Contents lists available at ScienceDirect

Neurocomputing

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

Implementation of context-aware workflows with multi-agent systems

Javier Alfonso-Cendón [a], José M. Fernández-de-Alba [b], Rubén Fuentes-Fernández [b], Juan Pavón [b,*]

[a] Escuela de Ingenierías Industrial e Informática, Universidad de León, 24071 León, Spain
[b] Facultad de Informática, Universidad Complutense de Madrid, Avda. Complutense, s/n., 28040 Madrid, Spain

ARTICLE INFO

Article history:
Received 10 March 2014
Received in revised form
9 October 2014
Accepted 21 October 2014
Available online 8 May 2015

Keywords:
Context-aware systems
Workflow design
Multi-agent systems
Ambient Intelligence

ABSTRACT

Systems in Ambient Intelligence (AmI) need to manage workflows that represent users' activities. These workflows can be quite complex, as they may involve multiple participants, both physical and computational, playing different roles. Their execution implies monitoring the development of the activities in the environment, and taking the necessary actions for them and the workflow to reach a certain end. The context-aware approach supports the development of these applications to cope with event processing and regarding information issues. Modeling the actors in these context-aware workflows, where complex decisions and interactions must be considered, can be achieved with multi-agent systems. Agents are autonomous entities with sophisticated and flexible behaviors, which are able to adapt to complex and evolving environments, and to collaborate to reach common goals. This work presents architectural patterns to integrate agents on top of an existing context-aware architecture. This allows an additional abstraction layer on top of context-aware systems, where knowledge management is performed by agents. This approach improves the flexibility of AmI systems and facilitates their design. A case study on guiding users in buildings to their meetings illustrates this approach.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Ambient Intelligence (AmI) systems integrate a diverse set of technologies and devices in order to provide services to users in an unobtrusive way. Raw data coming from sensors of the environment are aggregated and filtered to create more abstract information [22], which can be processed by application components at a higher level [20], in order to decide what actions should be performed on the environment [21].

This process involves several activities. For instance, finding the available sources of information and their types, gathering the data from these sources, facilitating the fusion (aggregation and derivation) of the different pieces of data [4,19], building and updating a representation of the environment with this information (what is known as the *context*) to be used by applications, and triggering actions in actuator devices [8]. This kind of environment is highly dynamic because of the changing behavior of human users, as well as rearrangements in system topology when devices fail, or are added or removed.

These activities and the adaptation to changing conditions can be organized in terms of *workflows*. These workflows are context-aware because their activities are triggered by changes in the *context*, a representation of the state of the environment. They may involve multiple actors, including systems and users, which require coordination. Adaptation here implies performing tasks according to the actual context taking into account, for instance, resources and user configuration. A correct evaluation of the context relies on systems making a proper interpretation of the available data, and using the inferred context to fill in the missing information needed by their services. This adaptation requires an infrastructure that resolves abstract representations of existing tasks into runtime processes that operate the sensors and actuators of the smart environment, either to monitor or to perform the required actions. It also needs to integrate the domain logics of different entities, both human and physical devices (see, for instance, the management of the context with mobile devices in [6]).

A suitable way to design such information workflows is by applying the concept of Multi-Agent Systems (MAS). MAS allow modeling distributed heterogeneous entities (the agents) and their cooperation (for instance, in a workflow) toward the accomplishment of common goals. The application of MAS for designing context-aware workflows in this case can take advantage of other works that have validated the use of MAS for AmI. There are works approaching this integration from general and methodological

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

http://dx.doi.org/10.1016/j.neucom.2014.10.098
0925-2312/© 2015 Elsevier B.V. All rights reserved.

perspectives focused on system development [27,29] and knowledge management [25]. Other works address specific aspects of the problem, such as sensor integration [2], the interpretation of gathered data [12], or the use of reputation techniques to evaluate the trustability of service providers and manage user' profiles and their related security issues [24].

Although MAS have been used for processing context-aware information in AmI applications [27], many of them are too focused on low abstraction levels or domain-specific tasks. For instance, agents are integrated at hardware level in [1,28]. In order to get a more systematic way to apply MAS in AmI applications, it would be convenient to be able to work with context representations at different levels of abstraction. This requires processing raw data to extract relevant information to dynamically build the context at a higher level of abstraction. This facilitates the semantic processing of information by agents in order to make consistent decisions on how to act on the environment. The organization of this process is what is described in this paper as *context-aware workflow*, because it is triggered by changes in the context and finally acts on the environment, with the corresponding effects on the context.

To put all pieces together, this paper presents an architecture where MAS manage context-aware workflows for AmI applications. The design of the MAS uses common agent design concepts, as extracted from the INGENIAS agent-oriented methodology [13]. The infrastructure for developing the AmI system is the Framework for AmI: Extensible Resources for Intelligent Environments (FAERIE) [11], which supports distributed management of AmI contexts and workflows. This framework can be downloaded from [14]. FAERIE proposes splitting the context representation into several abstraction layers. Each layer considers certain information and its processing, both horizontal (i.e., on the same abstraction layer) and vertical (i.e., between successive abstraction layers). It also establishes how to coordinate the components of these layers in order to process information, which facilitates reasoning on the context. Workflow management (i.e., detection, tracking and execution of workflows) is built upon the previous functionality. Workflows are represented as activity diagrams that operate by using abstract expressions as data. The context-aware applications can make use of all these features.

This approach is illustrated with a case study of a system that guides a user in a building to meet a person. The system has information about a map of the environment (in this case, rooms, corridors, stairs, etc., in the building), and an activity diagram that describes the guiding process (i.e., the workflow). The context information changes to show the location of people in the building. The workflow status is updated to track users and to deduce the completion degree of the workflow. The infrastructure coordinates the available sensors and actuators to perform the actions described in the workflow. A software agent is the actor responsible of the workflow for the guiding activity. This agent uses dialog management to interact with the user.

The rest of the paper is organized as follows. Section 2 reviews alternative approaches to deal with context and workflow management in context-aware systems and their tradeoffs. Section 3 presents the support developed in FAERIE for context-aware workflows, and Section 4 the software architecture for the integration of MAS with them. The case study in Section 5 illustrates its use with an application to guide a person in a building, and shows a complete execution of the workflow. Section 6 uses these results to discuss the evaluation of the approach. Finally, Section 7 presents some conclusions and proposals for the evolution of the approach.

2. Management of context-aware workflows

Most approaches for the management of context-aware workflows clearly separate the control of the workflow execution from the management of context information. The works of Ranganathan and McFaddin [23], uFlow [16] and CAWE [3] are examples of this. The first two propose wrapping the context management system and using it to check conditions in a workflow execution environment, while the third wraps the workflow management as another context provider/consumer into a context-aware architecture. The approach of Ranganathan and McFaddin uses the context-aware component in order to choose a suitable workflow definition, and then proceeds to its execution. However, it does not consider the changes in the conditions affecting that choice once it has been done. On the contrary, the other two approaches work with an abstract workflow definition, in which actions are instantiated at runtime depending on current context conditions. The approach chosen in FAERIE [11] is similar to that of uFlow, but it supports the use of any workflow definition language, as the corresponding engine is wrapped with modules that adapt the inputs and outputs as required. This is not the case for uFlow and CAWE, which describe definition languages of their own, and of Ranganathan and McFaddin, which propose using the Business Process Execution Language (BPEL), a standard language for the definition of business processes. The advantage of the first two alternatives over the third is that they include context dependent concepts. This allows defining workflows conditions and actions in terms of context information. However, the third alternative uses a well-known and established language, which facilitates its reuse in different contexts.

The use of the agent paradigm is also widespread in AmI literature [29]. For this work, it is of particular interest their relationships with the management of context information and workflows.

Regarding the management of context information, works are usually focused on solving specific tasks taking advantage of the knowledge-based capabilities of agents. For instance, the extension of the MAS Amadeus in [15] provides capabilities to learn the user's behavior. Some more general works try to provide formal descriptions of knowledge for some tasks and procedures to apply it. This is the case of [17], which provides an ontology for intrusion detection events, and uses prevention rules based on pattern detection and clustering algorithms to work on it. Finally, works like [26] provide full architectures for context management. Their main issue is their commitment to quite specific structures for the resulting systems. This is useful to facilitate development guidelines and understand the running systems, as it adjusts to a known model. However, this reduces flexibility as it imposes a structure that may not be well-suited for the problem at hand.

Illustrative examples of the use of agents to manage workflows in AmI systems are the projects of Aiello et al. [1,2], Castillo et al. [7], and CAKE [5]. The work in [1] describes a framework to program light agents for Wireless Sensor Networks (WSN), while [2] in focused on mobile agents on this domain. They do not provide support for defining and monitoring workflows, but [2] considers the detection of activities by means of body sensors. The definition of these activities is mainly achieved through assisted automatic learning, as there is no explicit definition of them. This is the same approach adopted in [7], in this case to determine activities from accelerometer data. The automated discovering and learning of activities offers great flexibility for monitoring activities and changing their definition in runtime. The drawback is the usual limited complexity of the learned activities regarding the types of actions and number of participants. The other example shown here is CAKE [5]. Its agents organize themselves using abstract workflow definitions, depending on the situation details obtained from a case-based reasoner. However, the architecture does not offer specific support to facilitate information interpretation in order to identify a suitable case. It leaves to the agent implementations the task of determining which the situation is to request the case. Anyway, this kind of work is more concerned with providing infrastructures, libraries, or procedures to develop MAS using them, than with how to use these MAS to provide context-aware services.

Download English Version:

https://daneshyari.com/en/article/406086

Download Persian Version:

https://daneshyari.com/article/406086

Daneshyari.com