Available online at www.sciencedirect.com
ScienceDirect
Pervasive and Mobile Computing 4 (2008) 481–505
pervasive and mobile computing
www.elsevier.com/locate/pmc

Composing software services in the pervasive computing environment: Languages or APIs?☆

Jon Robinson*, Ian Wakeman, Dan Chalmers

Department of Informatics, University of Sussex, Brighton, UK

Received 30 October 2006; received in revised form 7 December 2007; accepted 2 January 2008
Available online 6 January 2008

Abstract

The pervasive computing environment will be composed of heterogeneous services. In this work, we have explored how a domain specific language for service composition can be implemented to capture the common design patterns for service composition, yet still retain a comparable performance to other systems written in mainstream languages such as Java. In particular, we have proposed the use of the method delegation design pattern, the resolution of service bindings through the use of dynamically adjustable characteristics and the late binding of services as key features in simplifying the service composition task. These are realised through the Scooby language, and the approach is compared to the use of APIs to define adaptable services.

Keywords: Pervasive computing; Programming languages; Performance evaluation

1. Introduction

In this paper we present the Scooby Service Composition system.[1] The main contribution of this research is a service composition language for pervasive computing

☆ Funded under the EPSRC projects Nathab GR/S26408/01 and TrustUs GR/S69016/01.
* Corresponding author.
E-mail addresses: j.r.robinson@sussex.ac.uk (J. Robinson), ianw@sussex.ac.uk (I. Wakeman), d.chalmers@sussex.ac.uk (D. Chalmers).
URL: http://www.informatics.sussex.ac.uk/softsys/ (J. Robinson).
[1] Scooby is our acronym for *Service Composition Objects Ordered By You.*

1574-1192/$ - see front matter
doi:10.1016/j.pmcj.2008.01.001

environments. Past research in distributed systems suggests that separating service and configuration is a viable approach [1,2], but pervasive computing provides a new set of challenges because of connectivity, the need to take account of context in choosing services and the dynamicity of the services within any pervasive computing environment. Scooby attempts to provide programming abstractions and design patterns that make it easy for programmers to compose new services that are context-aware, and provides a platform to explore the following questions: *how do we combine services which have been produced by different developers?* and *How can we compose services to meet the demands of users?*.

1.1. Our research goals

One of the goals of our research has been to determine if the coupling of a domain specific language and middleware is an effective way to enable programmers to create services and compositions. In Fig. 1 we outline a basic Scooby service which provides the ability to route messages to different devices depending on their co-location with the user. The current user's location is used to dynamically rebind to the various output services. When the stock service provides a notification the event handling code routes this to the most appropriate output at that time. The example highlights the simplicity of the Scooby domain specific language, when considering the alternative of programming such a service in another high-level language, such as Java with the use of APIs to access middleware functionality. In Sections 4 and 5 we describe the Scooby language and constructs in more detail.

There are a number of technologies that can provide the building blocks towards such a system, including service discovery, remote invocation and messaging systems, such as CORBA and J2EE. However, we have opted to take the approach where we are not directly reliant on such technologies as we are targeting the system towards low-powered devices with limited processing power, such as PDAs and devices that can be embedded within home appliances. When taking the nature of the environment and the limited scope of the devices available into account, using heavy-weight technologies is not the most practical or viable way forward. Instead, we have chosen to utilise the publish/subscribe paradigm as the method for relaying event information between devices, using the content-based router Elvin [19]. This provides a light-weight communication medium on which we can then build our middleware. One of the characteristics of a pervasive computing environment is the ad hoc combination of devices and intermittent and unpredictable availability of devices. There is no guarantee of the device being available over the course of time due to a number of physical reasons such as network disruption, the device dropping outside of the influence of a wireless network and power loss/power saving modes. The adoption of a publish/subscribe mechanism therefore would not have an impact on the environment when taking into account the lack of message guarantee. Additionally, the utilisation of key-value pairs available within publish/subscribe is heavily relied upon when disseminating service characteristics.

The Scooby language acts as the medium in which a user can compose services. As the language and resultant compiler are lightweight, much of the complexity of implementation is pushed down to the middleware level instead of the language. If there were to be millions of services and thousands of events per second, then such an approach