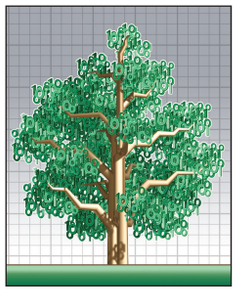
Editor: Munindar P. Singh • *m.singh@ieee.org*



Understanding Location-Based User Experience

Pradeep K. Murukannaiah and Munindar P. Singh • *North Carolina State University*

As location-based applications increase in scope and variety, engineering them to deliver a high-quality user experience becomes increasingly important. The authors describe how user experience criteria map to location-based applications, the special demands these criteria place on modeling, and how to realize such applications to obtain high-quality user experience.

Location is a crucial component of user context. Location-awareness can help enrich virtually any truly mobile application — one whose functionality depends on or reflects users' mobility across locations or presence in a particular location.

Some applications, such as step-by-step navigation, rely on location for their core functionality. Several social applications expose users' locations to others as just one feature; examples include finding a friend nearby or commenting on one's current location. Additionally, classical applications can benefit from incorporating location, including search (for a restaurant near you), social media (tweet from where you are), and weather lookup (provide the outlook for where you're headed, as with Google Now).

User experience (UX) is a well-established research and practice area addressing human-computer interfaces. Specifically, mobile UX addresses challenges such as smaller screen sizes and difficult text entry on mobile devices. To complement such work, we propose location-based UX (LbUX) as a distinct subtheme.

Location-Based UX

Adapting Mike Kuniavsky's definition,[1] LbUX is the totality of a user's perceptions about location-based applications. We understand these perceptions via the following questions, specialized to location:

- *Effectiveness*. Does the application fulfill users' needs by providing location-sensitive functionality?
- *Efficiency*. How much work must users perform to benefit from location? To what extent can the application operate automatically on users' behalf?
- *Intelligibility*. Following Victoria Belotti and Keith Edwards,[2] do users understand what location information the application possesses about them, how it employs such information, and why it needs it?
- *Privacy*. To what extent can users control the sharing of personal information pertaining to location?
- *Satisfaction*. Do users find the overall experience satisfactory?

How we answer these questions and trade-offs between the related criteria determines the LbUX we would deliver.

Designing for LbUX

We posit that delivering LbUX isn't a unitary challenge, but a composite of challenges in the following solution domains:

- *Information architecture*. How should an application represent location information? To deliver an effective LbUX, the representation should be close to a user's cognitive model of location.
- *Service design*. How should an application acquire location information? To be effective, an application must match a user's cognitive model; to be efficient, it must learn about that model unintrusively.

 Published by the IEEE Computer Society 1089-7801/14/$31.00 © 2014 IEEE

- *Interaction design*. Users, applications, and sources of information about users' locations must interact with each other. For transparency and control, users should be able to observe and configure such interactions.
- *Interface design*. The interface should support the first three solutions by conveying how the application understands a user's locations, and give the user an option to correct this understanding.

Aligning these solution domains is crucial. The location information architecture (LIA) should match users' conceptions; the interface should present location accordingly. The service design should compute this LIA, and interactions should enable the right queries, responses, and configuration commands to propagate across the concerned modules.

Information Architecture

Information architecture refers to the structure of the information underlying an application, which significantly affects the experience it delivers. An ideal LIA should be able to represent all locations of interest to a user in a way that maximizes the quality of the experience.

Several existing location-aware applications conceptualize locations as *positions* — that is, as spatial coordinates. Positions are required for applications such as navigation, but are often too low-level to capture users' conceptions of location.[3] To express preferences and policies, users often categorize locations based on relevant activities and social interactions. Consider a bank versus a seminar room: users might wish to allow interruptions from their spouse while at the bank but not while in the seminar room. Moreover, users might want their phones to ring loudly in noisy locations. This suggests that an LIA embodies choices along some key dimensions:

- *Abstraction*. What is the organizing principle for categorizing or grouping locations? Examples include space (spatial proximity, contiguity, or containment), activity (what the user is doing there: reading whether at work or on a train), and society (who the user is interacting with: family or a supervisor).
- *Granularity*. How fine or coarse-grained are the locations?
- *Perspective*. Is the location treated as objective (Denali), subjective (where I saw a rainbow), or intersubjective (where we met)?
- *Attributes*. Does the location incorporate elements of user state (including mobility), device state (battery life), and environmental state (brightness, temperature, ambient noise, and crowdedness)? Attributes of the environmental state act as bases for localization, which involves classifying a location within a broad objective category such as "restaurant" or "museum."[4]

An ideal LIA should be able to represent all locations of interest to a user in a way that maximizes the quality of the experience.

Viewed in these dimensions, position involves spatial proximity at the finest granularity from an objective perspective, and without regard to any additional attributes. Juan Ye and colleagues' spatial model incorporates spatial concepts and thus achieves a coarser granularity than position, but disregards perspective and abstractions that capture nonspatial invariants.[5] *Place* involves a combination of activity and society at a coarse grain from a subjective perspective, and incorporates environmental attributes.[3]

An application must maintain a model of the user's location according to the appropriate LIA its designer has specified. Because users generally employ multiple applications, delivering a compelling and intelligible experience presupposes that these applications adopt consistent LIAs. Consistent doesn't mean identical because the applications would serve different purposes. For example, a user can select a destination socially on the basis of where his or her colleagues are meeting, use a position-based navigation application to arrive there, and use an activity-based notification manager to suppress notifications while the user is addressing the group.

Given multiple applications, LIA alignment is thus crucial for LbUX. Today, such alignment primarily occurs at the position level. Achieving it at a higher level presumes a common ontology from which LIAs are specified.

Service Design

Information about users' location originates from their devices (such as sensors on phones), external infrastructure (Wi-Fi access points and GPS satellites), historical user behavior (for instance, activities carried out, such as messages sent or applications used), external information services (which band is playing at the arena), and user input (a name for a place).

The challenge in service design is how to computationally realize the selected LIA with its abstraction, granularity, perspective, and essential attributes. We can instantiate an LIA in two main ways.

In a traditional framework, each application would obtain location information directly from selected sources, build a suitable model, and present it

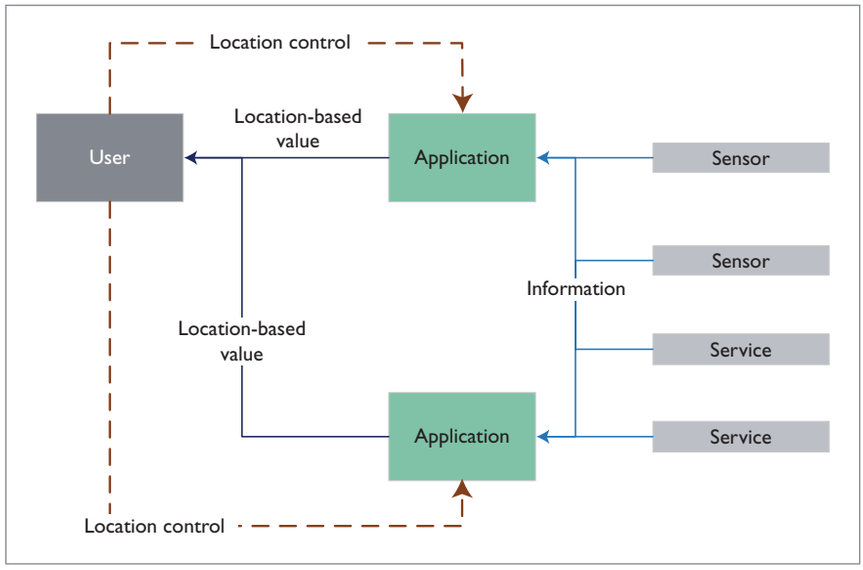


Figure 1. Traditional framework. Each application obtains location information from external sources (sensors and services) and presents its own location-information architecture (LIA) as part of the user experience it offers. The user controls each application's location information separately and must thus learn each application's LIA separately.

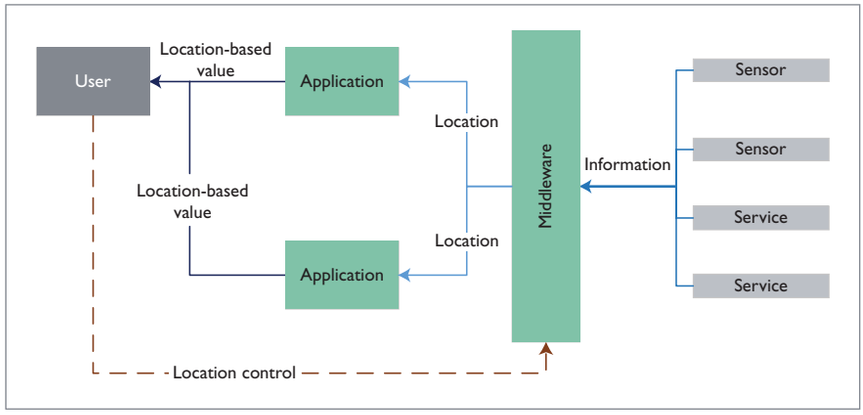


Figure 2. Our recommended framework. A location middleware obtains location information from external sources. The user controls the middleware's location information in a logically central manner. Each application specifies its location-information architecture, and the middleware instantiates the LIA using the available information and user preferences. Each application provides its user experience on top of its LIA. Hence, the user gains from increased uniformity across applications.

to the user packaged with its functionality (see Figure 1). Only an adaptive approach can handle multiple users. Even so, any designer-provided settings would yield an unintelligible experience for a user of multiple applications.

We advocate a middleware-based framework wherein the middleware obtains location information, builds a model (via a user agent), and provides each application with an instantiation of its LIA (see Figure 2). Such a middleware can work if the applications respect a common ontology. The middleware hosts a user agent that builds a personalized model for the user, taking limited input as needed. For multiple applications, the agent provides a personalized experience that remains intelligible and consistent across applications.

Interaction Design

In the middleware-based framework, in addition to the user interacting with the application, the application interacts with the middleware, and the middleware interacts with location-information sources.

The challenges of interaction design are, first, to enable interactions that are intelligible to users and, second, to equip users with means to control the information exchanged in those interactions.

User to middleware. Some LIAs require explicit user input. For example, knowledge of the user's activities is needed for some abstractions but not for others, and for those with the subjective perspective, it can help to have explicit user input. In addition, a user should be aware of and control the location information sources that will be employed.

When the following user interactions are supported by the middleware, they can help improve the user model and thereby potentially benefit all applications:

- *Guidance*. Users can tag locations or otherwise guide the middleware as to the meaning they associate with those locations.
- *Configuration*. Users can specify what sources (sensors and services) to employ, with what frequency, as other constraints on those services' invocation.
- *Control*. Users can examine and control what information is made available to each application.

Middleware to sensors. Once configured, the middleware automatically and continually interacts with the sources. These interactions can be

asynchronous. The middleware would expose a programming interface by which developers could plug new sensors into the middleware.

Application to middleware. The middleware would provide an API with which it would field location information requests from applications and in compliance with the user's chosen configuration. In addition, an application can provide information about a user's preferences and activities, which can help the middleware refine its location model for that user.

Interface Design

Jonathan Raper and his colleagues review location interface designs.[6] A key challenge is how to present location information in a way that accords with users' cognitive model of locations.

We subscribe to the gestalt-theoretic approach for organizing location information that suggests principles of perceptual organization[7]:

- *Proximity*. Objects close in space and time are perceived as belonging together.
- *Similarity*. Objects that share descriptions (in our case, abstractions and attributes) are perceived as belonging together.
- *Past experience*. A location is perceived not as isolated in time, but as a bundle of the user's experiences at that location.

We could potentially adopt additional principles as long as the LIAs and middleware support the resulting interfaces.

Extended Example: Ringer Manager Application

We illustrate these concepts via the Ringer Manager Application (RMA), a mobile application that offers the following location-based functionality:

- Manages the ringer mode (silent, vibrate, or loud) of a user's phone automatically based on his or her current location.
- When the user misses a call, shares the user's location with the caller by sending a message such as "sorry for missing your call; I am in (xyz) location."

The following LbUX components are pertinent:

- *Effectiveness*. Control the ringer mode for a user's phone at all locations of interest.
- *Efficiency*. Support the user quickly and easily in setting and updating desired ringer modes.
- *Privacy preservation*. Equip a user to determine when, how, and to what extent the RMA communicates location information to others.

Service Design

Assuming that RMA employs the place LIA (or similar), is computing places effective and efficient (for users)?

Current application development platforms provide little support: fixed heuristics can be wrong or inconsistent with other applications (ineffective). Repeated reliance on subjective user guidance indicates inefficiency. In contrast, a middleware can capture user guidance independent of applications (efficiency) and produce locations consistent across applications (effectiveness).

Interface and Interaction Design

How can users visualize locations to set or update ringer modes and control how their location information is shared?

The position LIA suggests a map display, which looks simple until the user attempts to figure out correct settings for each position. Grouping by proximity is ineffective because the user's preferences might not group by space. Grouping by ringer mode can simplify setting sharing (efficient), but it might not be what the user wants (ineffective). Similarly, sharing suggests two extreme solutions: ask each time (inefficient) or set to share always or never (ineffective).

Current application development platforms provide little support: fixed heuristics can be wrong or inconsistent with other applications.

We identify key questions an RMA developer must answer in each solution domain and discuss how the traditional and proposed approaches compare.

Information Architecture

Can the user effectively assign a ringer mode for each location in the LIA? Can RMA support multiple ringer modes at a location? Will RMA support differential treatment of the user's spouse and unknown callers, for example? Is the number of locations sufficiently small for the user to maintain settings efficiently?

Answering these questions via place as opposed to position as LIA suggests gains in effectiveness (place is closer than position to users' cognition of location) and efficiency (place is of coarser granularity than position). In contrast, a place-level interface can be more intelligible. Furthermore, because the place LIA composes features of space, activity, and other attributes, users can control interactions with respect to any such features.

Bringing LbUX to Practice

To realize this vision of LbUX we must address two challenges.

The first is this vision's reliance on automation. Each point in the

multiple dimensional LIA space potentially requires a distinct approach. As an illustration, consider place, which incorporates the user's position, activities, and social circles. We developed a machine learning method that learns users' places based on continually sensing their position, activities, and social circles and requesting their advice from time to time to figure out what physical descriptors correspond to what places for the users.[3,8] Our method makes realistic assumptions that user guidance is infrequent, and sensor readings are from multiple sources and intermittent. This method learns spatial and mobility parameters such as radius and users' length of stay.

In an empirical study, six subjects employed our middleware to label places of interest and collect sensor readings from GPS, Wi-Fi, and Bluetooth sensors, and points of interest (POI) information from an external service.[8] Unsurprisingly, we found that no fixed values for spatiotemporal parameters (place radius and users' lengths of stay, specifically) are optimal for all users — thus, adaptivity is essential. Furthermore, we found that our semisupervised approach employing infrequent labels and intermittent sensor readings performs better than traditional techniques and achieves more than 80 percent accuracy in recognizing places.

The second challenge is engineering location-based applications. To this end, we proposed Xipho, a methodology[9] that treats location as a cognitive construct and models an application's location-based requirements in terms of established cognitive constructs[10] such as a user's goals, the application's plans, and dependencies between applications and external services.

We evaluated our methodology and middleware empirically in terms of time for modeling and development, and ease of understanding designs. Our study involving 46 subjects developing location-aware applications supported the hypotheses that, when compared to a traditional methodology, developers employing Xipho both spend less time modeling a location-aware application and produce location-aware application designs that are easier to comprehend.[9] Specifically, the means and standard deviations of time spent (in hours) per feature were 2.3 and 1.6 for the Xipho group and 3.4 and 2.9 for the control group.

Although the middleware helps consolidate disparate knowledge, a potential shortcoming is the risk arising from it possessing the user's entire location information. However, we can mitigate this threat by realizing the location middleware on a server or mobile device that the user owns or controls. For instance, the Platys middleware, which instantiates part of the LbUX approach we describe here, runs on a user's smartphone.

LbUX can suggest Thus the risk remains if the user's phone is compromised, but is no worse than under the traditional framework ways to formulate and address the broader problem of ubiquitous UX design. Additionally, location can be generalized to users' context, which incorporates a more expansive notion of what is (most) relevant to users at a particular moment, given their history of activities and interactions.

Acknowledgment

We thank the US National Science Foundation for support under grant 0910868.

References

1. M. Kuniavsky, *Smart Things: Ubiquitous Computing User Experience Design*, Morgan Kaufmann, 2010.
2. V. Bellotti and K. Edwards, "Intelligibility and Accountability: Human Considerations in Context-Aware Systems," *Human-Computer Interaction*, vol. 16, no. 2, 2001, pp. 193–212.
3. P.K. Murukannaiah and M.P. Singh, "Platys Social: Relating Shared Places and Private Social Circles," *IEEE Internet Computing*, vol. 16, no. 3, 2012, pp. 53–59.
4. M. Azizyan, I. Constandache, and R. Roy Choudhury, "SurroundSense: Mobile Phone Localization via Ambience Fingerprinting," *Proc. 15th Ann. Int'l Conf. Mobile Computing and Networking*, 2009, pp. 261–272.
5. J. Ye et al., "A Unified Semantics Space Model," *Proc. Int'l Conf. Location and Context-Awareness*, 2007, pp. 103–120.
6. J. Raper et al., "A Critical Evaluation of Location-Based Services and Their Potential," *J. Location-Based Services*, vol. 1, no. 1, 2007, pp. 5–45.
7. J. Paay and J. Kjeldskov, "Understanding the User Experience of Location-Based Services: Five Principles of Perceptual Organization Applied," *J. Location-Based Services*, vol. 2, no. 4, 2008, pp. 267–286.
8. C-W. Hang, P.K. Murukannaiah, and M.P. Singh, "Platys: User-Centric Place Recognition," *Proc. AAAI Workshop on Activity Context-Aware Systems*, 2013, pp. 14–20.
9. P.K. Murukannaiah and M.P. Singh, "Xipho: Extending Tropos to Engineer Context-Aware Personal Agents," *Proc. 2014 Int'l Conf. Autonomous Agents and Multi-Agent Systems*, 2014, pp. 309–316.
10. P. Bresciani et al., "Tropos: An Agent-Oriented Software Development Methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, 2004, pp. 203–236.

Pradeep K. Murukannaiah is a PhD student in computer science at North Carolina State University. His research interests include machine learning and software engineering of context-aware systems. Murukannaiah received an MS in computer science from North Carolina State University. Contact him at pmuruka@ncsu.edu.

Munindar P. Singh is a professor in computer science at North Carolina State University. His research interests include multiagent systems and service-oriented computing. Singh has a PhD in computer sciences from the University of Texas at Austin. He's an IEEE fellow, a former editor in chief of IEEE Internet Computing, and the current editor-in-chief of ACM Transactions on Internet Technology. Contact him at m.singh@ieee.org.