Personalized Dynamic Composition of Services and Resources in a Wireless Pervasive Computing Environment

M. Howard Williams, Yuping Yang, Nick Taylor, Sarah McBurney, Elizabeth Papadopoulou, Fiona Mahon and Micheal Crotty

Abstract— **A wireless pervasive computing environment needs to dynamically adapt its functionality and behaviour to changes in the resources and services available to a user at any point in time. For a mobile user this means that as the user changes location, the pervasive environment should take account of any changes to the services that are available. Thus initially when the user requests a service the most appropriate service must be provided, but as the user moves around, the service provided could change. To handle this, a wireless pervasive system needs to provide dynamic service composition (and re-composition) based on the user's personal preferences and current context. The Daidalos project is developing a platform to support pervasive services, which provides dynamic personalized service composition. This paper describes the problems and the role of personalization in the approaches adopted in Daidalos to deal with dynamic service composition and re-composition. The basic ideas have been prototyped and demonstrated, and are currently being integrated and extended.**

Index Terms— **service composition, personalization, pervasive, dynamic adaptation.**

I. Introduction

PREVIOUS work in the areas of distributed computing and mobile computing has led naturally to the notion of pervasive computing [1]. In particular, the proliferation of heterogeneous communication networks and devices is enabling people to interact with one another through a growing number of devices at different locations [2]. This together with the growth in sensing technology (and the availability of low-cost sensors) and the advances in networked devices provide the impetus for developing pervasive computing environments where the user can take advantage of these developments while being protected from the complexity underlying them. Pervasive computing needs to be minimally intrusive and yet dynamically adapt its functionality and behaviour to changes in the environment, taking account of user preferences and current context [3], [4].

Thus there has been considerable interest in the issue of adaptation and how this might be achieved in a pervasive computing environment for the benefit of mobile users. One important aspect of this is the automatic composition of services at runtime. This process is a major component of a number of pervasive environments, e.g. GAIA [5], AURA [6], PCOM [7], etc. Similar ideas are also being explored in other areas such as multimedia [8] and the web services community [9].

Personalization is another important feature that goes hand in hand with adaptability. In the case of a wireless pervasive computing system, the system needs to adapt its functionality and behaviour in accordance with the context of the user [10]. As the context of a user changes, a pervasive system should react to the changing context, and adapt its behaviour when necessary. Personalization is key to this adaptation, and the system needs to keep track of a user's personal preferences and their dependence on context, and to adapt its behaviour to meet the needs of the user with minimal user intervention

This close link between adaptation and personalization in pervasive computing systems is recognized in Daidalos, an integrated research project funded under the European Sixth Framework Programme with 45 partners. Daidalos [11] stands for "Designing Advanced Interfaces for the Delivery and Administration of Location independent Optimized personal Services". Its goals are:

- to develop and demonstrate an open architecture (based on IPv6) that will combine diverse complementary network technologies in a seamless way, and
- to develop a pervasive computing environment on top of this to provide pervasive services.

This paper focuses on the dynamic composition and re-composition of services for the mobile user in a wireless pervasive computing environment, and describes the approach being used in the Daidalos project. This approach takes account of the initial composition of services and resources to meet a user request as well as subsequent re-composition as the context of the user changes. The following section describes the problem of composition and re-composition, and

Manuscript received July 28, 2005. This work was supported in part by the European Union under the FP6 programme (Daidalos project) which the authors gratefully acknowledge.

M. H. Williams is with the School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, UK (phone: +44 131 4513430; fax: +44 131 4513327; e-mail: mhw@ macs.hw,ac.uk).

Y. Yang, N. Taylor, S. McBurney and E. Papadopoulou are with the School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, UK (e-mail: ceeyy1, nkt, ceesmm, ceeep1@macs.hw.ac.uk).

F. Mahon and M. Crotty are with the Telecommunications Software & Systems Group, Waterford Institute of Technology, Waterford, Ireland (email: fmahon, mcrotty@tssg.org).

outlines part of a scenario being used by Daidalos to illustrate the functionality involved. Section III describes very briefly the architecture adopted in Daidalos. Section IV gives a brief account of the strategy adopted for personalized selection and composition of services. Section V discusses some of the issues of re-composition while section VI provides details of the approach used to respond rapidly to changing QoS. Section VII summarizes and concludes.

II. Service and Resource Selection and Composition

The basic problem that is being addressed here is that of selection and composition of a set of services and resources to best meet the user's needs and preferences. In Daidalos the focus is on a limited set of services, networks and devices that might be available in any particular context to meet a specific user request at any particular time. The types of requests being dealt with relate mainly to telecommunication services. Thus we are concerned with choices made between (at most) tens of options rather than hundreds or thousands of possible options, which might occur in the case of choosing between web services to meet a specific request for information.

The particular request may be made directly by the user or indirectly on the user's behalf. For example, a user may want to send a message to another user, watch a video screen or initiate a third party service. Alternatively, an application may want to do something on behalf of a user. For example, consider a patient with diabetes who is wearing a blood sugar monitor. If the blood sugar level goes out of bounds (too low or too high), the system may want to send a message to the user/her parents/her doctor/any other appropriate person in the vicinity.

When the request is made in a pervasive environment, the system needs to determine what services and resources are available to meet the request. Occasionally a simple service discovery process may suffice; however, in general this will not be the case. The proliferation of devices, networks and services referred to earlier is more likely to lead to a variety of different options in terms of services and resources that can be used to meet the request.

At this point one needs to take into account the user's context and preferences, as well as the attributes of the services and resources. Quality of Service (QoS) is important in this regard (as recognized in systems such as Q-CAD [12], which perform the selection to optimize QoS) although other factors such as cost and preferred service supplier also need to be taken into account in making this decision. Equally important is the context of the user (at home, at work, driving a car, etc.) as this will affect the user's preferences and priorities.

Having made the selection of services and resources needed to meet the user's request, these are composed to create a single composite service which can then be executed. However, in a pervasive environment the problem does not end here.

As mentioned previously, an important aspect of a pervasive computing environment is its ability to adapt as the context of the user changes. Most important and obvious in this regard is the location of the user – a pervasive environment needs to be able to support mobile users. For example, suppose that a user is traveling in a car. As he/she moves, the QoS of the network being used may drop. Another network may become available with better QoS. In a pervasive environment one would expect the system to adapt and change the underlying network without consulting the user, However, this is not quite as simple as it sounds as once again the context and user preferences need to be taken into account, and the network switched without any significant interruption.

Within Daidalos we have built up a number of scenarios to guide the development of our pervasive environment. The following is a very brief excerpt from one of these, which illustrates the nature of the problem that is being addressed:
"Bart is a chauffeur working for company X. One morning he is sitting at home watching a newscast on his PC when a call comes in from his boss. The system puts the newscast on hold and connects the call to his mobile phone. His boss asks Bart to go to the airport urgently to collect a client. While still talking to his boss, Bart goes to his car. When he enters the car, his call is transferred automatically and seamlessly to the car phone. Bart drives off. When the call is finished, the system transfers the suspended newscast to the car and continues where it left off, only in audio mode since Bart is driving. When Bart approaches the airport, the system automatically connects to the flight information system to get information on the client's flight arrival time and relays this to Bart…"

III. Architecture of Daidalos

This section provides a very brief description of the architecture adopted in the Daidalos project to handle the integration of heterogeneous networks and the provision of a pervasive environment.

The architecture is based on two platforms. At the lower level one has the Service Provisioning Platform (SPP), which is responsible for the low-level functionality needed to integrate heterogeneous networks and to provide a range of multimedia and support services. At the upper level one has the Pervasive System Platform (PSP), which is responsible for providing the pervasive environment and support for pervasive services. This paper is concerned with the PSP.

The PSP is composed of six main software components, namely:

(1) *Pervasive Service Management (PSM)*. This component is invoked whenever a service is requested directly by a user or indirectly on the user's behalf. It is also called whenever conditions change that significantly affect the services that are running. Its function is to discover the services that will best fulfill the user's needs, select those that best fit the user's preferences and compose them into a single composed service that will best meet what the user requires.

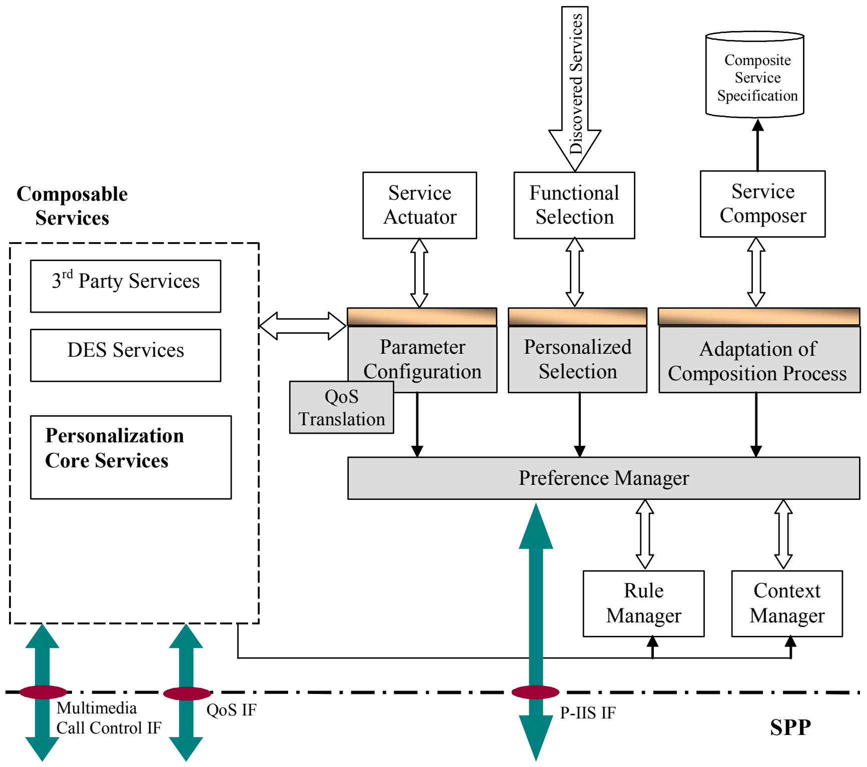

Figure 1. Personalization subsystem architecture

(2) *Personalization (P)*. This component is responsible for managing the user's preferences and using them to provide the user with a personalized experience of the services. It is responsible for the selection of most appropriate services for the PSM, for tailoring these to the user's preferences and for part of the composition process. In addition to this it is responsible for various other functions such as personalized redirection of communications and the learning of new user preferences (or refinement of existing ones).

(3) *Context Management (CM)*. This handles all the context information relating to users as well as to other objects (such as devices). It keeps track of simple attributes such as the location of a user or the status of a device as well as more complex ones such as the set of rules expressing the preferences of a user.

(4) *Rule Manager (RM)*. Some of the control of the PSP is expressed in the form of rules. These range from system-defined rules that are user independent and determine what action to take in particular circumstances, to user-defined rules that represent the specific actions required by an individual user. The RM is responsible for managing these rules - storing them, selecting them and executing them.

(5) *Event Manager (EM)*. Whenever something significant happens, an event occurs. Whether this be generated by the user (e.g. a user request for a service) or caused by a change in context, the result is that an event is generated. This component monitors events and informs other components when a relevant event has occurred.

(6) *Security and Privacy Manager (SPM)*. This component is responsible for maintaining the security and privacy for the user in the PSP. It does this through the use of virtual identifiers and controls access to personal preferences and context information on this basis.

This paper is concerned primarily with Personalization and its use in the PSM. The architecture of the Personalization subsystem relevant to this is shown in Figure 1.

IV. Personalized Service Selection and Composition

From the architecture described in the previous section, when a request for a service is received this is passed to the PSM to deal with. The PSM can be regarded as consisting of two main parts:

(a) *Service and Resource Discovery* is responsible for finding possible services and resources that can be used to satisfy the user's request.

(b) *Service Selection and Composition* selects the most appropriate of these and assembles them to create a composite service that will satisfy the user's request.

The request is first processed by the Service and Resource Discovery component, which does not attempt to apply any filtering based on user preferences but merely searches for all possible services and resources that can be used. This follows a conventional service discovery approach. The result returned by this component is a list of services and resources. This is passed to the Service Selection and Composition component. Here three important processes are carried out.

A. Personalized Service Selection

The first step carried out by the Personalization subsystem is to rank the candidate services found by the Service Discovery Service and select the one which best satisfies the constraints in the current context. To make this choice a rich set of metadata must be employed.

The main criteria used are the user preferences. The user may have specific requirements on the cost, speed, QoS, location, mobility, etc. of a service. Moreover, these requirements will frequently depend on the context of the user. For example, if the user is at work, he/she may prefer the service with the highest QoS or services provided by a particular supplier or even a specific service. On the other hand if the user is at home, he/she might prefer the cheapest service subject to a minimum level of QoS.

In addition to user preferences there are also a number of system default criteria. These are used to guide the selection of services in such a way as to improve performance of the composite service. An example might be to choose component services that are located near each other to reduce the overhead incurred in transferring data and in communication among component services.

In the future the set of criteria will also include system inferred criteria, derived from monitoring the user's behaviour.

The result of this is an ordered list in which the individual components satisfy the user's preferences although we cannot guarantee that the composite service produced from these will do so too. For example, individual components may satisfy QoS (or cost) constraints but the composite service produced from them may not. Thus once the component services have been identified, the personalized selection process needs to apply a "global" selection mechanism to the resulting composite services. This is done by accumulating vectors of all the execution paths, and constructing a matrix, which is used as the basis for selecting an optimal execution path from multiple candidate ones.

B Service Parameter Configuration

Once an appropriate set of services has been selected, the Personalization component personalizes any service that allows itself to be personalized through an interface for personalization. For such a service the service parameters are configured according to user preferences. These personalizable parameters will vary with service.

The advantage of this is that it allows the way the service presents information, interacts with the user or functions to be personalized. For example, a restaurant finding service may provide a list of restaurants to the user in different order, layout, font and colour depending on the preferences the user has concerning both restaurants and display.

The representation chosen for such parameters involves both identifying the parameter and passing across a value (which may be a complex structure). To handle this, an ontology (based possibly on DAML-S) will be adopted in the future although currently a simplified parameter list is used. The parameter value itself can be wrapped as a common object which effectively hides the concrete details and facilitates the definition of the interfaces.

Besides taking account of the user's preferences, interaction between the services selected also needs to be taken into account in the parameters. For example, if a photo service is connected to a large screen, it may display multiple images simultaneously; otherwise, it will display them one at a time. In this case, the photo service needs to be parameterized to take account of the size of the display. In other words, the context of a service may need to be taken into account when parameterizing a service.

Personalized service parameterization can be static or dynamic depending on when the values of the parameters are set. Static parameterization refers to the situation when the personalizable parameters of a service are configured before the service is executed. On the other hand, if any of these parameters change during the course of execution (due to change in context or in user preferences) this is dynamic parameterization - e.g., the resolution of an image may be decreased if it is transferred from a user's PC to his mobile phone.

C Adaptation of Composition Process

After the parameters of the individual services have been set, the component services are assembled into a single composite service. Here personalization plays a further role in that sometimes the composition process itself may be affected by user requirements. This may occur in one or more of the following ways:

(1) The user may wish to constrain when and where a task can be executed.

(2) Under certain circumstances a particular component service may be added to or removed from a composite service.

(3) The logical order in which component services are placed needs to be changed to suit user preferences.

An example of the latter is that a user may not want to book airline tickets until the accommodation at the destination is booked.

In order to handle this, the script corresponding to a composite service that allows itself to be adapted, needs to provide an interface with its process description. Based on the user preferences, the Personalization component acts upon the description of the composition process and adapts it appropriately.

V. DYNAMIC RE-COMPOSITION

As noted previously, one of the fundamental assumptions in the Daidalos pervasive environment is that it will handle user mobility. Thus as the user moves around, his/her context will change and this can affect the composed service. The simplest example of this is the device that is used by the user. In the example given in section II, Bart is talking to his boss on his mobile phone. When he gets into his car, the call is transferred to the car phone. Similarly, the newscast that was suspended on Bart's home PC, is continued in audio mode on the car PC. And so on. Although it is not always necessary to re-compose the whole service (see section VI), the situation often does require this.

Besides the devices used, the service itself may need to change. In particular, as the user moves around, different networks may become available (or the existing networks may no longer be available), and this may result in network-dependent services needing to be changed. This extends not only to different networks but also to different network types (e.g. WLAN, DVB-T, TD-CDMA). Alternatively, better or cheaper services may become available.

The pervasive system monitors any changes that might give rise to a change in the composed service, and when such an event occurs, the system re-composes the service in a manner that is transparent to the user. Such a change may be the availability of a service or it may be that the preferences themselves have changed, since these are context-dependent and any change in the user's context may result in a change in the preferred service

VI. Optimizing QoS

When the context of a user or device changes, it may be necessary to re-compose the services selected to meet a user request. However, this is a time consuming process and in some cases could be problematic to the user.

An obvious example of this is when the QoS changes. Suppose that the user is mobile and is engaged in conversation with another user on his mobile phone. Suppose further that the performance of the network gradually degrades until it becomes necessary to consider an alternative network. Instead of resorting to re-composing the whole service it is more useful if the system could simply replace the network by an alternative choice without affecting anything else. To achieve this Daidalos needs to provide a lightweight mechanism to enable rapid response to changes in QoS due to the underlying network.

Once again the role of Personalization in this process is to ensure that a user's preferences are taken into account. For this purpose it determines the user's preferences relating to the choice of network and stores a copy of them in the underlying layer (SPP) via the Intelligent Interface Selection (IIS). This copy is updated by Personalization whenever the associated user preferences are changed. In the case that multiple networks are available, the IIS selects one according to the user preferences stored locally taking into account factors such as the price, QoS, provider, etc. of the network. The way in which this operates is as follows:

- The first time the user logs on to a device, a set of default network preferences will be used for establishing network connections.
- During the bootstrapping process, Personalization passes to the SPP the last saved network preferences which are used for the initial network configurations. Once the CM is able to retrieve user network preferences from the remote server, a local copy of user preferences should always reside on the terminal until the user finishes using the device.
- Personalization has a mirror of user preferences, in the case that network connection is temporarily not available and the local CM cannot obtain user preferences from the remote server. Personalization will send IIS the locally stored preferences, based on which IIS will set up network connections.
- Personalization updates the IIS with new preference information whenever the user changes his/her network preferences.

When multiple networks (e.g., WLAN, DVB-T, TD-CDMA) are available, the personalization information stored in the network layer will be used as the basis for choosing one which best matches the user's preferences. The change of personalization information may lead to network handover.

VII. Conclusion

This paper describes the role of personalization in the approach used within Daidalos to handle the dynamic composition and re-composition of services and resources in a wireless pervasive environment to meet the needs of the mobile user with minimum user intervention. The basic approach has been implemented and demonstrated in December 2004. A fully integrated system is currently being prepared for demonstration in December 2005. A simple scenario has been described to give a flavour of the functionality of the system. This and a number of other simple scenarios will be demonstrated at this time.

Acknowledgment

The authors thank all our colleagues in the Daidalos project developing the pervasive system. Apart from funding the Daidalos project, the European Commission has no responsibility for the content of this paper. The information in this document is provided as is and no guarantee or warranty is given that it is fit for any particular purpose. The user thereof uses the information at his sole risk and liability.

References

[1] M. Satyanarayanan, "Pervasive computing: vision and challenges," *IEEE PCM*, 8(4), 2001, pp. 10-17.

[2] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265(3), pp. 94-104, 1991.

[3] J. Sun, "Mobile ad hoc networking: an essential technology for pervasive computing," in *Proc. Int Conf on Info-tech & Info-net*, Beijing, China, 2001, pp. 316-321.

[4] A. Zaslavsky, "Adaptability and interfaces: key to efficient pervasive computing," in *NSF Workshop on Context-Aware Mobile Database Management*, Providence, Rhode Island, 2002, pp. 24-25.

[5] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. Campbell and K. Nahrstedt, "A Middleware Infrastructure for Active Spaces," *IEEE Pervasive Computing*, 1(4), 2002, pp. 74-83.

[6] D. Garlan, D. Siewiorek, A. Smailagic and P. Steenkiste, "Project Aura: Towards Distraction-Free Pervasive Computing," *IEEE Pervasive Computing*, 1(2), 2002, pp. 22-31.

[7] M. Handte, C. Becker, and K. Rothermel, "Peer-based Automatic Configuration of Pervasive Applications," in *Proc. IEEE Int. Conf on Pervasive Services 2005 (ICPS'05)*, Santorini, July 2005, pp. 249-260.

[8] X. Gu, K. Nahrstedt, R. Chang and C. Ward, "QoS-Assured Service Composition in Managed Service Overlay Networks," *Proc. 23rd IEEE Int Conf. on Distributed Computing Systems*, 2003, pp. 194-204.

[9] B. Raman and R.H. Katz, "An Architecture for Highly Available Wide-Area Service Composition," *Computer Communication Journal*, 26(15), 2003, pp. 1727-1740.

[10] M. H. Williams, I. Roussaki, M. Strimpakou, Y. Yang, L. MacKinnon, R. Dewar, N. Milyaev, C. Pils, and M. Anagnostou, "Context Awareness and Personalisation in the Daidalos Pervasive Environment," in *Proc. IEEE Int. Conf. on Pervasive Services 2005 (ICPS '05)*, Santorini, July 2005, pp. 98-107.

[11] B. Farshchian, J. Zoric, L. Mehrmann, A. Cawsey, H. Williams, P. Robertson, and C. Hauser, "Developing Pervasive Services for Future Telecommunication Networks," in *Proc. WWW/Internet 2004*, Madrid, Spain, October 2004, pp. 977-982.

[12] L. Capra, S. Zachariadis, and C. Mascolo, "Q-CAD: QoS and Context Aware Discovery Framework for Mobile Systems," *Proc. IEEE Int. Conf. on Pervasive Services 2005 (ICPS'05)*, Santorini, July 2005, pp. 453-456.