The Journal of Systems and Software 88 (2014) 138–146

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A context awareness framework for cross-platform distributed applications

C. Ntanos*, C. Botsikas**, G. Rovis, P. Kakavas, D. Askounis

Electrical and Computer Engineering Department, National Technical University of Athens, 9, Iroon Polytechniou str., Athens 157 80, Greece

ARTICLE INFO

Article history:
Received 17 August 2013
Received in revised form 5 October 2013
Accepted 10 October 2013
Available online 6 November 2013

Keywords:
Context awareness
Webinos
Cross-platform

ABSTRACT

With the introduction of interconnected cross-platform middleware, a new area of opportunities for ubiquitous/pervasive computing has emerged. Context aware applications can be enhanced to practically and realistically incorporate multiple facets of human–machine interactions in everyday life that are not limited to a device-centered model for deducing context. In this paper, we propose that they can rather extend this model to a human-centered, device and platform independent model, based on a personal distributed application and data cloud ecosystem. For this to be achieved, webinos, a set of web runtime extensions that enable web applications and services to be used and shared consistently and securely over a broad spectrum of converged and connected devices, is used to provide this ecosystem. The webinos Context Awareness Framework described here is accessible to each webinos-enabled application. After strict policy enforcement, it can collect contextual information, either via an automatic mechanism that intercepts native calls made by webinos applications through the various webinos APIs, via an automatic polling mechanism to these APIs, or via custom, application-specific context schema extensions. It can then distribute the contextual information from its own personal cloud storage mechanism, in the form of simple, manageable and intuitive Context Objects, to and from all webinos-enabled devices owned by the same user, or even other, authorized users.

1. Introduction

Context is a broad concept. According to the Merriam-Webster dictionary, context is defined as "the interrelated conditions in which something exists or occurs". In computer systems, this context involves every measurable metric surrounding a device, its use, or the user himself. Determining and utilizing this context is a key factor in improving human–machine interfaces. Various areas of computer science have been investigating this concept over the last forty years, in order to relate information processing and communications to aspects of the situations in which such processing occurs. We will use a definition that is specific to the context-aware computing field. "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" (Dey, 2001).

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. Note that context information is dependent on individual systems, as a type of information might be considered as context information in one system but not in another.

Context can be either a single piece, a combination, or a time series of data, for example, the current state of the compass on a mobile phone, the current state of the ambient light sensor of a laptop computer or the geographic location and the closest Wi-Fi networks. The context can then be used to make an automatic contextual reconfiguration (e.g. increase the brightness of the monitor, enable Wi-Fi connectivity) or enable a proximate selection (e.g. highlight POIs geographically located near the user or print a document on the closest printer).

A ubiquitous computing environment is characterized by a diverse range of hardware (sensors, user devices, computing infrastructure, etc) and an equally diverse set of applications, which anticipate the need of users and act on their behalf in a proactive manner (Ngo et al., 2004). The vision of ubiquitous computing is only recently becoming a reality in a scale that can be practically distributed to end users.

Being context-aware allows software not only to be able to deal with changes in the environment the software operates in, but also being able to improve the response to the use of the software. This means that context-awareness techniques

* Principal corresponding author.
** Corresponding author.
E-mail addresses: cntanos@epu.ntua.gr (C. Ntanos), cbot@epu.ntua.gr (C. Botsikas), askous@epu.ntua.gr (D. Askounis).

0164-1212/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.10.018

aim at supporting both functional and non-functional software requirements.

Three important context-awareness behaviors are (Dey et al., 2001):

1 The representation of available information and services to an end user.
2 The automatic execution of a service.
3 The tagging and storing of context information for later retrieval.

A context-aware system must be able to gather context information available from user interface, pre-specified data or sensors and add it to a repository (*Context Acquisition and Sensing*). Furthermore, the system converts the gathered raw context information into a meaningful context, which can be used (*Context Filtering and Modeling*). Finally, the system uses context to react and make the appropriate context available to the user (*Context Reasoning, Storage and Retrieval*).

Some of the main challenges in the area of context-aware computing are:

- The development of a taxonomy and uniform representation of context types.
- The infrastructure to promote the design, implementation and evolution of context aware applications; and a discovery of compelling context-aware applications that assist our everyday interactions with ubiquitous computational services.

2. State-of-the-art

Several cross-device context-aware application middleware systems have already been developed. In their majority, especially the most recent ones, these are web service-based systems. However, there exist a wide variety of middleware systems, developed mainly in the early '00s that does not rely on web services and is not designed to function under web service-based environments (Truong and Dustdar, 2009).

2.1. Non-web service-based context aware systems

1. RCSM (Yau and Karim, 2004) is a middleware supporting context sensitive applications based on an object model, where context-sensitive applications are themselves modeled as objects. RCSM supports situation awareness by providing a special language for specifying situation awareness requirements. Based on these requirements, application-specific object containers for runtime situation analysis will be generated. RCSM runtime system obtains context data from different sources and provides the data to object containers, which conduct the situation analysis.
2. The JCAF (Java Context Awareness Framework) (Bardram, 2004) supports both the infrastructure and the programming framework for developing context-aware applications in Java. Contextual information is handled by separate services to which clients can publish and from which they can retrieve context. The communication is based on Java RMI (Remote Method Invocation).
3. The PACE middleware (Henricksen and Robinson, 2006) provides context and preference management, together with a programming toolkit and tools for assisting context-aware applications to store, access and utilize contextual information managed by the middleware. PACE supports context-aware applications, in order to make decisions based on user preferences.
4. The GAIA project is a CORBA-based middleware, supporting active space applications (Roman et al., 2002). GAIA middleware provides a context service to support applications to retrieve as well as to publish contextual information.
5. CAMUS is an infrastructure for context-aware network-based intelligent robots (Ngo et al., 2004; Kim et al., 2005). It supports various types of context information, such as user, place and environment, and context reasoning. However, this system is not based on Web services and it works in a closed environment.
6. SOCAM is a middleware for building context-aware services (Gu et al., 2005). It supports context modeling and reasoning based on OWL. However, its implementation is based on RMI.
7. Citron is a context information acquisition framework for a personal device (Yamabe et al., 2005). It captures context information about a user and his surrounding environment; and the information is used to adapt the behavior of applications running on the personal device.

2.2. Web service-based context-aware systems

1. CoWSAMI is a middleware supporting context-awareness in pervasive environments (Athanasopoulos et al., 2008). CoWSAMI provides a context manager to manage context sources. Contextual information is represented as relations and defined by using context collectors, while context information can be queried.
2. The ESCAPE framework (Truong and Dustdar, 2009) is a web service-based context management system for teamwork and disaster management. ESCAPE services are designed for a front-end of mobile devices and the back-end of high end systems. The front-end part includes components supporting context sensing and sharing that are based on web services and are executed in an ad hoc network of mobile devices. The back-end includes a web service for storing and sharing context information among different front-ends.
3. Han and colleagues present the Anyserver platform, which supports context-awareness in mobile web services (Han et al., 2005). The Anyserver platform uses various types of contextual information, such as device information, networks and application type. However, it does not support direct context sharing and is not fully based on web services.
4. The inContext project (Truong and Dustdar, 2009) provides various techniques for supporting context-awareness in emerging team collaboration. It is designed for web service-based collaborative working environments. inContext provides techniques for modeling, storing, reasoning and exchanging contextual information between web services.
5. CARISMA is a mobile computing middleware, which exploits the principle of reflection to enhance the construction of adaptive and context-aware mobile applications (Capra et al., 2003). The middleware provides software engineers with primitives to describe how context changes should be handled using policies.
6. The AmbieSense looks into the future of the ambient intelligence landscape. Miniature and wireless context tags are mounted in everyday surroundings and situations. Project vision: "Relevant information to the right situation and user" (Kofod-Petersen and Mikalsen, 2005).
7. Hydrogen Context-Framework is a three-layered architecture and a software framework trimmed to the special needs of mobile devices, which is extensible to consider all kind of context information (Hofer et al., 2003).

Data control and privacy are common missing features. Most existing middleware do not help users keep control of their personal data across multiple devices, despite the rise in web applications storing data in the cloud.