Expert Systems with Applications 41 (2014) 1875–1884

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Opportunistic control mechanisms for ambience intelligence worlds

José M. Fernández-de-Alba [a,*], Pablo Campillo [b], Rubén Fuentes-Fernández [a], Juan Pavón [a]

[a] Dept. Software Engineering and Artificial Intelligence, Universidad Complutense de Madrid, Madrid, Spain
[b] Dept. Communications and Information Engineering, Universidad de Murcia, Murcia, Spain

ARTICLE INFO

Keywords:
Ambient intelligence
Opportunistic control
Context-awareness
Virtual worlds simulation
FAERIE
UbikSim

ABSTRACT

Ambient intelligence (AmI) worlds consist of heterogeneous collections of interconnected devices that integrate smoothly in the environment to offer services to their users. These devices can be added, change or fail, modifying the system topology. Also, other circumstances may change, affecting users' activities, e.g., time, location, or the presence of other users. These changes modify the information the system has available to satisfy users' needs, i.e., the context. AmI systems need to adapt to these evolving conditions in order to be able to provide their services, but being as unobtrusive as possible for their users. There are also performance requirements that the system must fulfill to provide responses to environment stimuli in real time. Opportunistic control mechanisms address these issues by monitoring the context, and suspending or resolving goals when the appropriate conditions are met. This paper presents FAERIE, a software framework that supports the development of AmI applications with facilities for context management that rely on opportunistic control. The development of FAERIE systems uses a 3D simulator tool for testing and validation called UbikSim. A case study on an artistic installation illustrates the use of this infrastructure.

1. Introduction

Ambient intelligence (AmI) worlds are composed of multiple devices that are interconnected and tightly integrated with the environment where they are embedded (Remagnino, Hagras, Monekosso, & Velastin, 2005). They provide services to their users based on the information they gather, what allows them minimizing the need of explicit actions by users. This information is called context and comes from different sources, e.g., sensors, users' input, historical data, or external sources (Abowd et al., 1999).

Building applications with this functionality implies dealing with the heterogeneity of devices and the information they manage. These applications have also to consider the evolution of user related features, such as location, time, or activities. There are also dynamic changes in system configurations. Device connections can change or fail, modifying the topology and configuration of the system. These changes, in turn, modify the available information, providing new or redundant data, or making impossible to get an accurate representation of the current context. There are also requirements of quality of service, such as performance, cost, or energy consumption, which are important for real-time applications, to improve device autonomy, or to build affordable systems (Baldauf, Dustdar, & Rosenberg, 2007).

AmI systems must deal with the previous issues requiring as little as possible user intervention for their configuration and functioning. Users should not be aware of all the devices in the environment in order to use it (Kieffer, Lawson, & Macq, 2009). Therefore, the system itself must be able to adapt dynamically to all those changes, building context representations with available information and making decisions based on the reliability of such representations.

Opportunistic control mechanisms can be used to support the design of AmI applications with these features. Following this approach, when the system receives a goal, it does not solve it immediately, but suspends it. Then, it monitors changes in the context to detect a suitable moment to solve the goal, i.e., needed resources and information are available, and the context fulfills certain conditions (Patalano & Seifert, 1997). The observation of the context continues during goal resolution, so the system can suspend the process if the goal becomes unreachable. This behavior implies the existence of mechanisms to facilitate context-awareness: components are able to determine when conditions depending on context change.

Some recent works (Huang, chan Lan, & Tsai, 2008) have proposed platforms that support these opportunistic features. They are mostly focused on the layers at the lowest levels of abstraction. These layers are related with establishing opportunistic connections among the nodes in an unstable network (Arnaboldi, Conti, & Delmastro, 2011; Boldrini, Conti, Delmastro, & Passarella, 2010), and the opportunistic use of unknown remote abstract resources (Conti, Giordano, May, & Passarella, 2010; Kurz &

* Corresponding author. Tel.: +34 696633877.
E-mail addresses: jmfernandezdealba@fdi.ucm.es (J.M. Fernández-de-Alba), pablocampillo@um.es (P. Campillo), ruben@fdi.ucm.es (R. Fuentes-Fernández), jpavon@fdi.ucm.es (J. Pavón).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.08.084

Ferscha, 2010). A further step is to prepare systems for identifying abstract conditions on their current context in order to produce abstract interpretations and to execute opportunistic behaviors (Challa, Gulrez, Chaczko, & Paranesha, 2005). This opportunistic information fusion is relevant to AmI applications where there exist behaviors dependent on multiples inputs. Although there are architectural solutions for all these levels individually, there is a lack of architectures including opportunism in every level.

The development of AmI systems does not only require infrastructure for applications, e.g., platforms or libraries, but also tools for that development. One of the most problematic issues is how to test and validate real applications due to their deployment costs. A solution to this problem is simulating most of the involved physical devices and their deployment with virtual spaces. There exist works that develop test simulations using virtual sensors (Park, Moon, Hwang, & Yeom, 2007), and others that apply 3D scenarios to manage (Shirehjini et al., 2005) or demonstrate (Bylund & Espinoza, 2002) smart spaces in specific settings. However, little work has been done regarding general platforms that use 3D scenarios for assisting in the design and validation of AmI systems.

This paper presents an infrastructure that addresses the previous issues. It includes two elements. There is a library providing the mechanisms for the opportunistic management of context in AmI applications called FAERIE (Framework for AmI: Extensible Resources for Intelligent Environments) (Fernández-de Alba, Fuentes-Fernández, & Pavón, 2012b). This is complemented with UbikSim (Campillo-Sánchez & Botía, 2012), a platform for the simulation of 3D AmI worlds, which supports the testing of FAERIE-based applications.

FAERIE provides different services for building AmI worlds (Fernández-de Alba et al., 2012b). AmI components are defined as context observers that observe and update shared context containers, which manage parts of the abstract representation of the real context. These context containers transparently coordinate among them to offer a virtual globally shared representation of the context, which is distributed among different nodes. When a context observer modifies a piece of the context representation, every other context observer interested in that piece of information is made aware of the change, which triggers successive behaviors. FAERIE uses these components to support the development of workflow-based context-aware applications. This implies that applications are designed around the definition of sets of interconnected activities involving different actors (Ardissono, Furnari, Goy, Petrone, & Segnan, 2007).

UbikSim is a platform that allows running simulations of 3D intelligent environments with different configurations (Campillo-Sánchez & Botía, 2012). It supports the specification maps of the physical spaces and their sensors and actuators. Engineers can simulate the signals of these sensors or connect them with actual external sensors. Active elements are implemented as software agents, which can represent users and system controllers of actuators. Simulations can be executed in batch mode (i.e., agents representing users have predefined behaviors) or interactively (i.e., those agents are controlled via keyboard).

The use of this overall infrastructure is shown with an application that belongs to an artistic installation. The application guides spectators through different rooms using sensors to find their positions and speakers to give them instructions.

An initial version of this paper was presented at the IDEAL 2012 conference (Fernández-de Alba, Campillo, Fuentes-Fernández, & Pavón, 2012a). This paper offers more detailed information on system components and extended experimentation.

The remaining paper is organized as follows. Sections 2 and 3 introduce the infrastructure. The former explains how FAERIE supports opportunistic control behavior, and how this support drives the design of AmI applications; the later presents UbikSim and discusses its integration with FAERIE. Sections 4–6 elaborate the case study. Section 4 presents the problem addressed in it. Section 5 describes how to design the application as a context-aware workflow whose components follow an opportunistic control. Section 6 shows the actual behavior of the application and its adaption in different scenarios. Results are discussed and compared with related work in Section 7. Finally, Section 8 presents some conclusions and discusses future work.

2. Opportunistic control in FAERIE

FAERIE adopts an opportunistic control to organize the behavior of its systems. It offers these features through components of the framework (see Section 2.1) whose functionality supports different kinds of opportunistic behavior (see Section 2.2). FAERIE also includes information on how to design systems that incorporate opportunism at the application level using the previous mechanisms (see Section 2.3).

2.1. FAERIE components

FAERIE conceives an AmI system as a set of interconnected environments. Each one contains one or more devices running components of the framework that provide services for the AmI applications or other components. They also manage private context containers that maintain the context information. Examples of environments are a smart room populated with sensors and a computing node, or a mobile device with sensing and computational capabilities.

The context is a representation of the system's environment, which includes the physical environment and the computational environment. The first one represents the things existing in the real world, such as people and external devices, and the second one represents the things existing in the computer executing the framework, such as software components. Fig. 1 represents this idea. These virtual entities are known as context elements, and work as a snapshot of the actual context.

The representation of the context follows a "context conceptual" approach to represent the context, i.e., the context represents entities and their relationships through time, as opposed to "context theoretic" approaches, in which the context describes assertions or circumstances (Anagnostopoulos, Tsounis, & Hadjiefthymiades, 2007). Among the alternatives to implement the context, FAERIE adopts an object-oriented approach (Strang et al., 2004), based on the observer pattern.

The context elements that represent the context are updated by context observers, which represent the behaviors that rule the context changes for the fulfillment of objectives (Fernández-de Alba, Fuentes-Fernández, & Pavón, 2011). The way in which context elements and observers work together is what actually implements the opportunistic behavior of the framework.

2.2. Framework-level opportunistic behavior

Opportunistic behavior (Patalano & Seifert, 1997) consists in being able to execute tasks when the context meets certain conditions, and doing it in an adapted way to those conditions. This behavior needs control mechanisms to evaluate the conditions, and to suspend and restart tasks depending on the changes. This kind of control is relevant to AmI applications because of their dynamic changes and mobile nature. These are the result of the use of technologies as common today as geographic localization or "plug and play" devices.

There are different types of opportunistic control in AmI applications. At the application level, applications perform opportunistic planning. It consists of observing the context in order to exploit the