Understanding and Using Context

ANIND K. DEY

Future Computing Environments Group

College of Computing & GVU Center

Georgia Institute of Technology

Atlanta, GA, 30332-0280, USA

Tel: +1-404-894-5103

Fax: +1-404-894-2970

E-mail: anind@cc.gatech.edu

Abstract

Context is a poorly used source of information in our computing environments. As a result, we have an impoverished understanding of what context is and how it can be used. In this paper, we provide an operational definition of context and discuss the different ways that context can be used by context-aware applications. We also present the Context Toolkit, an architecture that supports the building of these context-aware applications. We discuss the features and abstractions in the toolkit that make the task of building applications easier. Finally, we introduce a new abstraction, a situation, which we believe will provide additional support to application designers.

1. Introduction

Humans are quite successful at conveying ideas to each other and reacting appropriately. This is due to many factors: the richness of the language they share, the common understanding of how the world works, and an implicit understanding of everyday situations. When humans talk with humans, they are able to use implicit situational information, or *context*, to increase the conversational bandwidth. Unfortunately, this ability to convey ideas does not transfer well to humans interacting with computers. In traditional interactive computing, users have an impoverished mechanism for providing input to computers. Consequently, computers are not currently enabled to take full advantage of the context of the human-computer dialogue. By improving the computer's access to context, we increase the richness of communication in human-computer interaction and make it possible to produce more useful computational services.

In order to use context effectively, we must understand what context is and how it can be used, and we must have architectural support. An understanding of context will enable application designers to choose what context to use in their applications. An understanding of how context can be used will help application designers determine what context-aware behaviors to support in their applications. Finally, architectural support will enable designers to build their applications more easily. This architectural support has two parts: services and abstractions.

In this paper, we will review previous attempts to define and provide a characterization of context and context-aware computing, and then present our own definition and characterization. We then discuss how this increased understanding informs the development of a shared infrastructure, the Context Toolkit[1], for context-sensing and context-aware application development. We discuss both the services offered by the toolkit and the programming abstractions it provides to designers.

---

[1] The Context Toolkit can be downloaded at http://www.cc.gatech.edu/fce/contexttoolkit

2. What is Context

To develop a specific definition that can be used prescriptively in the context-aware computing field, we will look at how researchers have attempted to define context in their own work. While most people tacitly understand what context is, they find it hard to elucidate. Previous definitions of context are done by enumeration of examples or by choosing synonyms for context.

2.1 Previous Definitions of Context

In the work that first introduces the term 'context-aware,' Schilit and Theimer [7] refer to context as location, identities of nearby people and objects, and changes to those objects. These types of definitions that define context by example are difficult to apply. When we want to determine whether a type of information not listed in the definition is context or not, it is not clear how we can use the definition to solve the dilemma.

Other definitions have simply provided synonyms for context; for example, referring to context as the environment or situation [1,4,8]. As with the definitions by example, definitions that simply use synonyms for context are extremely difficult to apply in practice. The definitions by Schilit *et al.* [6] and Pascoe [3] are closest in spirit to the operational definition we desire. Schilit *et al.* claim that the important aspects of context are: where you are, who you are with, and what resources are nearby. Pascoe defines context to be the subset of physical and conceptual states of interest to a particular entity. These definitions are too specific. Context is all about the whole situation relevant to an application and its set of users. We cannot enumerate which aspects of all situations are important, as this will change from situation to situation. For this reason, we could not use these definitions provided.

2.2 Our Definition of Context

> *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*

This definition makes it easier for an application developer to enumerate the context for a given application scenario. If a piece of information can be used to characterize the situation of a participant in an interaction, then that information is context. Take the canonical context-aware application, an indoor mobile tour guide, as an example. The obvious entities in this example are the user, the application and the tour sites. We will look at two pieces of information – weather and the presence of other people – and use the definition to determine whether either one is context. The weather does not affect the application because it is being used indoors. Therefore, it is not context. The presence of other people, however, can be used to characterize the user's situation. If a user is traveling with other people, then the sites they visit may be of particular interest to her. Therefore, the presence of other people is context because it can be used to characterize the user's situation.

3. Defining Context-Aware Computing

Context-aware computing was first discussed by Schilit and Theimer [7] in 1994 to be software that "adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time." Since then, there have been numerous attempts to define context-aware computing, most of which have been too specific [2].

3.1 Our Definition of Context

> *A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.*

We have chosen a general definition of context-aware computing. When we try to apply previous definitions to established context-aware applications, we find that they do not fit.

3.2 Features for Context-Aware Applications

Similar to the problem of defining context-aware, researchers have also tried to specify the important features of a context-aware application [3,6]. Again, these features have tended to be too specific to particular applications.

Our proposed categorization combines the ideas from previous taxonomies and attempts to generalize them to satisfy all existing context-aware applications. There are three categories of features that a context-aware application can support:

- *presentation* of information and services to a user;
- automatic *execution* of a service for a user; and
- *tagging* of context to information to support later retrieval

4. Support for Building Applications

With an understanding of what context is and the different ways in which it can be used, application builders can more easily determine what behaviors or features they want their applications to support and what context is required to achieve these behaviors. However, something is still missing. Application builders may need help moving from the design to an actual implementation. This help can come in two forms. The first is a combination of architectural services or features that designers can use to build their applications from. The second form is abstractions that allow designers to think about their applications from a higher level. We have built an architecture, the Context Toolkit, that contains a combination of features and abstractions to support context-aware application builders. In this section, we will discuss the features and abstractions in the Context Toolkit, and propose a new abstraction.

4.1 Features for Context-Aware Applications

The Context Toolkit makes it easy to add the use of context to existing non-context-aware applications and to evolve existing context-aware applications. In addition, the architecture makes context-aware applications resistant to changes in the context-sensing layer. It encapsulates changes and the impact of changes, so applications do not need to be modified.

Our architecture is built on the concept of enabling applications to obtain the context they require without them having to worry about how the context was sensed. In previous work, we presented the *context widget* [5], an abstraction that implements this concept. A context widget is responsible for acquiring a certain type of context information and it makes that information available to applications in a generic manner, regardless of how it is actually sensed. Applications can *access* context from widgets using traditional poll and subscribe methods, commonly available with graphical user interface (GUI) widgets.

With most GUI applications, widgets are instantiated, controlled and used by only a single application. In contrast, our context-aware applications do not instantiate individual context widgets, but must be able to access existing ones, when they require. To meet this requirement, context widgets *operate independently* from the applications that use them. This eases the programming burden on the application designer by not requiring her to maintain the context widgets, while allowing her to easily communicate with them. Because context widgets run independently of applications, there is a need for them to be persistent, available all the time.

Because an important part of context is historical information, the Context Toolkit provides support for the *storage* of context. Context widgets automatically store all of the context they sense and make this history available to any interested applications. Applications can use historical information to predict the future actions or intentions of users. This prediction or interpretations functionality is encapsulated in the *context interpreter* abstraction. Interpreters accept one or more types of context and produce a single piece of context. An example is converting from a name to an e-mail address. A more complicated example is interpreting context from all the widgets in a conference room to determine that a meeting is occurring.

Traditional user input comes from the keyboard and mouse. These devices are connected directly to the computer they are being used with. When dealing with context, the devices used to sense context most likely are not attached to the same computer running the application. For example, an indoor infrared positioning system may consist of many infrared emitters and detectors in a building. The sensors must be physically distributed and cannot all be directly connected to a single machine. The Context Toolkit makes the *distribution* of the context architecture transparent to context-aware applications, mediating all communications between applications and components.

The final abstraction supported by our architecture is aggregation. *Context aggregators* aggregate or collect context. The notion of an aggregator comes directly from our definition of context. We defined context as information used to characterized the situation of an entity. If we think of a context widget as being responsible for a single piece of information, we need an abstraction to represent an entity. This abstraction, a context aggregator, is responsible for all the context for a single entity. When designers think about context and interactions, it is natural for them to think in terms of entities, and that makes an aggregator the correct abstraction to use for building applications. Aggregators gather the context about an entity (e.g., a person) from the available context widgets, behaving as a proxy to context for applications.

To summarize, the Context Toolkit supports common features required by context-aware applications: capture and access of context, storage, distribution, and independent execution from applications. The toolkit provides three abstractions: widgets, interpreters and aggregators.

4.2 The Situation Abstraction

The support provided by the Context Toolkit has enabled us to build a number of applications that would otherwise have been difficult to build. However, we have recently been experimenting with a new type of abstraction for supporting application builders. This new abstraction, a *situation*, is at a level above widgets, interpreters and aggregators.

The idea of the situation abstraction was also derived from our definition of context. Currently, application designers need to explicitly poll and subscribe to widgets and aggregators for context information and call on interpreters to determine when relevant entities are in a particular state so they can take action. This collection of states can be described as a situation.

The situation abstraction is exactly that: a description of the states of relevant entities. We believe that providing this description requires less effort than determining which individual context components need to be contacted and determining when the collective situation has been realized or satisfied. Instead, the Context Toolkit is responsible for the translation of the description to the "wiring" of the context components and for determining when the individual elements of the situation have been collectively satisfied. Now context-aware application designers can concentrate on the heart of the design process: determining what context-aware features their application should support and when should they be enacted.

We currently have limited support for the situation abstraction. We are struggling with the tradeoff between supporting extremely complex situations and providing a simple method for describing situations. Ideally, we would like to support both simultaneously. By simplifying the process for determining when interesting events occur, the situation abstraction may prove to be useful for end users. One of the holy grails of context-aware computing is to have applications that do the right thing at the right time for users. While designers who have domain-specific expertise can determine part of the solution, they will obviously not think of everything that is needed to support individual users. It is the end user who is in the best position to further specialize context-aware applications to meet their individual needs. The situation abstraction may allow users to perform this specialization.

We would like to carry out user studies to investigate whether the situation abstraction is appropriate for both application designers and end-users and how it compares to the original abstractions of widgets, aggregators, and interpreters.

References

1. Schilit, B., Theimer, M. Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5). 1994. pp 22-32.
2. Brown, P.J. The Stick-e Document: a Framework for Creating Context-Aware Applications. In: Proceedings of Electronic Publishing '96. 1996. pp 259-272
3. Rodden, T., Cheverst, K., Davies, K. Dix, A.. Exploiting Context in HCI Design for Mobile Systems. Workshop on Human Computer Interaction with Mobile Devices (1998)
4. Ward, A., Jones, A., Hopper, A. A New Location Technique for the Active Office. IEEE Personal Communications 4(5). 1997. pp 42-47
5. Schilit, B., Adams, N. Want, R. Context-Aware Computing Applications. 1st International Workshop on Mobile Computing Systems and Applications. 1994. pp 85-90
6. Pascoe, J. Adding Generic Contextual Capabilities to Wearable Computers. In: Proceedings of 2nd International Symposium on Wearable Computers. 1998. pp 92-99

7. Dey, A.K. Abowd, G.D. Towards a Better Understanding of Context and Context-Awareness. CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness (2000)
8. Salber, D., Dey, A.K., Abowd, G.D. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In: Proceedings of CHI'99. 1999. pp 434-441