

Services Platforms for Context-Aware Applications¹

P. Dockhorn Costa, L. Ferreira Pires, M. van Sinderen and D. Rios

Centre for Telematics and Information Technology, University of Twente,
PO Box 217, 7500 AE Enschede, the Netherlands
{dockhorn, pires, sinderen, rios}@cs.utwente.nl

Abstract. Context-aware services platforms aim at supporting the handling of contextual information in order to provide better user-tailored services. This paper addresses our current efforts towards a configurable and extensible services platform for context-aware applications. It discusses the use of a language and ontologies to cope with configurability and extensibility aspects.

1 Introduction

Context-awareness has emerged as an important and desirable feature in distributed mobile systems. This feature deals with the ability of applications to utilize information about the user's environment (context) in order to tailor services to the user's current situation and needs [1].

Building context-aware systems involves the consideration of several new challenges mainly related to the gathering/sensing, modeling, storing, distributing and monitoring of contextual information. These challenges justify the need for proper architectural support. In this work, we are particularly interested in services platform architectures to support context-aware applications.

Ideally, a platform for context-aware applications should facilitate the creation and the dynamic deployment of a large range of applications, including those that are unanticipated at platform design-time. In this short paper, we briefly describe our current efforts towards a configurable and extensible services platform to support context-aware applications, which include the definition of a language to allow dynamic configuration of the platform, and the use of ontology support.

2 The Services Platform

The services platform forms the system environment for context-aware mobile applications. It supports the scenario in which context information is gathered from Context Providers (sensors or third-party information providers) and services are implemented by third-party service providers. The services platform aims at

¹ This work is part of the Freeband AWARENESS project (<http://awareness.freeband.nl>). Freeband is sponsored by the Dutch government under contract BSIK 03025.

delivering the most adequate services based on both application requirements and contextual facts (see Fig. 1). Applications describe their requirements by defining the desired services and the contextual conditions in which the services should be provided. The platform should autonomously react to reaction rules, which are defined in terms of conditions to be checked against contextual facts.

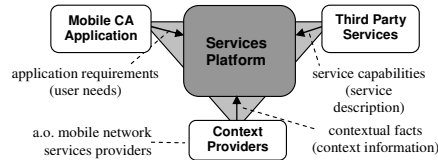


Fig. 1. Overview of the services platform

We have identified the essential requirements to be satisfied by a services platform for context-aware applications [2], which include: (i) Reactivity to stimuli from the environment: the platform should allow the specification of events, which reflect particular changes in the users’ environment (contextual changes). In addition, the platform should allow the specification of actions that are the response to (combination of) these events; and (ii) Support for context handling: the platform should provide efficient mechanisms to gather, store, distribute and monitor contextual information.

We have addressed some of these requirements when designing the platform architecture. Most of our efforts have been spent on developing an architecture with a high level of configurability. The proposed solution includes the definition of a subscription language, which allows applications to dynamically expose their needs to the platform. Fig. 2 depicts the proposed services platform architecture, which contains three main components: *Monitor*, *Registry* and *Context Interpreter*.

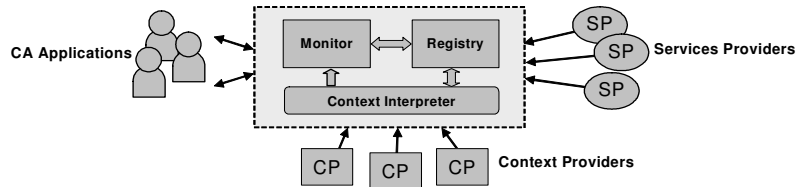


Fig. 2. Services platform architecture overview

The Context Interpreter gathers contextual information from Context Providers (sensors or third-party providers), manipulates contextual information and makes it uniformly available to the rest of the platform. The Registry maintains information necessary to support the interpretation of application requirements and the execution of services. The Monitor is the core of the platform, since it is responsible for receiving and interpreting application requests and making them active within the platform. The details of the services platform architecture can be found in [1].

In our approach, application↔platform interactions are dynamically configured through the definition of application subscriptions. In a subscription, an application is capable of dynamically exposing its requirements to the platform, which composes new tailored services from the set of available services, at runtime.

2.1 The WASP Subscription Language (WSL)

WSL is a descriptive language that we have developed in order to be able to specify application subscriptions. The main clause of this language is the ACTION-GUARD clause with which it is possible to specify condition and response actions.

The ACTION-GUARD clause defines that one or more actions should be triggered as a consequence of a correlation of events. This clause allows one to represent actions that are performed by the platform as a reaction to stimuli defined in a logical expression. To illustrate the usage of this clause, consider a simple health scenario in which a patient is being monitored for possible medical emergencies. In case an emergency occurs, help (an ambulance or a doctor) should be sent to the patient’s location; his close relatives need to be contacted; and the hospital needs to be informed of his arrival. An application can express this scenario by submitting the following application subscription to the support platform:

```
ACTION sendHelp; callRelatives; informHospital
  GUARD john.condition == emergency
```

This is just an abstract view on the ACTION-GUARD clause semantics: condition is placed after the GUARD keyword, and the actions are listed after the ACTION keyword. More information on the language can be found in [1].

2.2 Ontology Support

The unambiguous comprehension of the meaning of contextual information among system parts is essential for the correct operation of context-aware systems. The current platform support for knowledge representation is restricted to UML Class diagrams [1]. This approach may lack in expressiveness and rigor required to model context-aware systems.

With ontologies, we can formalize the properties and structure of contextual information to guarantee common semantic understanding among system parts. In addition, ontologies offer inference and reasoning mechanisms that are necessary to derive complex contextual information, and reason about the context, respectively [3].

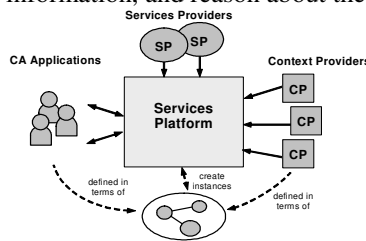


Fig. 3. Services platform using ontology support

We are currently evaluating the applicability of ontologies to our services platform. Fig. 3 presents an extension of the services platform introducing the use of ontologies. In this figure, the Context Providers provide the context information by means of messages defined in terms of a set of ontologies explicitly referenced, allowing the

platform to understand these messages by using the semantics represented in the referenced ontologies. The context-aware applications subscribe to services provided by the platform by means of context specifications in the application subscription. We would like support ontologies not only to formalize context but also to exploit it for simple service descriptions and more complex service composition.

The expected benefits of using ontologies are (i) Flexibility: knowledge is defined in terms of an ontology instead of “hardcoding” within the platform; (ii) More intelligent behavior: knowledge can be derived from the factual knowledge explicitly represented in the ontologies; (iii) Semantic interoperability: semantics of the (possibly several) languages used by the platform external parties can be defined in terms of a set of interrelated ontologies; and (iv) Expressiveness and consistency checking: context information is represented using a formal representation language, which enables to automatically check the consistency of the models.

3 Final Remarks

We have briefly presented in this paper our efforts towards a configurable and extensible services platform for context-aware applications. Our approach supports the configuration of applications↔platform interactions at runtime. In order to allow dynamic configuration of interactions we have introduced WSL, which is a descriptive language developed especially for this purpose.

We have seen that WSL presents limitations, especially with respect to the definition of more complex event interrelations. The platform allows the definition of single independent events and the correlation of them using logical operations. Currently these events are tight to one specific application subscription. We would like to be able to define events that depend on other events (event composition) and also to specify events that can be reused by other application subscriptions. In our current efforts we consider applying rule-based languages (e.g. Jess [4]) to support more complex event interrelationships.

Furthermore, we argue that UML Class diagrams may lack in expressiveness and rigor required to model context-aware systems. We have briefly discussed the benefits of using ontologies within the services platform.

References

1. Dockhorn Costa, P., Ferreira Pires, L., van Sinderen, M., Pereira Filho, J.G.: Towards a Services Platform for Context-Aware Applications. In: S. K. Mostefaoui et al (eds.): Proc. of the 1st Inter. Workshop on Ubiquitous Computing (IWUC 2004). Portugal (2004)
2. Dockhorn Costa, P., Pereira Filho, J.G., van Sinderen, M.: Architectural Requirements for Building Context-Aware Services Platforms. In: E. Halasz et al (eds.): Proc. of 9th Open European Summer School on Next Generation Networks (EUNICE 2003). Hungary (2003)
3. Rios D., et al.: Using Ontologies for Modeling Context-Aware Services Platforms. In: Workshop on Ontologies to Complement Soft. Arch. (OOPSLA 2003). USA (2003)
4. Friedman-Hill, E.: Jess in Action: Java Rule-based Systems. In: Manning Publications, Greenwich, July 2003.