system should do or what task is in progress.

- **Exports:** N/A.

4   Applying the Location Stack

In this section we provide two examples of how the stack model can be applied to real ubiquitous computing systems and discuss what leverage is gained by doing so.

4.1   Labscape

In order to meet its goal of discovering "languages and tools for biology that scientists will find indispensable for their own work, and that directly support communication

Figure 2. An experimental biology laboratory in which the Labscape project locates people, equipment, and a variety of experimental materials in order to automatically gather biological research data and anticipate biologist's experimental needs.

and collaboration", the Labscape project [2] must exploit several types of location information within an environment such as the one shown in Figure 2. The ubiquitous laboratory assistant must be able to locate people, equipment, and a variety of experimental materials in order to automatically gather biological research data and anticipate biologist's experimental needs.

1. **Sensors.** Labscape uses Meepers [15], short-range infrared proximity badges worn by people in the lab, not dissimilar from other "Active Badge"-like infrastructures. Mobile Meepers worn by the biologists transmit unique IDs via infrared to receivers placed throughout the environment. Labscape also employs barcode scanners to locate and identify laboratory equipment that is explicitly scanned by a biologist.

2. **Measurements.** The Labscape sensors generate two types of proximity and asserted position measurements. The Meepers report transmitter-receiver proximity and the barcode scanners report the proximity of particular tags. Asserted position measurements are generated to configure the location of the Meeper receivers and bar code scanners. Additional sensors will eventually generate measurements about the locations of samples, sample trays, pipetters, and other lab equipment.

3. **Fusion.** The Fusion layer of Labscape continually combines streams of proximity and asserted position measurements into a probabilistic belief about the po-

sition and orientation of each biologist. Labscape initially implemented its location service by placing a Meeper receiver at each bench to identify biologists as they approached. However this proved unworkably fragile and rigid: Transmit power of the mobile badges had to be precisely tuned such that only one receiver saw a badge at a time and receivers had to be carefully positioned in the environment for the same reason. The Location Stack, however, enables far more flexibility and scalability. Meeper receivers may be installed wherever convenient (including multiple receivers at a single bench) and transmitters can use any output power they can quantify. The key is in the separation of concerns. The initial implementation tied the Measurements, Fusion, and Arrangements layers together through a rigid dependency on properties of a single technology – an approach seen quite often in the current literature. If we instead separate sensor modeling, location, and arrangement concerns into three separate abstractions we achieve great flexibility and scalability. In the case of Labscape, introducing a new radio proximity technology and augmenting the Meepers to provide an RF time-of-flight measurement is simple.

4. **Arrangements.** Currently possible are proximity and containment queries to place biologists at certain work areas and near other biologists. Eventually, relative locations various glassware and lab tools will also be recognizable.

5. **Contextual Fusion.** Biologists bring to the lab an experimental plan (a directed, acyclic graph) of the steps they will perform. The Contextual Fusion layer uses this graph along with context (non-location) events such as equipment activation and liquid dispensing events to filter the observed location arrangements. The system can then recognize or predict specific actions like pouring (glassware inverted above other glassware), pipetting (pipetter activated above glassware), inserting, and photographing as well as system tasks such as projecting possible walking paths of biologists.

6. **Activities.** The system observes sequences of contextual actions in order to recognize tasks from the experimental plan (e.g. combine, incubate, separate, agitate) in order to suggest them as completed in the workflow. High-level activities external to the plan such as "Dr. Jones is back at her office PC planning which experiments to perform next" are recognizable as well.

7. **Intentions.** Biologists' intentions are to complete specific experiments within proper times and parameters. The Labscape experimental plan graph is an explicit

Figure 3. The EasyLiving system actuates behaviors in the environment dependent on the activities of occupants.

representation of this intention to complete an experiment.

4.2 EasyLiving

The EasyLiving system [4] provides several behaviors which are dependent on the activities of occupants in the environment shown in Figure 3. In particular, if a user has a desktop (interactive computing session) open using one set of devices and he moves to another suitably equipped location, the desktop moves to those devices. Additionally, the lights come on and off based on the user's location, and an appropriate set of audio speakers can be selected using knowledge of the location of the speakers and user.

1. **Sensors.** EasyLiving has 4 different sensors to measure location and identity: cameras, pressure mats, a thumbprint reader, and a keyboard login system.

2. **Measurements.** 3-D stereo cameras made by Point Grey Research measure the location of foreground people-shaped blobs and reports position measurements as a 3 coordinate tuple. The pressure mats periodically report their binary state, i.e. whether a person is sitting or standing on them. The thumbprint reader and the login system report once whenever someone uses them to identify themselves. The only uncertainty modeled in this system is on the asserted position measurements from the camera – they are assumed to have a Gaussian distribution with $\sigma = 15cm$. This layer allows new sensors with similar measurement semantics to be included in the system at very low cost. For example, the thumbprint scanner could be replaced with a smart card reader and nothing above this layer would

need to be changed because both sensors provide proximity measurements with nearly-identical semantics.

3. **Fusion.** The person-tracking module implements the Fusion layer in EasyLiving. It combines past person-track history, knowledge about where people are likely to appear in the room, pressure-mat measurements, and the most recent camera measurements to produce a database of continually updated estimates of where particular people are located. Since the camera sensors cannot determine identity, but rather can only keep track of the identity of a blob once it is assigned, this layer combines measurements from the login sensors with the person tracking information. Knowing where the keyboard and thumbprint reader are located, when a person-identification event occurs, the person track can be assigned a unique identity (The person tracking system need never know the actual identity of the person it is tracking – all it knows is an ID number which it assigns when the person is arrived). The output of the person-tracking module is a list of locations of people in the 2d plane.

4. **Arrangements.** The EasyLiving system uses a SQL database to store its model of the world. The geometric model consists of entities described by a relative position and polygonal extent and the Fusion layer provides updates to this graph of measurements. Two types of arrangement queries are supported: The first returns the list of entities which intersect a given extent and the second returns the location of one entity relative to another. This layer keeps only a instantaneous representation of arrangements, with no history. The Arrangements layer is directly queried by some applications. For example, the media player application can query the database for a list of all speakers in the room, and provide that list to the user. By providing this layer, applications can ignore the particulars of the sensor measurements and sensor fusion process when querying for world state.

5. **Contextual Fusion.** Entities in the geometric model database can also have other information stored under the same ID. All non-geometric state in the EasyLiving system can be considered "context." By associating under the same ID as the geometric entity, applications can interrogate the database for location and state information simultaneously. This database provides the naming authority for all entities of which the system is aware.

6. **Activities.** In order to actuate events, EasyLiving has a Behavior Engine which polls the world model database and issues commands to software agents representing the devices in the room. For example, when a person enters the extent representing the entire space of the room, if the lights aren't already on, they are turned on. Another set of rules is responsible for implementing a follow-me desktop display [17] behavior. The engine examines all appropriate context variables, including both location and user's desktop state in order to decide what actions to take. This is an example of using non-location context.

7. **Intentions.** This is not part of EasyLiving. The geometric and world model databases provide a great deal of information about the presence and actions of people and devices in the world, but as yet, no explicit interpretation of this information is performed.

5 Future Work

5.1 Uncertainty representation

While it is clear that representing the precise nature of a sensor's measurement uncertainty is critical, a general mechanism for this remains elusive. Traditional Gaussian representations [18] suffer from problems with nonlinear transformation between coordinate frames and the scalability of particle filters to large domains remains a challenge, although scalable state estimation techniques used in mobile robotics [8] are an excellent place to start and are the approach taken by our reference implementation.

5.2 Concrete APIs

The Contextual Fusion and Arrangements layers will be the standard interfaces through which most applications will access location information. Determining appropriate APIs (which undoubtedly depend on the uncertainty representations chosen) is an important area of future research. We believe these APIs will emerge as more applications are developed on top of standard infrastructure, just as HTTP emerged to support web applications on top of IP.

5.3 Clarification of the Activities and Intentions layers

Few location-aware ubiquitous computing systems have been built which take sensor information all the way up to activity inference. While the lower layers' functions are clear, the ideal breakdown of the upper layers is itself somewhat uncertain. Further work and collaboration with the machine learning community is needed to determine if a general approach to these layers is feasible.

6 Conclusion

Through our work on two ubiquitous computing applications, we have learned that building a location system dependent on properties of a single technology, although seen often in the literature, is inappropriate. Based on five design principles, we propose the Location Stack to structure the components of what location information provides for ubiquitous computing applications into a layered system architecture with robust separation of concerns. We illustrate the leverage provided by the stack through its application to the Labscape and EasyLiving ubiquitous computing systems. The Location Stack enables the constant evolution of our systems as we deploy new technologies and allows us to partition the work and research problems appropriately. Future research will analyze various implementations of all layers in the context of additional applications. We propose our seven-layer location stack as a model to be adopted by ubiquitous system designers to foster a common vocabulary and encourage interoperability.

7 Acknowledgments

The authors acknowledge discussions with participants of the Ubicomp 2001 Workshop on Location Modeling and location technology discussions held at Intel Research, Seattle. These interactions have been instrumental in developing and refining the Location Stack.

References

[1] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a sentient computing system. *Computer*, 34(8):50–56, August 2001.

[2] L. F. Arnstein, S. Sigurdsson, and R. Franza. Ubiquitous computing in the biology laboratory. *Journal of the Association for Laboratory Automation (JALA)*, 6(1), March 2001.

[3] P. Bahl and V. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of IEEE INFOCOM*, volume 2, pages 775–784, Tel-Aviv, Isreal, March 2000.

[4] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. A. Shafer. Easyliving: Technologies for intelligent environments. In *2nd Intl. Symposium on Handheld and Ubiquitous Computing*, pages 12–27, September 2000.

[5] M. Bylund and F. Espinoza. Using quake III arena to simulate sensors and actuators when evaluating and testing mobile services. In *CHI 2001 Extended Abstracts*, pages 241–242. ACM, March-April 2001. Short Talk.

[6] F. C. Commission. Fcc wireless 911 requirements fact sheet, January 2001. http://www.fcc.gov/e911/.

[7] A. K. Dey. *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, College of Computing, Georgia Institute of Technology, December 2000.

[8] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):325–244, June 2000.

[9] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, August 2001.

[10] J. Hightower and G. Borriello. Real-time error in location modeling for ubiquitous computing. In M. Beigel, P. Gray, and D. Salber, editors, *Location Modeling for Ubiquitous Computing - Ubicomp 2001 Workshop Proceedings*, pages 21–27, Atlanta, GA, September 2001.

[11] J. Hightower, R. Want, and G. Borriello. SpotON: An indoor 3d location sensing technology based on RF signal strength. UW-CSE 00-02-02, University of Washington, Department of Computer Science and Engineering, Seattle, WA, February 2000.

[12] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen. Smart-its friends: A technique for users to easy establish connections between smart artifacts. In *Proceedings of Ubicomp 2001*, pages 116–122, September 2001.

[13] B. Incorporated. Website, 2001. http://www.bluesoft-inc.com/.

[14] N. Marmasse and C. Schmandt. Location modeling. In M. Beigel, P. Gray, and D. Salber, editors, *Location Modeling for Ubiquitous Computing - Ubicomp 2001 Workshop Proceedings*, pages 121–126, Atlanta, GA, September 2001.

[15] C. Mitchell and L. F. Arnstein. *Location Tracking in a Laboratory Environment*. University of Washington CSE, 2001. labscape.cs.washington.edu/meeper/meeper.htm.

[16] T. O'Connell, P. Jensen, A. Dey, and G. Abowd. Location in the aware home. In M. Beigel, P. Gray, and D. Salber, editors, *Location Modeling for Ubiquitous Computing - Ubicomp 2001 Workshop Proceedings*, pages 41–44, Atlanta, GA, September 2001.

[17] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, Jan/Feb 1998.

[18] R. Smith and P. Cheeseman. On the estimation and representation of spatial uncertainty. *International Journal of Robotics Research*, 5(4), Winter 1987.

[19] R. Want, B. Schilit, N. Adams, R. Gold, K. Petersen, D. Goldberg, J. Ellis, and M. Weiser. The parctab ubiquitous computing experiment. In T. Imielinski, editor, *Mobile Computing*, chapter 2, pages 45–101. Kluwer Publishing, February 1997. ISBN 0-7923-9697-9.