Exploiting Context in Mobile Applications

Benou Poulcheria
University of Peloponnese, Greece

Vassilakis Costas
University of Peloponnese, Greece

INTRODUCTION

Pervasive computing is nowadays becoming a reality, exploiting the capabilities offered by both computing infrastructure and communication facilities. The pervasive computing environment encompasses a multitude of diverse devices, operating systems, protocols, and standards. It includes mobile devices such as cellular phones, smart phones, PDAs, and handheld computers for information access, smart cards, and smart labels for identification and authentication, smart sensors, and actuators that perceive the surroundings and react accordingly. Voice technologies such as automatic speech recognition (ASR), text to speech (TTS) and VoiceXML enable the construction of convenient user interfaces and Web services are a key mechanism for interoperability. Wireless wide area networking allows long distance communication through cellular radio while wireless local and personal area networking and standards such as the Wi-Fi, Bluetooth and IrDA allow short distance communication through radio waves and infrared beams.

In the mobile and pervasive computing environment, software engineering should not treat diversity and mobility as problems to overcome, but seek methods of which it could take advantage instead. In these environments, the selection of purpose-oriented and timely information, tailored to user preferences and media characteristics will ensure optimised information delivery. To this end, the context—the information that surrounds the human-computer interaction—plays a key role and is rapidly changing in mobile settings, and the understanding of it is indispensable for application designers in order to choose, capture and exploit it. The importance of the context is to use it to make context-aware applications, that is, those applications that are interested in who, where, when and what, in order to determine why the situation occurs and adapt their behavior accordingly.

BACKGROUND

The Concept of Context

An important dimension of mobile computing is "mobility", which is primarily concerned about people moving in space and doing their personal, social and professional activities in a wide temporal space. The informative support of the mobile user can be accomplished through terminals, which are movable and operate regardless of the location and time and offer wireless access to information and services. Although mobility—spatial and temporal—is an important aspect of mobile computing, it constitutes only one dimension of the "context".

The term "context" is defined as "the interrelated conditions in which something exists or occurs" in Merriam-Webster's dictionary. In the domain of context-aware computing, researchers have defined context as "location, identities of nearby people and objects, and changes to these objects" (Schilit & Theimer, 1994), "location, identities of the people around the user, the time of day, season, temperature, etc." (Brown, Bovey, & Chen, 1997) and "knowledge about the user's and the device's state, including surroundings, situation and location" (Schmidt & Laerhoven, 2001).

Dey and Abowd (1999) propose a more generic definition according to which context is any information that can be used to characterize the situation of an entity. An entity is a person or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves. Dey's definition is more comprehensive and generic and makes it easier for an application designer to enumerate the context for a given application and choose the appropriate desirables.

The contextual information can be classified, according to which entity it concerns, into the following categories (Schilit, Adams, & Want, 1994):

- **User Context:** User identity, location, collection of nearby people, social situation, activity, user's profile, and so forth.

- **Computing Context:** Hardware characteristics, software characteristics, network connectivity, communication bandwidth, nearby resources such as printers, displays and other devices and so forth.
- **Physical Context:** Lighting, noise level, temperature, humidity, and so forth.
- **Time Context:** Time of the day, week, month, season of year, time zone, and so forth.

Orthogonally to these classifications, context can be divided into two broad classes: *primary* and *secondary context.* Primary context derives directly from sensors or information sources while secondary context is inferred from the primary context. For example the name of a city is a secondary context because it derives from GPS coordinates through a relation mechanism.

Context can be also distinguished according to a range of temporal characteristics and it can be classified as *static* or *dynamic*. Static context does not change very quickly (or at all; e.g., a person's date of birth), while dynamic context does (e.g., the location of a person who is driving). When applications are not only interested in the current state of the context (*present context*), but past context is of importance too, *context histories* are stored; in some situations there is a necessity for context prediction (*future context*).

The interactions between context sources and context sinks can be characterized as "context push", when the context sources update periodically context information in context sinks and "context pull", when the sinks demand information from the context sources.

The types of context, according to the manner of its acquisition, can be divided in three categories:

- **Sensed context:** This context is acquired from the environment by means of physical or software sensors (identity, temperature, time).
- **Derived context:** This kind of contextual information is computed (for example the name of the city from GPS coordinates).
- **Context explicitly provided:** The context that the user provides explicitly (for example the entries in the user profile).

Defining Context-Aware Applications

Context-awareness is a concept that consists of two notions: the notion of perceiving the *context* and the adaptivity that derives from the awareness. Adaptivity or adaptability is defined as the ability of a service/application to react to its environment and change its behavior according to the context. Context-aware computing was first introduced by Schilit and Theimer (1994) as the use of software that adapts according to the location of use, the collection of nearby people and objects, as well as to changes to such elements over time. Fickas, Korteum, and Segall (1997) define context-aware applications (called environment-directed) as applications that monitor changes in the environment and adapt the operation according to predefined or user-defined guidelines. Dey and Abowd (1999) characterize a system as context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

The functions that a context-aware application should implement (Schilit et al., 1994) are:

- **Proximate selection:** A user interface-level technique where the nearby located objects are emphasized or otherwise made easier to choose.
- **Automatic contextual reconfiguration:** A process of adding new components, removing existing components, or altering the connections between components due to context changes.
- **Contextual information and command:** Queries on contextual information can produce different results according to the context in which they are issued. Similarly, context can parameterize "contextual commands".
- **Context-triggered actions:** "If-then" rules used to specify how context-aware systems should be adapted.

Dey and Abowd (1999) propose that the features that context-aware applications may support are:

- **Presentation** of information and services to the user or use context to propose appropriate selections.
- **Automatic execution** of a service according to context changes.
- **Tagging of context** to information for later retrieval.

Related Work

The first context-aware system was the Active Badge System developed at Olivetti Research Lab. In case an employee on duty was not in his office, this would direct phone calls to the closest appliance according to the employees' location in the office environment. An evolution of this system is the ParcTab system, which was developed at the Xerox Palo Alto Research Center, relied on PDAs to offer a range of context-aware office applications (Schilit et al., 1994). Besides the above projects, a number of efforts have resulted in context-aware applications that can be categorized as follows (Dockhorn, 2003):

- **Conceptual frameworks:** They focus on the architectural aspect of context-aware systems and introduce methods of gathering, interpreting and disseminating