Information Fusion 21 (2015) 100–113

Contents lists available at ScienceDirect

Information Fusion

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

Architecture for management and fusion of context information

José M. Fernández-de-Alba *, Rubén Fuentes-Fernández, Juan Pavón

Dept. Software Engineering and Artificial Intelligence, Universidad Complutense de Madrid, Madrid, Spain

ARTICLE INFO

Article history:
Available online 13 November 2013

Keywords:
Context-awareness
Context-aware framework
Distributed blackboard model
Context aggregation
Context propagation

ABSTRACT

Information in a context-aware system has diverse natures. Raw data coming from sensors are aggregated and filtered to create more abstract information, which can be processed by context-aware application components to decide what actions should be performed. This process involves several activities: finding the available sources of information and their types, gathering the data from these sources, facilitating the fusion (aggregation and interpretation) of the different pieces of data, and updating the representation of the context to be used by applications. The reverse path also appears in context-aware systems, from changes in the context representation to trigger actions in certain actuators. FAERIE (Framework for AmI: Extensible Resources for Intelligent Environments) is a framework that facilitates management and fusion of context information at different levels. It is implemented as a distributed blackboard model. Each node of the system has a private blackboard to manage pieces of information that can be accessed by observer components, either locally or remotely (from other nodes) in a transparent way. The use of the framework is illustrated with a case study of an application for guiding people to meetings in a university building.

1. Introduction

Last years have seen the proliferation of technologies for recognizing location, identity, movement, orientation, face and speech in a variety of devices. Especially relevant has been their integration in mobile devices, such as notebooks, tablets, and smart phones, and within facilities, for instance surveillance cameras and presence sensors in buildings. The combination of these resources gives rise to environments with numerous data information sources that can be dynamically assembled to solve high-level tasks.

Ambient Intelligence (AmI) [1] aims to integrate these technologies and devices to build *smart environments*. These environments combine information from sensors to infer the activities that are taking place at each moment, use that knowledge to anticipate users' needs, and build appropriate responses. In addition, smart environments are flexible enough to integrate new devices of different types and tolerate their potential failures without a noticeable configuration effort by users. This ability of systems to adapt themselves to the acquired *context* (both their own and that of users) is known as *context-awareness*. The *context* refers to any information related to people, places or objects that is relevant for the operation of applications [2]. It includes, for instance, preferences, current tasks, location and time, or available resources.

According to previous definition, it is essential for a *context-aware system* to facilitate integration of different kinds of information fusion processes. Information fusion is the merging of information from heterogeneous sources with different representations in order to obtain a more convenient or synthesized version of the information [3]. To develop this integration there exist numerous alternatives [4]. Among them, some are focused on the sources being combined [5], and other are focused on the data [6]. Each alternative has a certain impact on the adopting architecture.

Most architectural approaches to build context-aware systems use multilayered architectures [4,7,8]. These architectures facilitate the conceptualization and modularization of abstraction levels, but constrain the ability of components to work across multiple layers. They usually require complex management mechanisms for components, information and processes. Looking to overcome these limitations, researchers are considering alternative architectures with mechanisms that bring more flexibility and robustness. This is the case of works based on blackboard models [5]. However, this kind of models also presents some undesirable restrictions, for instance, the centralization in the management of context information.

FAERIE (*Framework for AmI: Extensible Resources for Intelligent Environments*) tries to overcome this issue by providing a distributed solution that implements a federated blackboard model. This model provides the view of a unique virtual blackboard for all components, hiding the details of the actual distribution of the system. This structure addresses several issues in earlier applications

* Corresponding author.
E-mail addresses: jmfernandezdealba@fdi.ucm.es (J.M. Fernández-de-Alba), ruben@fdi.ucm.es (R. Fuentes-Fernández), jpavon@fdi.ucm.es (J. Pavón).

1566-2535/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.inffus.2013.10.007

Table 1
Comparison table.

Systems/Frameworks	Context acquisition	Context modeling	Context processing	Distribution/Layers
iRoom [5]	Blackboard	Key-value tuples	Data-centric	Central server/dynamic
Context toolkit [6]	Widgets	Key-value tuples	Process-centric	p2p/Static
Context Management Framework [17]	Blackboard	Ontology-based	Data-centric	Central server/dynamic
Hydrogen [18]	Context server/point-to-point	Object-oriented	Process-centric	Central server/static
Haya's [19]	Blackboard/context graph	Key-value tuples	Data-centric	Central server/dynamic
OCP [20]	Virtual blackboard	Ontology-based	Rule-based	p2p/Static
Henricksen's [21]	Context server	Ontology-based	Process-centric	p2p/Static
Ejigu's [22]	Context server	Ontology-based	Process-centric	p2p/Static
Hermes [23]	Context server	Object-oriented	Process-centric	p2p/Dynamic
Venturini's [24]	Context broker	Ontology-based	Process-centric	p2p/Dynamic
FAERIE	Blackboard	Object-oriented	Data-centric	p2p/Dynamic

of blackboards to manage context, such as the dependency on a central server and the considerable network traffic generated. Besides, it provides to context-aware components a flexible access to information at any level of abstraction. It uses a subscription mechanism that is independent of the logical organization of systems in layers. With it, components can interpret and combine any intended information, either to increase or decrease its abstraction level. This mechanism facilitates the implementation within the components of the mentioned information fusion procedures. Moreover, since this architecture provides transparency for most of the issues related to distributed context management, it allows engineers to focus on the development of business logic, and reduces the errors not related directly with it. This will be illustrated with the case study of an application for guiding people in a building for a meeting.

The rest of the paper is organized as follows. Next two sections describe the FAERIE framework, with Section 2 presenting the architecture and Section 3 illustrating its use through the implementation of the case study. Section 4 provides a review of other context-aware architectures and compares them with our work. Finally, Section 5 draws some conclusions and issues on the evolution of the FAERIE framework.

2. The FAERIE framework

FAERIE conceives a *context-aware system* S as a set of interconnected *environments* $\varepsilon_1, \ldots, \varepsilon_n$. Each environment ε_i comprehends:

- A *physical space* p_i, which contains a single *main device* d_{i0} and a set of *peripheral devices* $d_{i1}, \ldots, d_{im}$ physically connected to d_{i0}. These devices include the sensors and actuators deployed in the space, as well as other elements that may provide additional services (e.g. a database connection). The type P is defined as the set of all possible combinations of states of devices in a physical space. The *location and coverage* of p_i are determined by the location, type, and range of the devices with sensing or acting capabilities in it, and it may change over time. This way, many physical spaces from different environments may overlap.
- A *computational space* c_i (also called *node* in a computer network), which is represented by the software framework running on d_{i0}. It contains the components implementing the logics of AmI applications. Concretely, it contains one *context container* κ_i holding the *context model*, and a set of *context observers* $o_{i1}, \ldots, o_{ip}$ that manipulate that model. The type K is defined as the set of all possible combinations of context elements in a context container.
- A set of *known environments* $\varepsilon_{i1}, \ldots, \varepsilon_{iq}$ such that each node $c_{i1}, \ldots, c_{iq}$ share a network with c_i.

Fig. 1 represents this structure. An example of connection among nodes is modeled in Fig. 2. It shows three main devices: a

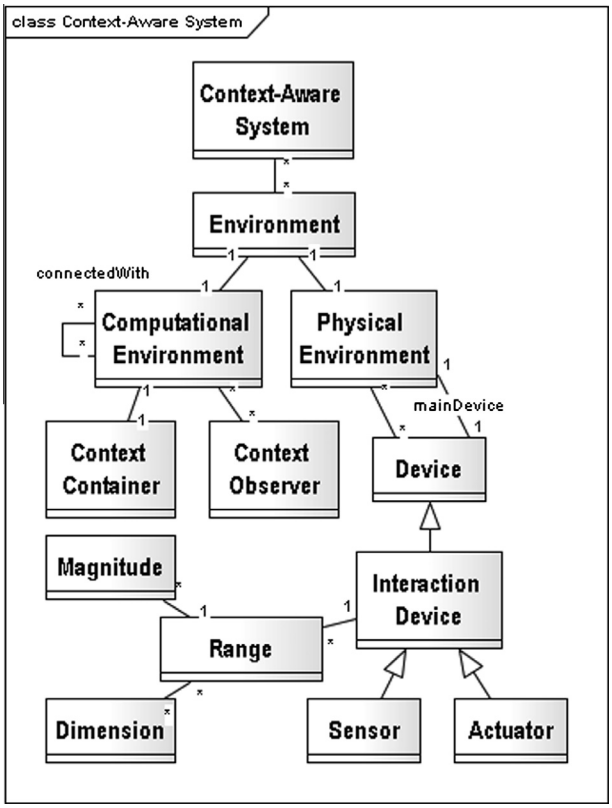

Fig. 1. Structure of a FAERIE-based context-aware system.

smartphone, a desktop computer, and a laptop. There are several peripheral devices (e.g., sensors and actuators), applications and files. Each main device runs its node, which contains its respective context container and context observers. Some of the context observers drive the peripheral devices.

A context container acts as a blackboard, and the context observers examine or update its information according to certain logic. When there is a change in the representation of the context, a notification is sent to the interested (i.e., subscribed) context observers, which are able to modify their behavior accordingly. This, in turn, may modify the current context or the behavior of the driven devices. Thus, the context observers are responsible of making the state of the context progress over time.

In a formal way, a general context observer o_j acts as a function of the form $o_j : K \times P \to K \times P$. This means, given a state of the physical environment and the context container, the observer generates a new state of the context container and changes the physical environment. Assuming a discrete division of the time,