CONTEXT-AWARENESS AND MULTIMODALITY

Daniel Salber

IBM T.J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532, USA
Email : salber@us.ibm.com

Abstract— User interfaces must adapt to the growing dissemination of computing power in our everyday environment. Computing devices and applications are now used beyond the desktop. Mobile, wearable, and pervasive computing allow users to integrate computing in the flow of their activities in the physical world. But most of our systems are still deaf and blind to anything that isn't explicitly input by the user. By taking the environmental interaction context into account, context-awareness promises easier interaction and new possibilities for applications. On the surface, there are many similarities between the needs of multimodal and context-aware applications. What can we learn from multimodality to build context-aware systems ? What are the common research problems ? To investigate these issues, we give a brief overview of context-aware systems, including a simple classification. We describe some abstractions we have found useful to help build context-aware applications. We then turn to the main problems facing the developers of context-aware systems and relate them to issues encountered in multimodal systems development.

1. Introduction

A radical shift is taking place in computing. The typical user is not facing a desktop machine in the relatively predictable office environment anymore. Rather, users have to deal with diverse devices, mobile or fixed, sporting diverse interfaces and used in diverse environments. In appearance, this phenomenon is a step towards the realization of Mark Weiser's ubiquitous computing paradigm, or "third wave of computing," where specialized devices outnumber users [18]. However, many important pieces necessary to achieve the ubiquitous (or pervasive) computing vision are not yet in place. Most notably, interaction paradigms with today's devices fail to account for a major difference with the desktop computing model. Devices are used in changing environments, yet they do not adapt to environmental conditions or the changing context. Although moving away from the desktop brings up a new variety of usage situations in which an application may adapt to or take advantage of the current conditions, computing devices are left unaware of their surrounding environment.

Context-aware computing aims at making computing devices aware of their operating environment. Research prototypes developed in the last decade have shown the value, as well as the difficulties, of this approach. We give several examples of exisiting systems in section 2.2. Among the main difficulties are a poor understanding of the nature of context, and the lack of conceptual models and tools to support the development of context-aware applications. In this paper, we give a short introduction to context and context-aware applications and introduce a model we developed to help deal with context information. This model was implemented in the Context Toolkit which is described in greater detail elsewhere [6, 15]. We then turn to an analysis of the similarities between multimodal and context-aware systems and we argue that context-aware systems actually represent an extension of multimodal systems to a greater number and variety of modalities.

2. Context and context-aware applications

Different disciplines such as linguistics, semantics, and computing have different definitions of context. In context-aware computing, a number of definitions of context have been proposed, usually based on enumerations of context information that can be sensed by applications (e.g., location, identity of the user). We provide a more general definition of context and then identify classes of context-aware applications.

2.1. Definition of context

We define ***context*** as any environmental information that is relevant to the interaction between the user and the application, and that can be sensed by the application. As a preliminary categorization of context information, we have identified the following entities that provide context information, and relevant context attributes that characterize these entities :

- Providers of context information : people, places, physical objects, computing objects.
- Context attributes : location, identity, activity, state.

For example, the location and the identity of the user (people) are used by a museum tour guide application to guide her through an exhibit. An office awareness system keeps track of meetings (activity) occuring in different rooms (places). A conference assistant application displays a thumbnail of the speaker's current Powerpoint slide (state of a computing object) on the user's handheld device [5].

2.2. Classes of context-aware applications

The way context-aware applications make use of context can be categorized into the three following classes : presenting information and services, executing a service, and tagging captured data.

Presenting information and services, refers to applications that either present context information to the user, or use context to propose appropriate selections of actions to the user. There are several examples of this class of applications in the literature and in commercially available systems: showing the user's location on a map and indicating nearby sites of interest [1, 4, 7]; presenting a choice of printers close to the user [16]; or presenting in/out information for a group of users [15].

Automatically executing a service, describes applications that trigger a command, or reconfigure the system on behalf of the user according to context changes. Examples include: the Teleport system in which a user's desktop environment follows her as she moves from workstation to workstation [17]; car navigation systems that recompute driving directions when the user misses a turn [8]; and a recording whiteboard that senses when an informal and unscheduled encounter of individuals occurs and automatically starts recording the ensuing meeting [2].

Attaching context information for later retrieval, refers to applications that tag captured data with relevant context information. For example, a zoology application tags notes taken by the user with the location and the time of the observation [13]. The informal meeting capture system mentioned above provides an interface to access informal meeting notes based on who was there, when the meeting occurred and where the meeting was located. Some of the more complex examples in this category are memory augmentation applications such as Forget-Me-Not [9] and the Remembrance Agent [14].

These categories clarify how applications can use context. To design and build context-aware applications, designers and builders need abstractions to reason about context, unencumbered by the details of actually acquiring and managing context information.

3. Software abstractions for context

Designing and building context-aware applications raises new challenges. The three following issues are most relevant to this discussion:

1) Context is acquired from unconventional sensors. Mobile devices for instance may acquire location information from outdoor GPS receivers or experimental indoor positioning systems. Tracking the location of people or detecting their presence at a given place may require Active Badge devices, floor-embedded presence sensors or video image processing.

2) As a consequence, there are no readily reusable libraries of software components that deal with context. Even for fairly well understood systems like GPS receivers, programmers have to deal with hardware issues (connecting, embedding), as well as configuration (choosing a coordinates format) and software issues (parsing the GPS output and combining location information with other user or context input).

3) Context must be abstracted to make sense for the application. GPS receivers provide geographical coordinates. But tour guide applications would make better use of higher-level information such as street or building names. Similarly, Active Badges provide IDs, which must be abstracted into user names and locations.

High-level components, such as widgets found in graphical user interface (GUI) toolkits provide designers and developers with not only reusable software components, but also tools for reasoning. The precise interaction mechanics of a menu or a button are taken for granted. The designer is only concerned with the adequation of the widget to the user's task and the correct handling by the application of the information acquired from the widget. The context widget abstraction aims at fulfilling a similar role for context information.

3.1. Context widgets

A ***context widget*** is a software component that provides applications with access to context information from their operating environment. In the same way GUI widgets insulate applications from some interaction concerns, context widgets insulate applications from context acquisition concerns.

Context widgets have a state and a behavior. The widget state is a set of attributes that can be queried by applications. For example, a Location widget may have attributes for the current street address closest to the user, elevation, and heading. Applications can also register to be notified of context changes detected by the widget. The widget triggers callbacks to the application when changes in the environment are detected. The Location widget for instance, provides callbacks to notify the application when the current street address or heading changes.

A context widget is typically used to encapsulate one context attribute of one context entity (as described in section 2.1). Examples include : the location of the user, the number of people in a given room, the identity of a visitor stepping into a user's office, etc.

3.2. Context widgets components

Context widgets are actually made up of several components (see Figure 1). At the lowest level, interfacing directly with sensors, are ***generators***. Their role is to encapsulate a given sensor and hide the specifics of the particular sensor from the rest of the widget, to allow easy upgrade or replacement of the sensor by another providing similar functionality.

Interpreters are in charge of abstracting the data provided by sensors. They might combine information from several sensors (e.g., computing an acceleration vector for a mobile user by collating information from several accelerometers), and perform elaborate transformations that require access to external data (e.g., deriving a street name from geographical coordinates).

The ***widget controller*** is the heart of the widget: it coordinates the other components and is the entry point for applications to query the widget state. It also triggers callbacks for applications that registered interest in the context information the widget is handling.

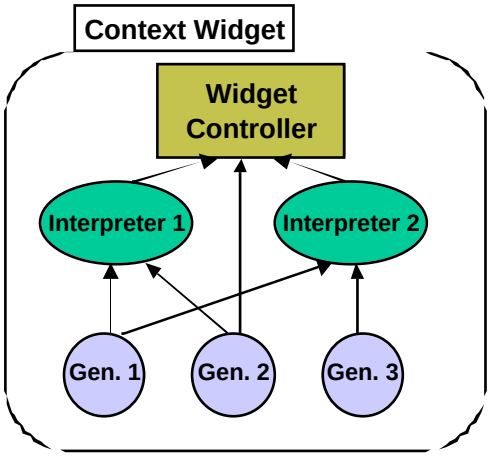

Figure 1 — Anatomy of a context widget. Arrows represent data flow. Generators 1 through 3 interface with sensors and feed sensor data to interpreters and the widget controller. The widget controller maintains state and triggers callbacks.

3.3. Benefits of context widgets

A context widget is a software component that provides applications with access to context information from their operating environment. In the same way GUI widgets insulate applications from some presentation concerns, context widgets insulate applications from context acquisition concerns.

Context widgets provide the following benefits:

- They hide the complexity of the actual sensors used from the application. Whether the location of people is sensed using Active Badges, floor sensors, video image processing or a combination of these doesn't change the application.
- They abstract context information to suit the expected needs of applications. A location widget for example, provides street or room names instead of geographical coordinates and room IDs. Widgets provide abstracted information that we expect applications to need the most frequently.
- They provide reusable and customizable building blocks of context sensing. A widget that tracks the location of a user can be used by a variety of applications, from tour guides to office awareness systems. Furthermore, context widgets can be tailored and combined in ways similar to GUI widgets. For example, a Presence widget senses the presence of people in a room. A WhiteboardActivity widget senses when the room's whiteboard is being used. A Meeting widget may rely on the Presence and WhiteboardActivity widgets and assume a meeting is beginning when two or more people are present and the whiteboard is in use.

From the application's perspective, context widgets encapsulate context information and provide methods to access it in a way very similar to a GUI toolkit. However, due to the unique characteristics of context information, context widgets are different from GUI widgets in several ways.

3.3. Differences between context widgets and GUI widgets

There are several important differences between context widgets and GUI widgets. These differences deal with reliability, distribution, persistence, and with the availability of a complete widget hierarchy.

Context information is usually acquired from sensors, many of which are notoriously unreliable. They may provide noisy data, drift, or simply fail under certain conditions. To account for this, confidence factors must be associated with context information at all stages, and must be propagated up to the application so that it can make informed decisions.

Context may be acquired from multiple distributed sources and used in yet another location. A typical mobile application for instance, will require information about the status of remote or nearby resources, such as the nearest printer that is not busy. Furthermore the location of the user may be known to a central system (e.g., an Active Badges server) that the mobile devices need to query. Thus, a supporting distributed infrastructure is required. In our view, each context widget is an independent process that may be distributed anywhere on the network.

Context may be needed at any time by an application. Thus a context widget is active all the time, and its activation is not, as with GUI widgets, driven by applications. Furthermore, the application may need information acquired at any time in the past. For instance, an application might pull up the notes the user took the last time she was meeting with the same people she is currently meeting with. To this aim, context widgets maintain a history of context information, typically stored in a database.

Finally, the structure and completeness of a context widgets library is still an open research issue. Although our categories of section 2.1 might provide an initial breakdown roughly analog to Foley's categories of interaction widgets, we still need to develop new applications and explore the utility and richness of context information to be able to address this issue.

4. Multimodality and context-awareness

Looking at some definitions of multimodality is an interesting and useful exercise for anyone interested in context-awareness. Indeed, system-centric definitions of multimodal systems [11] can be extended to account for context-aware systems without difficulties. Multiplicity of communication channels between the user and the system and high power of abstraction (for input), are the two key features that characterize multimodal systems. To

ensure that context-aware systems are covered in all their diversity, it might be necessary to stretch the notion of « user » to « user and environment ». But this extension is already implicit in some systems presented as multimodal, such as some biometrics systems. A more salient difference between these systems is actually found when looking at them from a usage perspective : do users act and perform commands, or are they observed by the system, which then adapts its behavior ? In other words, does the user intentionnally direct commands at the system, or does the user act in the real world, and her actions and other changes in the environment are picked up by the system without explicit user action ? Or, to adopt a system-centric perspective, does the system expect well-formed complete commands or does it sense its environment and tries to derive commands (or parameters to commands) from the context information it gathers ?

This discussion suggests that multimodal systems and context-aware systems differ mainly in the way the information is acquired by the system : explicit user input vs. implicit sensing. It is thus valuable to analyze further some issues related to the construction of context-aware systems and relate them to key issues raised in multimodal systems development : data fusion and abstraction. (Let us note in passing that context-aware systems research suffers from the very same problem that multimodal systems once encountered : too much focus on input, and little concern for output. Output in the case of context-awareness relates to the control of actuators, or of the sensors themselves.)

4.1. Data Fusion

Fusion of context information is a key issue for context-aware systems. Fusion comes into play at several levels : to enhance reliability, multiple similar or heterogeneous sensors can be used to acquire the same context information. For example, sensing the presence of a user in a room can require the deployment of several presence sensors, e.g., because the space to cover is greater than the range of a single sensor or to compensate for the innacuracies of the sensors used. A more complex example of fusion would be a context widget that determines if a meeting is taking place in the room : it might combine calendar information from a public schedule, sensors to detect the presence of people, if they're sitting, if they're talking together, etc. In multimodal systems, fusion has been recognized early on as a key mechanism and generic fusion engines have been developed [10, 12]. A preliminary study of the Open Agent Architecture for example, showed that it provides an adequate distributed platform and fusion mechanism for context-aware systems. The applicability of the multimodal fusion mechanisms to context-aware systems still needs to be investigated thoroughly. Useful for assessing fusion needs, the CARE properties framework [3] is a promising tool for the analysis of context-aware systems. Indeed, by simply considering a context information that needs to be acquired, instead of a command, the CARE properties can be easily adapted to context-awareness.

4.2. Abstraction

Our presentation of context-aware systems and the context widget model we have introduced emphasizes the need to abstract low-level information acquired from sensors to suit the needs of applications. Currently, this abstraction step is usually more complex than what is typically needed in a multimodal system. However, some similar questions are raised. For instance, when abstracting, should we keep track of the way (e.g., device) the information was acquired in the first place ? In our model, we decided against it because of the sheer variety of sensors available. Instead, we aim at propagating to the application meta-information that characterizes the piece of context information provided. For now, this meta-information is limited to a confidence factor that is determined in part by the type of sensor used. But additional details that describe more thoroughly the quality of the context information might be useful.

5. Conclusion

Context-aware systems development faces significant challenges as multimodal systems did ten years ago. We have introduced rough classifications that identify categories of context information and classes of context-aware applications. A software model, based on the notion of context widget, provides an easy way for designers and developers to deal with context while being shielded from the intrinsic difficulties of context acquisition. Finally, we have shown that context-aware and multimodal systems share salient characteristics and that we should take a further look at how context-awareness and multimodality might benefit from each other.

Bibliography

[1] Abowd, G.D., Atkeson, C.G., Hong, J., Long, S., Kooper, R., and Pinkerton, M., Cyberguide: A mobile context-aware tour guide, ***ACM Wireless Networks, 5 (3)***, 1997, 421-433.

[2] Brotherton, J., Abowd, G.D., and Truong, K., ***Supporting capture and access interfaces for informal and opportunistic meetings,*** Technical Report GIT-GVU-99-06, Georgia Institute of Technology, GVU Center, 1999.

[3] Coutaz, J.l., Nigay, L., Salber, D., Blandford, A., May, J., and Young, R.M., Four Easy Pieces for Assessing the Usability of Multimodal Interaction: the CARE Properties, in ***Proceedings of INTERACT'95*** Lillehammer, Norway, 1995.

[4] Davies, N., Mitchell, K., Cheverst, K., and Blair, G., Developing a context-sensitive tour guide, in ***Proceedings of 1st Workshop on Human Computer Interaction for Mobile Devices***, 1998.

[5] Dey, A.K., Futakawa, M., Salber, D., and Abowd, G.D., The Conference Assistant: Combining context-awareness with wearable computing, in ***Proceedings of 3rd International Symposium on Wearable Computers*** San Francisco, CA, 1999, 21-28.

[6] Dey, A.K., Salber, D., and Abowd, G.D., A context-based infrastructure for smart environments, in ***Proceedings of 1st International Workshop on Managing Interactions in Smart Environments (MANSE '99)*** Dublin, Ireland, 1999, (to appear).

[7] Fels, S., Sumi, Y., Etani, T., Simonet, N., Kobayshi, K., and Mase, K., Progress of C-MAP: A context-aware mobile assistant, in ***Proceedings of AAAI 1998 Spring Symposium on***

Intelligent Environments Palo Alto, CA, AAAI Press, 1998, 60-67.

[8] Hertz (1999). NeverLost. Available at http://www.hertz.com/serv/us/prod_lost.html.

[9] Lamming, M. and Flynn, M., Forget-me-not: Intimate computing in support of human memory, in ***Proceedings of FRIEND 21: International Symposium on Next Generation Human Interfaces*** Tokyo, 1994, 125-128.

[10]Moran, D.B., Cheyer, A.J., Julia, L.E., Martin, D.L., and Park, S., Multimodal User Interfaces in the Open Agent Architecture, in ***Proceedings of IUI'97*** Orlando, FL, ACM Press, 1997, 61-68.

[11]Nigay, L., ***Conception et modélisation logicielles des systèmes interactifs : application aux interfaces multimodales,*** Ph.D. thesis, University Joseph Fourier, 1994.

[12]Nigay, L. and Coutaz, J.l., A Generic Platform for Addressing the Multimodal Challenge, in ***Proceedings of CHI'95*** Denver, CO, ACM Press, 1995, 98-105.

[13]Pascoe, J., Ryan, N.S., and Morse, D.R., Human-Computer-Giraffe Interaction – HCI in the field, in ***Proceedings of Workshop on Human Computer Interaction with Mobile Devices***, 1998.

[14]Rhodes, B.J., The wearable remembrance agent, in ***Proceedings of 1st International Symposium on Wearable Computers, ISWC '97*** Cambridge MA, IEEE Press, 1997, 123-128.

[15]Salber, D., Dey, A.K., and Abowd, G.D., The Context Toolkit: Aiding the development of context-enabled applications, in ***Proceedings of CHI'99*** Pittsburgh, PA, 1999, 434-441.

[16]Schilit, B., Adams, N., and Want, R., Context-aware computing applications, in ***Proceedings of 1st International Workshop on Mobile Computing Systems and Applications***, 1994, 85-90.

[17]Want, R., Hopper, A., Falcao, V., and Gibbons, J., The active badge location system, ***ACM Transactions on Information Systems, 10 (1)***, 1992, 91-102.

[18]Weiser, M., The computer for the 21st Century, ***Scientific American, 265 (3)***, 1991, 66-75.