Integrating Context Services Through Automatic Path Creation

Jason Hong, James Landay
Group for User Interface Research
University of California at Berkeley
{jasonh, landay}@cs.berkeley.edu

We believe that a middleware service infrastructure can greatly simplify the tasks of creating context-aware applications and maintaining the suite of sensors that supports these applications. By providing a uniform layer of abstraction and a common programming model, such an infrastructure would provide support for context-aware applications in a platform neutral manner. Furthermore, such an infrastructure would be "always on," allowing sensors and devices to be deployed incrementally without having to update every application or having to restart the entire system. The particulars of this infrastructure approach are detailed elsewhere [REF].

One of the key services in this infrastructure would be automatic path creation, a way of flexibly composing existing services to create new functionality. Automatic path creation has been previously used for data format and protocol interoperability (for example, [1] and [2]), but our position is that it can also be used to facilitate the creation of context services. As an example, suppose we have three services, one that converts GPS location to ZIP code information (denoted GPS → ZIP), another that uses ZIP codes to retrieve current weather conditions in that area (ZIP → Weather), and another that uses ZIP codes to find local movie theaters (ZIP → Movie Theaters). Individually, each of these services are not very interesting; however, they can be chained together into paths to form more interesting services, such as using GPS to retrieve weather conditions (chaining GPS → ZIP and ZIP → Weather together).

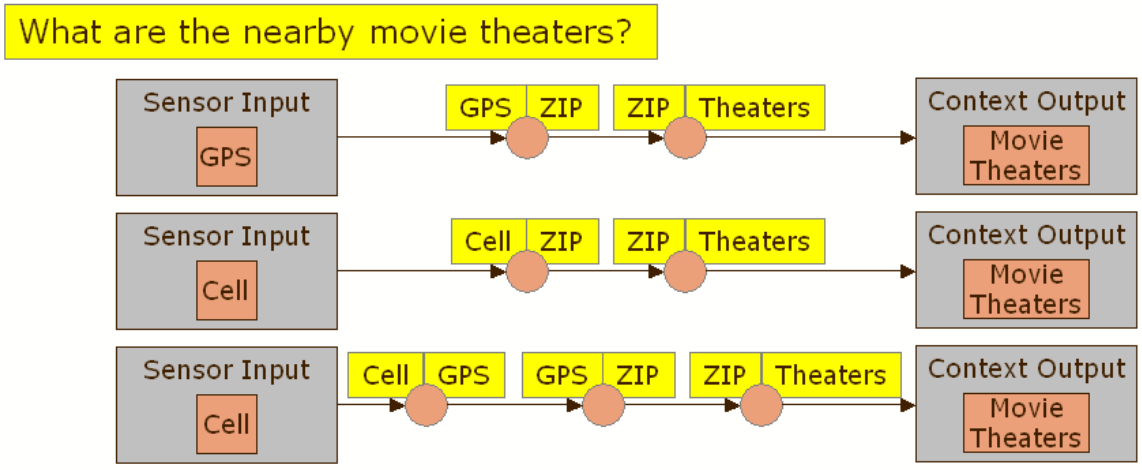

Figure 1 – Automatic path creation can provide a high-level abstraction for accessing context services. This figure shows three different paths for computing the answer to the question "What are the nearby movie theaters?" The first case uses GPS sensors. The second and third cases use cell phone location. With automatic path creation, any of these paths can be created on demand based on whatever resources and services are available.

Clearly these services can be combined manually, but what's interesting about automatic path creation is that these services can be composed automatically based on high-level needs. For example, Figure 1 shows three different ways of answering the high-level query "What are the nearby theaters?" In the first case the path goes from GPS → ZIP and ZIP → theaters. In the second case it goes from cell location → ZIP and ZIP → movie theaters. In the third case it goes from cell location → GPS, GPS → ZIP, and then ZIP → theaters. With automatic path creation, any of these three paths can be created dynamically on demand depending on what kind of location sensors and services are currently available. Automatic path creation relieves application developers from having to know about specific sensors and services. Instead, developers only need to worry about formulating the right context query.

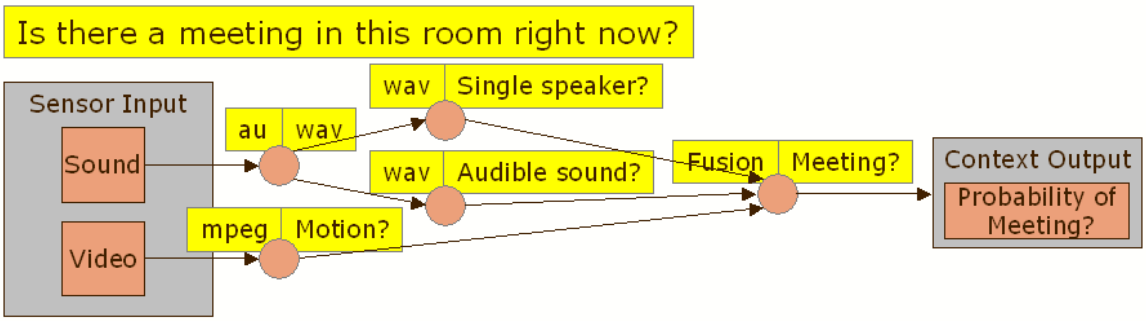

Figure 2 – An example of a complex path, one that tries to calculate the probability that there is a meeting right now.

Figure 2 shows a more complex path, which tries to answer the question "Is there a meeting in this room right now?" Again, this is useful from a software development standpoint. Instead of having to know what sensors are in the room and manually manipulating the sensor and context data, all a developer has to do is pose the right question and let the system figure it out.

We are currently investigating the effectiveness of automatic path creation with respect to context awareness. Currently, we are facing six issues. The first issue is one of engineering, dealing with how services are described and discovered, and where services are run. These are issues that others have looked at (again, [1] and [2]) and we can build off of their work.

The second issue is a matter of practicality: there needs to be a critical mass of useful services that can be composed before this approach is useful. This is just a matter of time and energy. The third issue is that standard data types and data formats need to be developed so that the output of one service can be piped in as the input of another. In many cases we can leverage existing file formats out there (such as gif, jpg, and mp3), but many other data types and data formats will have to be created (such as ZIP code).

The fourth issue is creating good paths. For example, Figure 1 shows three equally valid paths, but it's not clear which one may be the best in terms of quality of information, quality of service, latency, price, and so on. The fifth issue is representing the context queries. A query needs to be rich enough to pose interesting context questions, but also simple enough that it can realistically be understood and processed. The last issue deals with failure modes. For example, what happens if a path can't be generated, or if critical sensors are broken, or if the data is private and shouldn't be accessed? What kind of information should the system return in these cases?

In summary, we are investigating middleware service infrastructures for context-aware computing. In particular, we believe that automatic path creation is promising as a general technique for simplifying the creation of context-aware applications because it provides a high-level interface that abstracts out the need to know what sensors and resources are available.

References

1. Kiciman, E. and A. Fox. Using Dynamic Mediation to Integrate COTS Entities in a Ubiquitous Computing Environment. In Proceedings of *Second International Symposium on Handheld and Ubiquitous Computing 2000*, Sep 2000 2000.
2. Mao, Z.M., *Fault-tolerant, Scalable, Wide-Area Internet Service Composition*, University of California at Berkeley, Berkeley, 2000.