Adding Context-Aware Behaviour to Almost Anything: the Case of Context-Aware Device Ecologies

Seng W. Loke, Evi Syukur, Peter Stanski
School of Computer Science and Software Engineering
Monash University, VIC 3145, Australia
swloke@csse.monash.edu.au

Abstract

This position paper advocates a general approach to adding context-aware behaviour to applications, devices, objects, and appliances. We describe how a context-awareness framework (such as MHS) can be used to augment ecologies of devices with context-aware behaviour.

1. Introduction

Context-awareness has emerged as an important idea for achieving automatic behaviours in pervasive systems. For example, a system that senses a user's context (e.g., location and physical actions, time, etc) and reacts to it provides convenience for the user and acts as an invisible interface for driving the system's behaviour. Emerging research has begun to look at context-aware systems more generally, independently of specific applications, including context middleware and toolkits [3,4,7] and ontologies for describing context [5] which aim to be applicable for building different context-aware applications. Indeed, a simple abstract view of a context-driven system is depicted in Figure 1 below where the context-aware framework (which can be "plugged into" different applications) acquires the contextual information, optionally reasons with it and then issues commands or feeds contextual information to the application which then takes appropriate actions (perhaps after its own reasoning), and might then provide feedback to the context-aware framework. The decoupling of the context-awareness framework from the application itself means that better frameworks can be used replacing earlier frameworks, or that new applications, devices, etc can be added to an environment with an existing framework.

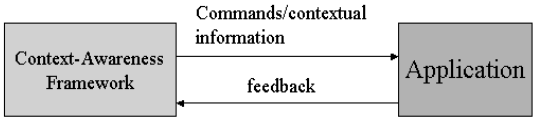

Figure 1 Abstract View of a Context-Driven Application

This paper explores this view, presenting a general approach to adding context-aware behaviour to applications, devices, objects, and appliances.[1] For example, applications or even appliances can be wrapped up as Web services. The Web services effectively provide an interface (in a standardized format) to these applications and appliances through which control or information can be issued from the context-awareness framework. We also illustrate our approach by showing an application which adds context-aware behaviour to what we call device ecologies [10], i.e. collections of devices (including appliances in the home, etc) which are in relationship with each other (detailed in Section 2), using a framework that acquires user context which we call the Mobile Hanging Services (MHS) framework for context-aware mobile services first presented in [15] (detailed in Section 3). In Section 4, we describe different kinds of context-aware behaviour possible with device ecologies using MHS. We conclude in Section 5.

2. Device Ecologies and Device Ecology Workflows

The American Heritage Dictionary defines the word "ecology" as "the relationship between organisms and their environment." We envision a computing platform that takes the form of *device ecologies* consisting of collections of devices (in the environment and on users) interacting synergistically with one another, with

[1] Replace "Application" by "device" or "appliance" or "object with an embedded Web server" in Figure 1.

users, and with Internet resources, undergirded by appropriate discovery and communication infrastructures that range from Internet-scale to very short range wireless networks. These devices will perform tasks and work together and will need to interact with the user from time to time.

Modelling Devices. We model the observable and controllable aspects of devices as Web services as done in [11]. Such device modelling is not inconsistent with emerging standard models for appliances such as the AHAM Appliance Models [1], where each appliance (such as clothes washer, refrigerator, room air conditioner, etc) is modelled as a collection of objects categorized according to subsystems. We assume that aspects of devices can be directly observed and controlled by means of a collection of Web services, described using the Web Services Description Language (WSDL). We note that there will be aspects of the device which are not exposed as Web services. Service actions on devices in the UPnP device model [9] are also modelled similarly.

Device Ecology Workflow. We consider a device ecology workflow or *decoflow* first presented in [10] for someone (say, Jane) involving a television, a coffee-boiler, bedroom lights, bathroom lights, and a news Web service accessed over the Internet. Figure 2 describes this decoflow. The dashed arrows represent sequencing, the boxes are tasks, the solid arrow represents a control link for synchronization across concurrent activities, and free grouping of sequences (i.e., the boxes grouped into the large box) represents concurrent sequences.

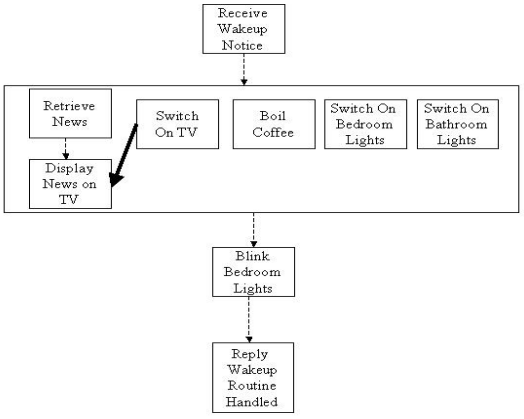

Figure 2 A Device Ecology Workflow

This decoflow is initiated by a wake-up notice from Jane's alarm clock which we assume here is issued to the Device Ecology Workflow Engine (which we call the *Decoflow Engine*) when the alarm clock rings. This notice initiates the entire workflow. Subsequent to receiving this notice, five activities are concurrently started: retrieve news from the Internet and display is on the television, switch on the television, boil coffee, switch on the bedroom lights, and switch on the bathroom lights. Note the synchronization arrow from "Switch On TV" to "Display News on TV", which ensures that the television must be switched on before the news can be displayed on it. After all the concurrent activities have completed, the final task is to blink the bedroom lights, in order to indicate to Jane that the decoflow tasks have completed. Such a decoflow can be scripted in a language such as BPEL4WS [2] which is an XML language for specifying business process behaviour based on Web services. BPEL4WS has language constructs for manipulating data, handling faults, compensating for irreversible actions, and various structured activities. BPEL4WS assumes that there is a central engine which is executing the workflow (e.g., BPEL Engine[2]). A BPEL4WS process or workflow is viewed as the central entity invoking Web services associated with its partners and having its own services invoked by partners (e.g., when the partner initiates the workflow or receives results for a previously sent request). For example, the following fragment shows a sequence of two invocations one to retrieve news and another (to display the news) on the TV, which is part of the decoflow of Figure 2 as specified in BPEL4WS.

```
<sequence>
   <invoke partnerLink="newsRetrieval"
           portType="lns:newsUpdatePT"
           operation="requestNews"
           inputVariable="newsRequest"
           outputVariable="newsInfo">
   </invoke>
   <invoke partnerLink="tv"
           portType="lns:tvControlPT"
           operation="displayOnTV"
           inputVariable="newsInfo"
          <target linkName="tv-to-news"/>
   </invoke>
</sequence>
```

3. Adding Context-Awareness – the MHS Framework

The simplest way to execute a decoflow as shown above is to employ a workflow engine that is networked with the devices in the home

[2] Two such engines are http://www.collaxa.com/ and http://www.alphaworks.ibm.com/tech/bpws4j

and the Web. Such workflows can be triggered by operations on particular devices or by the user manually via a user interface to the decoflow engine. However, since the devices are uniformly treated as a Web service just like other resources on the Internet (e.g., the news services), one can utilize a context-awareness framework to drive or trigger operations on individual devices or to initiate decoflows. For example, depending on the perceived context (of the user say), a user interface to a particular device or decoflow can be automatically downloaded to the user's mobile device, or particular operations on devices can be invoked or decoflows triggered. Below, we briefly describe the MHS context-awareness framework we have built which can be used to provide user context information to drive devices and decoflows. More details of MHS are found in [15].

Context-Awareness. The definition of context in this paper refers to a specific entity or property in the environment that can be used to help characterize the situation of a person or trigger behaviour. From [6,8,14], we observe that context can be divided into four categories: (A). Computing context: This can be any information that relates to the computing environments such as computer resources (e.g., printers, scanners and workstations), available bandwidth, and network connection. (B). User context: User context refers to any knowledge about the users, such as a user's profiles, intentions, location, and current social activity (e.g., in a meeting). (C). Physical context: Physical context has a general definition that involves any physical entities and properties in the user's environment. For example, lighting, temperature, traffic conditions, and noise levels. (D). Time context: Time refers to the current time and the user's schedule (in terms of days, weeks, months and years) that can be used to determine the mobile application's behaviours.

Harnessing Context-Awareness for Computing Relevant Services. The MHS framework provides a mechanism to map contextual information to relevant services. This high level architecture is illustrated in Figure 3 below.

The MHS framework relieves developers of context-aware applications from the low-level task of matching services with context (including the user location[3]), handling the computing of the set of services relevant to the given contextual information about the user. We describe the four most important components of the system: **A. Services Calculator.** The Services Calculator component computes, with a set of rules,[4] the set of services appropriate to the currently perceived context, using the services' descriptions database.[5] In addition, a set of services can also be computed using the information that is stored in a history log file. For example, at time t, the system takes information about the user's current context (which can include device resources) and history (e.g., a user's previous whereabouts and services used at those places) to compute the set of services that would be useful to the user at time t. This computation can be modeled as a function `f`, where

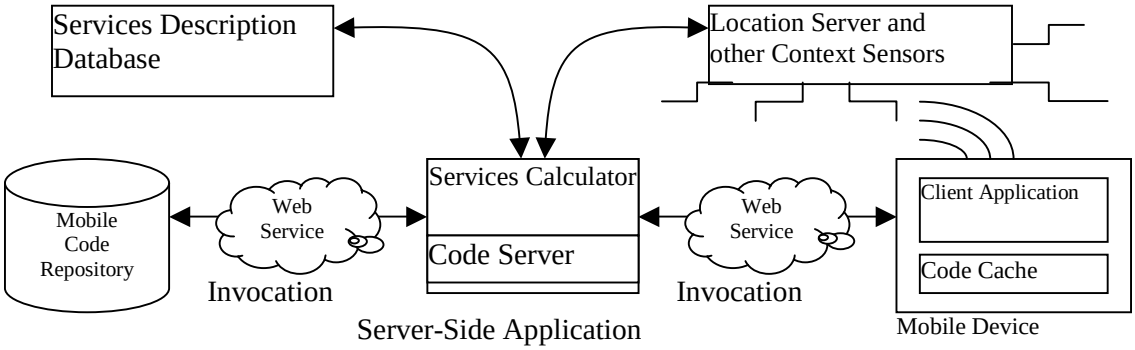

Figure 3 High Level Architecture of the MHS framework

```
f(current_context, history) = set of relevant services
```

A question is: *how often or when should* `f` *be computed?* There are two possible answers depending on how dynamic the environment is.

[3] The location server we employed in our current MHS is the Ekahau Positioning Engine (www.ekahau.com). Other positioning technologies can be also be used.

[4] The rules we use simply maps situations to services directly but more sophisticated ontology-based matching can be used.

[5] The database can be virtual and generated by querying for the descriptions of services in the current set of devices in a device ecology.

The first approach is to compute the function `f` when a change in `current_context` (e.g., someone's location) is detected. However, if a new service is just added for that location, the system will not be able to detect that particular service, as the user's location remains the same. The new service will only be detected if there is a change in the current context. The second approach is to recompute `f` periodically even though there might not be a change in current context,[6] thereby updating the list of services every 10 seconds. With this approach, the user will definitely be able to see new services that is just added to that location context. However, relatively more resources are needed for this approach. To proactively compute a new service based on context history - a mechanism that keeps analyzing the user's behaviour over a time span can be useful. For example, every Wednesday, the user comes to the office at 3 PM. The system records this information in the history log file. As the system works out what situation the user will be in, the system could then predict the set of required services proactively before the user gets into that situation (e.g., switch on the room light and equipment at 2.58 PM). This will avoid lag time between the user getting into a situation and the user being able to use a service. **B. Code Cache.** Code cache refers to the mobile code application that is stored on the device for future re-use. The idea is that once the list of services relevant to the current context has been computed and shown to the user, the user can then select from the list to use a service, which results in the mobile code for invoking the service being downloaded onto the mobile device – we have just-in-time downloading of the service's client code. In terms of managing the code cache on the mobile device, the following function `c` is considered, where

```
c(current_context, history) = code remaining in the cache
```

The function `c` above describes that at some time t, `c` takes information about the user's current context and history, to determine what application-code should remain in the cache i.e., what should be deleted. **C. Code Server.** Code server refers to the mechanism that handles a user's request from a device, responding to that request by transferring the relevant mobile code. In this case, the code server retrieves the mobile code that matches with the user's request and sends it back to the mobile client. **D. Client Application.** Client application resides on the mobile client side that manages the incoming mobile code and prepares to execute the code on the device.

[6] The set of services available for a location is not considered part of the context but used in the computation of `f`.

4. Context-Aware Device Ecologies with MHS

The MHS can be used to provide context-awareness to the device ecologies in the following ways:

(1) Depending on the perceived context, directly invoke services of a device. For example, news is fetched and displayed on TV when the user enters the living room (as detected by a location system) for the first time in the morning.

(2) Depending on the perceived context, directly perform actions on some decoflow, such as initiating a decoflow, stopping a decoflow, suspending the decoflow, resuming a decoflow, etc. For example, the decoflow in Figure 2 can be initiated not only when the alarm clock rings, but when the user has actually got off the bed (as detected by a weight sensor of the bed), and has entered the bathroom (as detected by a location system).

(3) Depending on the perceived context, assuming the user is carrying a mobile device, automatically download code to the user's device, where the code contains

(i) a user interface to particular device(s) or decoflows and

(ii) service client code to invoke operations on devices and decoflows.

Downloading of such code presumes a device on the user, whether it is a PDA or something smaller (such as a watchPC), and provides the user a means of manually accessing appropriate devices and decoflows from the user's device. Such code might also provide status information to the user, about the progress of decoflows.

The MHS framework already provides the facilities for **(3)**. For **(1)** and **(2)**, where device operations or actions on decoflows are directly triggered (without human intervention), the MHS system not only computes the list of suitable services but needs to be augmented with an appropriate module to then automatically invoke these suitable services and actions (which can be done entirely on the server-side without involving the mobile device). The module maps perceived context to operations (e.g., Web service calls) on devices, decoflow engine, appliances, or software applications.

Also, all three types of actions **(1)**, **(2)** and **(3)** might be carried out – for example, particular appliances are switched on immediately, a decoflow is executed to adjust over time devices in the room to create a comfortable ambience, and to search for suitable news and information of interest (and to pay for them) and to display them on specific devices, and the user device is endowed with code to control particular appliances (e.g., T.V. and radio). Alternatively **(3)** can be used as a means to involve the human in the loop in a decoflow – to supply parameters to operations or to obtain user authorization for particular operations.

5. Conclusion and Future Work

We have presented a practical application of context-aware device ecologies, with examples in home automation. But as important is the general approach of wrapping resources (e.g., appliances, workflow engine, Internet resources, etc) as Web services so that these can be driven by a context-awareness framework. Other context toolkits and middleware can be used besides MHS in order to provide the context sensing capabilities, depending on the application. Also, new devices added to an environment which has been fitted with a context-awareness framework can acquire context-aware behaviour by being exposed to the framework. For example, a new soft-toy (which supports a Web service that can be invoked to make it say "hello") brought into a house (with positioning engine) can be made to greet its owner each time the owner enters the same room as the toy.[7] Proactive devices, with appropriate authorization, should also be able to subscribe to the context-awareness framework in order to "gain awareness" about other people or objects in the environment. The extent to which devices and decoflows can be controlled by context depends on what functionality has been exposed as services. Future work will consider conversational web services with devices [12] and context-triggered jumps in decoflows [13]. The MHS framework has been implemented and we are working with integrating it with devices and the decoflow engine (which is also being implemented).

[7] We think that this can be done with minimal set-up effort, but it is beyond the scope of this paper.

References

[1] AHAM. Connected Home Appliances – Object Modelling, AHAM CHA-1-2002.

[2] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S. Business Process Execution Language for Web Services (version 1.1), May 2003.

[3] Campbell R. H., Ranganathan A., A Middleware for Context-Aware Agents in Ubiquitous Computing Environments, Proc. of the ACM/IFIP/USENIX Int. Middleware Conf., Rio de Janeiro, Brazil, June 2003.

[4] Chen H. An Intelligent Broker Architecture for Context-Aware Systems**,** PhD proposal in Computer Science, University of Maryland Baltimore County, USA, January 2003.

[5] Chen H., Finin T., Anupam J. An Ontology for Context-Aware Pervasive Computing Environments, Workshop on Ontologies and Distributed Systems, IJCAI-2003, Acapulco, Mexico, August 2003.

[6] Chen, H. and Kotz, D. A Survey of Context-Aware Mobile Computing Research. Dartmouth Computer Science Technical Report TR2000-381. Available at ftp://ftp.cs.dartmouth.edu/TR/TR2000-381.pdf

[7] Dey A. K. Providing Architecture Support for Building Context-Aware Applications, PhD thesis, November 2000, Georgia Institute of Technology.

[8] Dey A. K., Abowd G. D., Towards a Better Understanding of Context and Context-Awareness, GVU Technical Report GIT-GVU-99-22, June 1999. Available at ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-2.pdf

[9] Jeronimo, M. and Weast, J. UPnP Design by Example, 2003, Intel Press.

[10] Loke, S.W. Service-Oriented Device Ecology Workflows. Proc. of the Int. Conf. on

Service-Oriented Computing, LNCS 2910, (eds.) M. Orlowska, S. Weerawarana, M.P. Papazoglou, and J. Yang, pp. 559-574, Trento, Italy, December 2003, Springer-Verlag.

[11] Matsuura, K., Hara, T., Watanabe, A., and Nakajima, T. A New Architecture for Home Computing, In Proceedings of the IEEE Workshop on Software Technologies for Future Embedded Systems (WSTFES'03), pp. 71-74, Japan, May 2003.

[12] Petrone, G. Managing Flexible Interaction with Web Services. Proc. of WSABE 2003. Available at http://www.agentus.com/WSABE2003/program/petrone.pdf

[13] Reichert, M., Dadam, P., and Bauer, T. Dealing with Forward and Backward Jumps in Workflow Management Systems, Softw. Syst. Model, Vol 2, pp. 37-58, 2003.

[14] Schilit, B., Adams, N. and Want, R, Context-Aware Computing Applications. In Proc. of IEEE Workshop on Mobile Computing Syst. and App., pp. 85-90, Santa Cruz, California, December 1994. IEEE Computer Society Press.

[15] Syukur, E., Cooney, D., Loke, S.W., and Stanski, P. Hanging Services: An Investigation of Context-Sensitivity and Mobile Code for Localised Services. Proc. of the 2004 IEEE Int. Conf. on Mobile Data Management (MDM 2004), USA, January 2004, IEEE Computer Society Press.