A SYSTEM ARCHITECTURE FOR DYNAMIC DEVICE INTEGRATION AND COLLABORATION

Waltenegus Dargie[1]

Abstract
In this paper we present a comprehensive architecture that supports dynamic device integration and collaboration. These two aspects are useful to create and maintain smart and self-managing computing environments in which multiple devices complement each others' functionalities and harness their resources to provide users with freedom and flexibility of interaction. The architecture enables various stakeholders (users, application developers, administrators, service providers, etc.) to identify their place in creating or using a unified, consistent and wholesome system that is made up of many (and possibly heterogeneous) independent parts.

1. Introduction

Context-aware computing has gained a considerable attention in the research community in the recent past. Context-awareness has been considered in developing adaptive, multimodal interactive systems [1]; for creating smart systems [3]; for developing mobile applications [4]; for managing distributed systems [6]; for securing data integrity and building reputation systems [7]; for mobile transactions [8]; for designing configurable software components [9], and, more recently, for developing self-managing systems [2].

This is a good indication that the design, implementation and use of context-aware systems should take several concerns into account. As the system size and complexity increase, the development and maintenance task may not be the responsibility of the application developer and the user alone. Previously, context-aware systems (mostly applications) were developed with some specific tasks in mind and not much consideration was given as to how these systems should fit into a more comprehensive and more complex service landscape. As a result, their usefulness and scope is limited. This does not purport, of course, that some of the proposed middeware and smart environments, such as SOCAM [5] and Semantic Space [10], do not foresee or accommodate the integration of context-aware systems into existing infrastructures. To some extent system integration is addressed. However, the existing approaches should be extended with additional features that enable several stakeholders to access, configure, reconfigure and, if need be, modify or extend context-aware systems.

In this paper, we will report the progress we are making in developing a comprehensive conceptual architecture. The architecture is being implemented to develop a smart system that assist elderly people to live independently. The remaining part of this paper is organised as follows: in section 2., we give a brief scenario that will serve as a basis to motivate the architecture; in section 3., we will introduce the conceptual architecture; in section 4., we will demonstrate the modelling of tasks, devices and context to support collaboration; and finally, in section 5., we will summerise the implementation of the architecture and outline future work.

2. Scenario

Janet is a 70 years old retired old woman and lives by herself. Her pension does not allow her to live luxuriously, but the smart devices and appliances in her apartment (most of them being gifts from her two children) enable her to enjoy relaxed life at home and elsewhere. Below is given some typical examples of how she interacts with the devices:

1. While putting old dirty closes in the washing machine, she sees on the screen a warning message sent from the dishwasher controller. The dishwasher is running but does not pump out water. When Janet receives the message, she requests the washing machine controller to transfer the message to her interactive TV in the living room.
2. Janet enters into the living room and sees that the warning message has already been displayed on the interactive TV. Unlike the very brief warning displayed on the screen of the washing machine, this one provides a detailed

[1]Technical University of Dresden, Chair of Computer Networks, 01062, Dresden, Germany

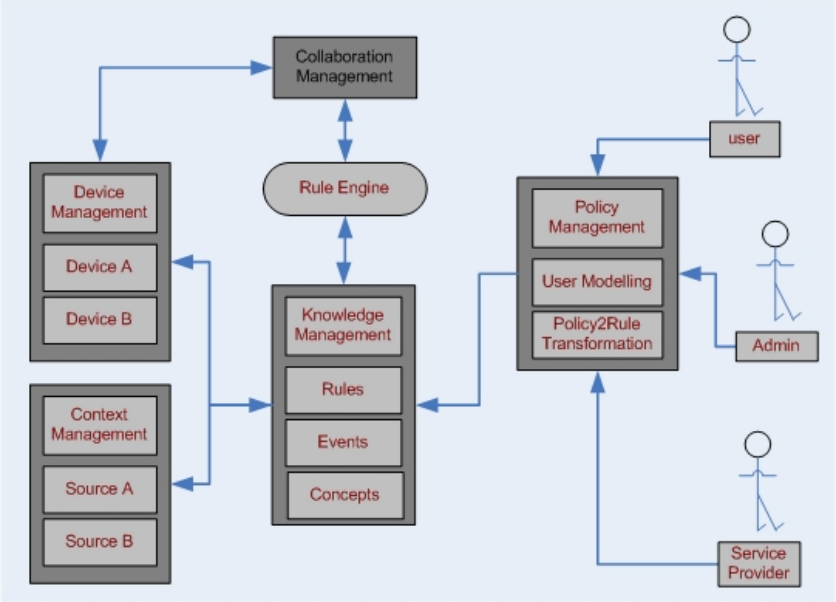

Figure 1. A system architecture for enabling autonomous decision making

description of the problem. Janet, however, could not figure our the problem, so requested the interactive TV to send the problem description directly to the vendor.

3. The vendor's "reception" service received the problem and examines its severity, which is of grade "B", implying that it needs a technician's intervention. Then it checks two prerequisite before sending out a confirmation: (1) whether the product's guarantee is still valid (in which case, it was still valid); and (b) whether there is a free technician whose free slot matches that of Janet's. The vendor's "reception" service has a limited access to Janet's calendar service, and the two services negotiate on a visit by a technician. As a result, the "reception" service sends out a confirmation. Moreover, the "reception" service connects Janet with an emergency team so that she can take immediate step to deal with the imminent problem.

3. Conceptual Architecture

The scenario above reveals three essential aspects of an autonomous, smart system: (1) Context-awareness; (2) dynamic cooperation between devices; and (3) autonomous decision making. Context-aware feature is reflected when the notification of the defect of the dishwasher is transfered to the wash machine (an active device near to the present whereabouts of Janet) and when there was a negotiation between the vendor's "reception" service and Janet's calendar service. Dynamic device cooperation enables Janet to access messages of different complexities that depend on her own activity and the device she can access; and to manage problems without much hectic and obstruction. These two features lead to autonomous decision making of various degrees. Figure 1 shows the system architecture we propose to support these three features.

The architecture consists of five management components and a rule engine. The policy management component enables users, service providers and administrators to define policies regarding the entire system's accessibility and operation. The user associates contexts with tasks. The administrator defines rules for specifying interactions and configures services and devices accordingly. The service providers provide description of their services, which include binding information.

The knowledge management component stores and manages knowledge regarding available devices and context sources and their current status. To simplify context use and modelling, we define a context to be an event that occurs inside devices and sensors. The relevant context type is defined by the user (by setting thresholds or naming a place or a person's ID). All events are entered into the knowledge management component which filters them according to their relevance to the user. The knowledge management component includes also a pool of rules that are generated by the policy management component.

The collaboration management access the knowledge management through the rule engine to determine which devices

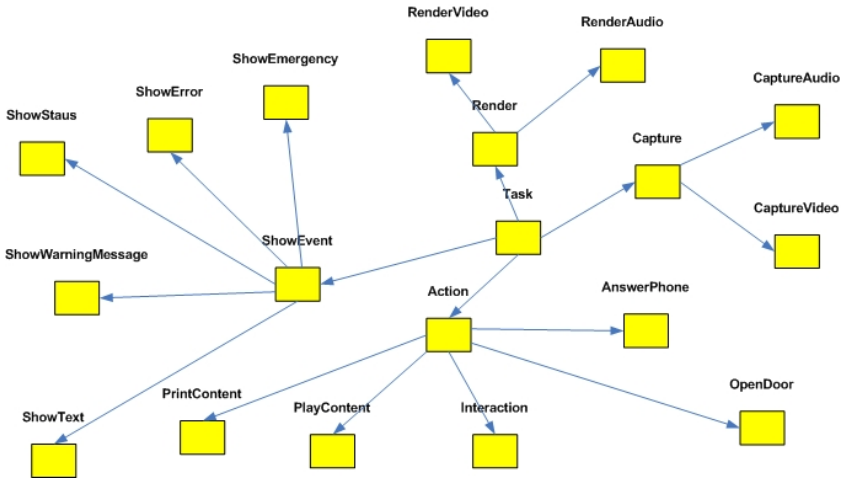

Figure 2. An overview of the task ontology

can establish relationship. For example, in a multimodal environment, the collaboration management component decides how best to present information to a user, based on the available devices that are near to the user.

The rule engine is the central part of our architecture. It gives meaning to the events that arrive at the knowledge management component and request the collaboration management component to carry out specific tasks. The device management component and the context management components register their devices by the knowledge management component and enable users to individually configure devices and context sources.

4. Collaboration

Collaboration is executed by modelling tasks, devices and contexts and by defining rules based on the concepts, relations and instances that make up these models. Tasks are modelled by someone who plays the administrator's role through the user modelling interface in the policy management component. Presently, we use ontology to define tasks and relationships between tasks and to categorise task events. Figure 2 shows a partial overview of the task model for the scenario we discussed in section 2.. Likewise, devices and relationship between them as well as device instances are defined by using a device ontology. The device ontology is created by the administrator, but it takes the device descriptions delivered by the device providers into account. This way it is possible to easily integrate devices into a unified collaborative environment. During the implementation of the conceptual architecture, we found that this was rather a difficult step, as devices and their descriptions (when available) are heterogeneous and the administrator should have a complete knowledge of the structure of the descriptions. An additional model is the context model which describes the user's social settings and physical environment.

Once these models are in place, both the user and the administrator can defined rules. At present, the user defines higher-level and user friendly policies from which rules are created by the Policy2Rule Transformation component. The higher-level policies are described using ECA rules (event-condition-action) and the Policy2Rule component transforms the ECA rules in a way that is both syntactically and semantically meaningful to the knowledge management component.

Apart from the different models that are created and managed, the administrator is also responsible to grant access rights to various stakeholders, including the users and people and agents which are related to the users. These are defined as ECA rules through the user modelling interface.

The rule engine is the central element of our architecture. It subscribes to events and rules and evaluate and fires rules whenever new events arrive. To reduce the number of events that arrive at the knowledge management component from devices and context sources[2], the user or the administrator can configure them to publish only the relevant events.

Finally, relevant tasks are carried out by the collaboration management component.

[2]Note that practically all devices are context sources and they create context events whenever their status change

5. Discussion

This work has been motivated by the premise that much freedom can be granted to users if all the computing devices and embedded systems they own can establish spontaneous cooperation based on the user's activities and whereabouts as well as events that are interesting to the users. We provided a scenario on independent living to magnify the premise and to demonstrate the usefulness and scope of the conceptual architecture we proposed.

Except the policy management component, the remaining components are implemented in an OSGi environment and a prototype was developed for the scenario described in section 2.. Initial test demonstrates that spontaneous collaboration is both useful and meaningful. Because of accessibility problem in wash machines and dishwashers, etc., we could not be able to test our prototype in real systems. But we simulate these devices. To capture the whereabouts of the user, we used IR beacons and IButton sensors. For defining the three ontology trees (device, task, and context), we used the OWL-DL ontology framework. The policy management component, particularly, the poliy2rule transformation component, becomes more complicated than we initially expected. It is left as a future work.

While initial test of our prototype was encouraging, there are, however, some open issues we would like to address in the future. Some of these are: (1) Usability: How intuitive is it for users (such as elderly people) to manage interacting with a diversity of devices? (2) Performance: Does the rule engine provide timely results so that the user can have real-time experience? and (3) How does the system behaves in events that are enirely unexpected by the user or the system administrator?

References

[1] Alexander Behring, Matthias Heinrich, Matthias Winkler, and Waltenegus Dargie. Emode - model-driven development of multimodal, context sensitive applications. *Journal of communication and cooperation media*, 6(3), 2008.

[2] Waltenegus Dargie. *Context-aware computing and Self-Managing systems*. Chapman & Hall/CRC Studies in Informatics Series, 2009.

[3] Waltenegus Dargie and Tobias Tersch. Recognition of complex settings by aggregating atomic scenes. *IEEE Intelligent Systems*, 23(5):58–65, 2008.

[4] Oleg Davidyuk, Jukka Riekki, Ville-Mikko Rautio, and Junzhao Sun. Context-aware middleware for mobile multimedia applications. In *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 213–220, New York, NY, USA, 2004. ACM.

[5] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A service-oriented middleware for building context-aware services. *J. Netw. Comput. Appl.*, 28(1):1–18, 2005.

[6] Rafik Henia and Rolf Ernst. Context-aware scheduling analysis of distributed systems with tree-shaped task-dependencies. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 480–485, Washington, DC, USA, 2005. IEEE Computer Society.

[7] Jinhwan Lee and Kwei-Jay Lin. Context-aware distributed reputation management system. In *ICEBE '08: Proceedings of the 2008 IEEE International Conference on e-Business Engineering*, pages 61–68, Washington, DC, USA, 2008. IEEE Computer Society.

[8] Nadia Nouali-Taboudjemat and Habiba Drias. A policy-based context-aware approach for the commitment of mobile transactions. In *NOTERE '08: Proceedings of the 8th international conference on New technologies in distributed systems*, pages 1–11, New York, NY, USA, 2008. ACM.

[9] Eunjeong Park and Heonshik Shin. Cooperative reconfiguration of software components for power-aware mobile computing. *IEICE - Trans. Inf. Syst.*, E89-D(2):498–507, 2006.

[10] Xiaohang Wang, Jin Song Dong, ChungYau Chin, SankaRavipriya Hettiarachchi, and Daqing Zhang. Semantic space: An infrastructure for smart spaces. *IEEE Pervasive Computing*, 3(3):32–39, 2004.