Context-aware Dynamic Personalised Service Re-Composition in a Pervasive Service Environment

Yuping Yang[1], Fiona Mahon[2], M. Howard Williams[1], Tom Pfeifer[2]

[1] School of Maths and Computer Sciences, Heriot-Watt University, Riccarton, Edinburgh, EH14 4AS, UK
yang_yuping@hotmail.com, mhw@macs.hw.ac.uk

[2] Telecommunications Software & Systems Group, Waterford Institute of Technology, Waterford, Ireland
{fmahon, tpfeifer}@tssg.org

Abstract. A pervasive environment needs to take account of a user's context and preferences in determining which services to provide to the user. Moreover, one of the important features of a pervasive service environment is its dynamic nature, with the ability to adapt services as the context of a user changes. In particular, as a user moves around, new services may become available or existing services may cease to be available. A user's requirements or preferences may depend on context attributes such as location and time, and hence the user's requirements will change as these do. This paper describes how these requirement changes can be sufficiently accounted for by using a personalisation component to 'decide' what a user needs, and a composition component to continuously monitor services and the changes associated with them. The paper presents how services can be recomposed dynamically if the changes in context require it. This approach has been incorporated into a platform to support pervasive services. The architecture for this and the service composition process used is described, and the way in which personalisation is incorporated into this process is shown. The paper finishes by providing a brief account two prototypes built as a proof of concept for these ideas.

1 Introduction

In developing solutions to handle the requirements of mobile users, the environment is becoming increasingly complex. The range of different services available to the user is growing rapidly. So too is the number of devices that can be used to access different kinds of services, and the different networks that can be used to communicate between users and applications. The user is therefore presented with a myriad of choices; too many to make the use of the environment a pleasant one, unless there is some intervention.

The goal of pervasive computing is to provide an intelligent environment to support the user in this increasingly perplexing task. Where possible, such an environment should help to select the most appropriate services for any particular user, and to employ the most relevant devices and supporting networks.

The situation becomes more complex when the mobility of the user is accounted for. As the user moves around the environment, the services, devices and/or networks that he/she is using may cease to be available, or new services, devices and/or networks may become available that the user would prefer to use. To complicate matters further, the user's preferences may depend on their location and thus as they move around, the service, device or network that is the most appropriate might change. In fact, location is merely one of the many attributes defining the user's current context and other context attributes including time or current activity can also affect the choice of service, device or network. Thus one of the most important functions of a pervasive environment is the dynamic discovery, selection and composition of services to meet a user's needs and preferences, and their reselection and re-composition if circumstances change [1, 2].

This paper presents a solution to the problems described above. Dynamic personalised re-composition of services allows the user to interact seamlessly with the pervasive environment, while hiding the complexity contained within it. The paper concerns itself with how personalisation is involved in service composition and re-composition, and how personalisation adapts services to user context and requirements in this dynamic process.

The service composition process is the process of creating customised services from existing services and resources by a process of dynamic discovery, integration and execution of those services in a planned order to satisfy a request from a client [3, 4]. Service composition should be context aware and personalisable, while it should be able to use rules and policies for making the composition decisions [5, 6]. On the other hand personalisation itself is developing rapidly. With the growing interest in context awareness and pervasive systems, personalisation functionality has evolved to take account of different aspects of the users' context, in an environment populated by many small, networked devices that can sense users' situation anytime, anyplace [7-9].

Daidalos is a large European research project that is developing a pervasive system for a mobile environment [10]. The Daidalos system incorporates personalisation in various forms, including its use in service composition. This approach benefits from exploiting user context including user preferences such as interests, expertise, workload, tasks, location, in order to increase the service flexibility and support intelligent personalisation and adaptability features. In the personalisation framework, the potential for seamless context-aware adaptation in the emerging service engineering framework will support the provision of customised value-added services that will address the high and continuously increasing customer demands.

The goal of the research reported in this paper was to develop the approach required for a general system to handle personalised service composition and re-composition dynamically depending on the preferences and changing context of the user. Based on this a sequence of prototypes with increasing functionality have been created and used to demonstrate the suitability of this approach. The next section gives an overview of efforts made in the areas of Personalisation and Composition. Section 3 describes the platform, to which the components described in the paper belong. Sections 4 and 5 detail the Personalisation and Composition components and how they work together to achieve dynamic service re-composition. Section 6 details the implementation of the prototypes built as a proof of concept for the ideas de-

scribed in this paper. Overall, the paper shows how important personalisation is to the whole process of service management in a pervasive service environment.

2 Related Work

Much of the work on composition to date, is on a static composition of services, where requirements of the composition do not change and the state of the constituent services remain the same.

Chakraborty [11] describes a static composition where a particular composition is attempted and achieved, end of story. There is no accounting for continuous change of the component services during the lifetime of the composed service. It is therefore not a sufficient solution within a mobile environment.

Tosik [12] describes a service composition with some level of management. This management however focuses primarily on usage privileges and QoS. Since Tosik deals with web services, which are inherently static, he does not expand into an area of composition management of services with volatile availability.

Casati [13] in his paper on 'eFlow' describes a dynamic service composition, which includes personalisation. The personalisation involved assumes user input of their requirements. The composed services are not technically dependent on each other, although they do complement one another. For example, in Casati's service composition, a service to organise a charity ball would be composed of services to book the banquet, order the invitations from the printers, and start the advertising campaign. Although the paper describes 'dynamic composition' it focuses more on the ability to change the definition of the composition, and so the dynamic composition is over many compositions over a period of time, and not within a single composition. The 'dynamicity' described does not include monitoring of the composition during the lifetime of the composite service, although the composition definition in 'eFlow' can be changed during the lifetime of a composite service. Real dynamic re-composition based on continually changing user requirements is not addressed.

Though personalisation has been an area studied for many years, there is little work on how to incorporate personalisation into the process of service composition in a pervasive environment.

Sheng et al. [14] define a personalised composite service specification architecture, based on which users can specify their needs by adjusting existing process templates. However, this approach requires a user to locate process templates and annotate them with contextual information. Thus, quite a few user interactions are involved when orchestrating a composite service, which is not very realistic in a pervasive environment where a large number of service compositions may occur dynamically.

The Tivoli Personalized Service Manager [15] developed by IBM provides an integrated infrastructure of software products for Internet service provisioning. It offers the ability to generate web pages for specific devices, allows users to personalise portal home pages, and provides services with functionality such as calendar, agenda and address book which can be used by ISPs (Internet Service Providers) to develop their own additional services. Its localization feature provides the capability to trans-

late into different languages. However, this software product only takes into account user's profile and preferences – while more dynamic context information such as time, current activity and people in the vicinity is not considered. In addition, it only allows ISPs to utilize the simple services supplied by the product itself and does not provide the essential function of composing any existing services/resources provided by other parties.

The SPE (Secure Persona Exchange) framework described by Brar and Kay [16] provides personalized services to users in ubiquitous computing environments based on user preferences stored on mobile devices. Like [15], it does not take account of dynamic contextual data while achieving personalization.

In summary, although a lot of work has been done independently in the areas of service composition, and in personalisation, there is little work done in exploiting the capabilities of one for the benefit of the other. Real dynamism seems to be missing from the composition solutions proposed to date. Dynamism is essential for a mobile environment, since the mobile environment itself is never static.

3 The Daidalos Pervasive Service Platform

The Personalisation and Composition components described in this paper, are part of the Daidalos overall platform architecture. The architecture of the Daidalos Pervasive Service Platform (PSP) can be seen in Figure 1. The PSP is the platform in Daidalos that adds pervasiveness to all services deployed in the Daidalos network. The PSP is analogous to an Applications Container in that it facilitates a certain amount of behaviour, without the service developers needing to control this behaviour. In the case of the PSP, it facilitates pervasive behaviour. In the context of this project, this means that

- user behaviour is continuously monitored using various environment sensors
- inferences are made by the system based on this information, which can then be used to create network behaviour specific to the user
- network behaviour is stored as rules in the system
- the service can be managed, started/stopped, transferred from node to node based on user preferences which are context aware
- the service can be personalised by having attribute values assigned, based on the user preferences which are context aware

There are six main components in the PSP, which are as follows:

Context Manager (CM): The task of the CM is to retrieve, process and provide context information. Context is the set of information that describes an identity's preferences, profiles and current situation. The Context Manager connects to various sensors and provides the context in a unified manner to interested parties.

Rule Manager (RM): The RM focuses on the management and processing of rules. Rules describe a set of events that have to occur and conditions that have to be met in order for some actions to be triggered. Users employ rules as part of the personalisation process while services use them to be notified about particular changes in the user's environment.

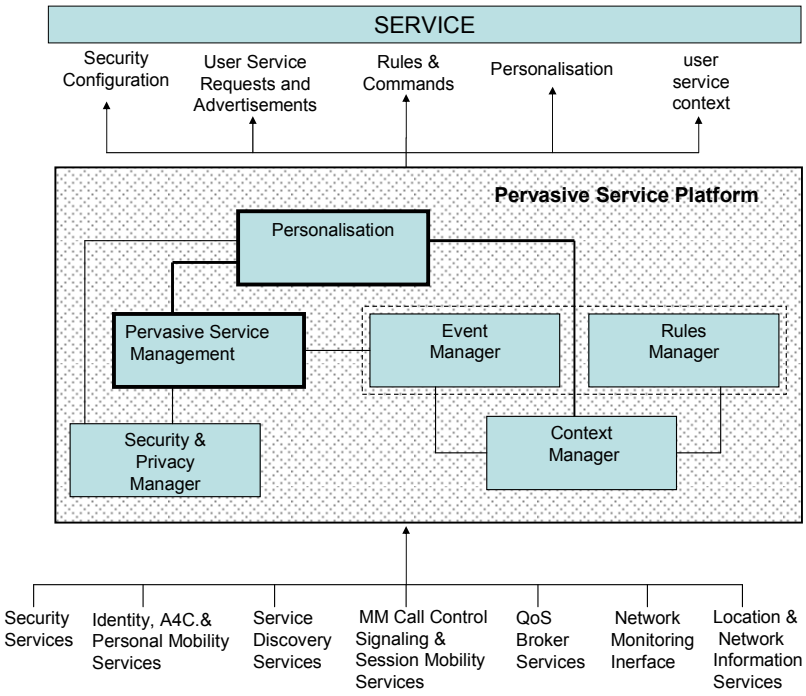

Figure 1 Pervasive Service Platform (PSP) Architecture

Event Manager (EM): The EM collects and distributes events. Events occur when context changes or when they are created by rules. Typically, events are used by the Rule Manager or by services that want to be notified about changes.

Personalisation (P): This tailors the services to the user's preferences. Personalisation takes into account preferences that the user has stated explicitly and also infers preferences to make services more personalised.

Pervasive Service Manager (PSM): This is responsible for service discovery, selection and composition. It is used to find available services, select them based on the user's context and compose them to a service session that can fulfil the user's task. It also continuously monitors composed services during their lifetime based on service availability and contextual relevance to the user.

Security and Privacy Manager (SPM): The SPM ensures that the user's real identity is not revealed to untrustworthy parties. Instead, it manages virtual identities that hide the user's real identity. Additionally, this component provides access control to user information (such as context) and the use of services.

All of these components described above interact and co-operate to form the PSP. The combination of their functions creates a unique pervasive service environment, and provides a fully functional, integrated solution for context-aware, personalised, rule-based and event-driven service discovery and composition. In the context of this paper, **P** depends on **CM** to determine a user's context, information that is vital to **P.**

P also uses **RM** and **EM** to execute rules to cause certain behaviour, based on a user's context, for example retrieval of specific services. **PSM,** which is the component responsible for service composition, uses **RM** and **EM** to monitor when service adaptation and re-composition should occur. The main topic of this paper is how Pervasive Service Manager uses Personalisation when deciding what services should be composed together, and how Context Management facilitates that. Therefore, for the rest of this paper we will not be considering the functionality of the other components in any detail.

4 The Role of Personalisation in Dynamic Service Management

In the context of the Daidalos pervasive platform, the major objectives of personalisation in the system lie in service selection, configuration of service parameters, and adaptation of the composition process based on user context and preference rules. In addition, it provides the core services whose focus is on improving the personalisation aspects of other third-party services as well as deriving new user preferences from the user's behaviour.

4.1 Personalised Service Selection

When a request for a service is passed to the Service Discovery component, it will in general return more than one candidate service, each of which is able to fulfil the same specific task required by a user. Due to the diversity of services, expecting the user to determine the most appropriate one among them would be unrealistic and time consuming for the user. Thus, the system needs to make a decision to select one for the user. This occurs during the process of service selection. In general for a composite service $S_c = S_0 + ... + S_t$, a list of services $S = \{S_0,...,S_n\}$ may be discovered, any of which could be used as a component service S_k of S_c. Given the user preferences and context, a personalised selection is performed that best suits the user's specific goal. Selection criteria used in our system include:

- User specified criteria: Users may have specific requirements on the cost, speed, QoS, location, mobility, etc. of a service. These may depend on where the user is located, what devices/networks are available, what mode the user is operating in, and so on. These requirements provide guidelines for finding an adequate match. The list of discovered services are ranked according to the criteria (e.g., rank the services from the lowest price to the highest one) and the highest ranked service is chosen as a component service (e.g., the cheapest service). Moreover, a user might have an explicit preference for a specific service (e.g., a specific provider's wireless network). In this case, if the specific service is available, it can be simply selected as a component service.
- System criteria: These criteria are used to improve the performance of a composite service. For example, selecting component services that are located close to each

other would reduce the amount of data transferred and reduce communication time among component services.

As described further in section 6, a limited set of simple criteria are used to choose services in the current prototype. The situation can become complicated when several criteria need to be combined together. How to make a rational compromise among conflicting criteria (e.g., the fastest service may not have the best QoS) is an issue that needs to be resolved in the next phase of this work.

4.2 Personalised Service Parameterisation

In order to personalise the individual services selected for a specific composition, the appropriate attributes need to be passed to each service as parameters. There are two categories of service parameters:

- Operating Parameters: These parameters are used by a service to control its functioning as well as running process implicitly. They are used to describe inherent properties of a service and thus their values can not be modified by others. These parameters are represented as $OP_s = \{p_0, \ldots, p_i\}$.
- Personalisable Parameters: While offering its major functions, a service may want to attract users by providing particular individual features. In order to cater for different users, it allows some parameter values to be configured according to user related aspects. These parameters are called personalisable parameters, represented as $PP_s = \{p_0, \ldots, p_j\}$, which are used to characterise a service and improve its performance. Compared with operating parameters, personalisable parameters can be configured with new values.

In order to distinguish between operating parameters and personalisable parameters, a service parameter needs to have a property indicating whether or not it is personalisable. Each parameter p_k has the form

$p_k = (pname, pvalue, pcategory)$

where *pname* and *pvalue* are the name and value of a parameter respectively, and *pcategory* indicates whether or not the parameter is personalisable.

A service that allows itself to be personalised, needs to provide interfaces for personalisation. In order to know what features of the service can be customized, a standardized interface is required which tells the personalisable parameters PP_s of this service. To determine the parameter values, user context and his/her preferences related to this service are analysed and appropriate values are decided accordingly (e.g., the QoS of the WLAN network service is set high when the user is watching an important football game). A parameter may have a default value, which applies to the situation, for example, the user has no special requirements. It can be overridden by the specific value derived from user preferences/context. An interface which allows setting the parameter values is also needed from the service.

Service parameters, including their names, meanings and types, may vary with services. Due to the variety of services and their degree of dependence on each other, it would be very difficult to interpret service parameters without a standard definition.

Thus, for each service type we need a list of general parameters that have common definitions and shared meanings across all services with this type. This parameter list acts as an ontology to be referred to by services that want their parameters to be personalised. In our first prototype the parameter list defined is of a simplified form, but is currently being developed using an ontology language (e.g., DAML-S).

The type of a service parameter varies from very simple ones (e.g., integer) to complex ones (e.g., data set), and thus an appropriate data representation is required to describe parameter values. An example of a complex service parameter is the QoS of multimedia services, which consists of traffic specification and reservation specification. These two specifications can be further broken down into peak rate, bucket size, minimal policed unit, maximum packet size, etc. In the prototype a parameter value is wrapped as a common object which effectively hides the concrete details and facilitates the definition of the interfaces. It is assumed reasonably that each service should have the knowledge of how to cast an object into its actual data type.

Personalised service parameterisation can be static or dynamic depending on when it takes place. When the personalisable parameters of a service are configured at its starting point, this is referred to as static parameterisation. Due to the change of user context or preferences during the service execution time, some service parameters may need to be adjusted to suit the user requirements dynamically. Dynamic parameterisation usually happens in relatively unstable environments (e.g., the resolution of an image may be lowered if it is transferred from a user's computer to his/her mobile phone).

4.3 Role of Personalisation in Service Composition

Service Composition involves combining a set of services together, to give a complete service offering. As discussed in section 2, this is not a novel idea. However, the solution outlined in this paper goes one step further by introducing 'dynamic service composition'. This means that even after the service has been composed, its constituent services continue to be monitored based on the applicability to the overall composition. This is done with the assistance of Personalisation.

Personalisation is first used at the initial stages of composition. A composed service is defined by its 'Service Model'. The Service Model defines a set of service types that are required to make a composed service. Some services are compulsory for the functioning of the composed service and some are optional. The Composer in PSM attempts to retrieve services of the service types defined in the Service Model. For each component service type in the Service Model, there may be many instantiations available in the user's current environment. It is up to Personalisation to select the most appropriate of the services to use for a particular user in a particular context. This means that two different users with different preferences might get different composed services based on the same Service Model. Moreover, the same user in a different setting could potentially get a different composed service based on the same Service Model.

Once the service is selected, the next interaction with Personalisation is to personalise the service itself to the user requirements. This involves adapting the service

itself in some way to suit the user e.g. large font for visually impaired users. The specification of this type of service personalisation is described in section 4.2.

Personalisation continues to be paramount to the dynamic capabilities of the Composer once the service is fully composed. The applicability of a service to the composition might change for reasons such as: the service is no longer available, a service with a higher preference becomes available, QoS available on a service changes, or changes to the service itself (e.g. cost changes based on time). When any of these guards are triggered, then a re-composition will occur transparently to the user.

The PSM, which contains the composition component, uses the Personalisation component to determine the user's preferred service. Personalisation (using the RM and EM) will trigger events to inform PSM when services of higher preference become available. Service preferences can be based on such attributes as price and QoS, and so when any of these attributes change, Personalisation will inform PSM.

In addition, even if these service attribute values remained the same, a user's preference for a service may change. In a mobile world, the user preferences are context sensitive and so change as the user's context changes. These preference changes are passed onto PSM to allow a service re-composition when Personalisation deems a user requirement for a currently running composed service to have changed.

5. Service Composition in Daidalos

Service Composition is part of the Pervasive Service Management (PSM) of the PSP. This PSM consists of five major components that cooperate to provide composite services. These components are: the Priority Processor, Service Discovery, Service Selection, the Service Composition Manager and the Service Actuator. The interactions of these components can be seen in Figure 2.

These components interact as described below to bring together a composed service offering.

1. The Service Composition Manager (SCM) in the PSM is responsible for performing reasoning on the selected composite service. It obtains the detailed service information from the composite service description, which includes the specific requirements for component services.
2. The SCM calls Personalisation to adapt the composition process of the selected composite service.
3. Personalisation refers to the Context Manager and the Rule Manager for user context and preference rules respectively. The composite service is adapted (e.g., adding/removing a component service, setting appropriate starting time for each component service or changing the order of component services) according to the user context/preferences.
4. The SCM then issues the requests for component services to the Service Discovery (SD) component in the PSM.
5. The SD searches the appropriate service directories for all possible services that could be used to meet the user request and returns a set of possible candidate services.

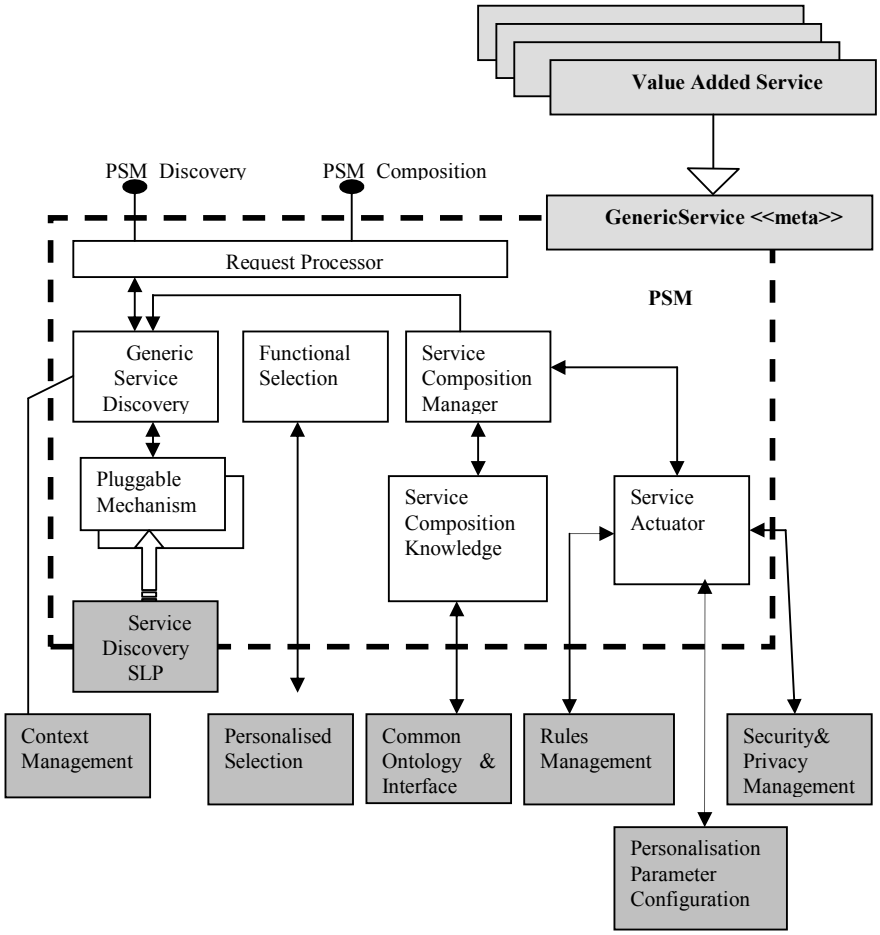

Figure 2 Pervasive Service Management

6. The candidate services are then evaluated and ranked in order based on non-functional preferences. Personalized Selection determines which services are most appropriate to be composed together to make a composite service that best meets the user's preferences.

7. In order to decide the most appropriate services from the ones available, Personalisation refers to the Context Manager and the Rule Manager for user context and preference rules respectively. Service selection is carried out based on the user context/preferences.

8. The chosen service is fed to the Service Composition Manager, which retrieves the "service model" for the service. This service model describes the template for composing the various contributing services of the composite service.

9. The Service Composition Manager looks for component services that match the functional and technical criteria (specified in the service model) of the composite service. This process is repeated until suitable component services have been found.

10. The list of component services making up the composite service then needs to be personalised. This is achieved through adjusting the parameters to these services. The Service Actuator in the PSM calls Personalised Parameterization to configure personalisable parameters of the services.

11. Personalisation refers to the Context Manager and the Rule Manager for user context and preference rules respectively in order to determine appropriate values for the parameters. Service parameters are configured accordingly.

12. Finally, the Service Actuator instantiates service instances if necessary, and returns a handle to the composite service, for use by the service consumer. The Service Actuator is the main component responsible for monitoring constituent services of the composed service during its lifetime. It is the component that adds the *real dynamicity* to the service composition process. A composite service is never in a stable state i.e. it is constantly open to service adaptation. The Service Actuator monitors the service through its lifetime. If specific context guard conditions are violated, the service actuator may request a re-composition of the service, giving an indication of the violating service component.

6 Implementation and Testing

The system has been implemented and demonstrated at two public meetings so far. Three separate scenarios have been used to drive the development and to demonstrate the resulting system.

The first scenario is that of a shared wall display. In this scenario a student is moving around the university looking at some pictures stored in the system. He requests a slideshow service. Initially the only device that is available is his PDA, and the PSP composes a photo service using the PDA display. As he moves he approaches a shared wall display that is free and the system recomposes the service to use this. A further change in context (e.g. professor approaching) triggers a return to the use of the PDA display).

The second scenario is that of a young lady suffering from diabetes who is wearing a blood sugar monitor to warn her if her blood sugar level drops too low. If this occurs, the immediate reaction of the system is to send her a message. However, if no response is received (she has not received the message or is not responding), the system attempts to send a message to other relevant people to help her depending on where she is – her parents or friends (if any are nearby) or in the last event to the hospital service.

These two scenarios are relatively small but demonstrate different aspects of the problems of pervasiveness. The final scenario used to demonstrate the feasibility of the PSP platform is more complex and involves a number of different events and actions. The basic flow of the relevant parts of this scenario is as follows:

Bart is at home watching his newscast on his home monitor. A VoIP call comes in for him from his boss. Based on his current preferences the call is directed to his PDA. The newscast is automatically paused while he is talking to his boss. His boss asks him to go to the airport to pick up a customer. When Bart hangs up the call, the newscast continues where it left off. Bart walks to his car. The newscast is automatically transferred to his car PC. When Bart starts driving, the newscast goes into sound only mode. It runs to the end and finishes. Additionally another service on the car is the Traffic Information Service that takes traffic information off the dvb-t network that services the area. As he reaches the airport, the Information Service is recomposed to become an Airport Information service, and now takes information off the Airport WLAN. When Bart sees the customer's plane has landed, he gets out of the car causing the Airport Information Service to transfer to his PDA. Bart goes to meet the customer.

The set of applications and their component services required for this scenario were a Newscast Service, a VoIP service and an Information Service including the Traffic and Airport specific services. The implementation of these services followed the requirements of the PSP, in order that they could be managed in a pervasive manner by the PSP.

In the scenario, there are examples of context triggers that cause certain behaviour within the system. This was implemented using rules that were triggered when certain contexts occurred, for example *Newscast* will pause when Bart's context is 'BUSY', this context value can be set by any other application, in this case the *VoIP* application. Similar behaviour is visible throughout the scenario.

An example of Personalisation is evident with the *Information Service*, which will display information based on the known native language of the user. In the demonstrated scenario, this was limited to German or English. Of course, in reality this could be extended to many other languages. This ability is made possible by the PSP, and requires very little effort on the part of the application developers themselves. They simply need to provide a method which allows Personalisation to ask what parameters need to be personalised, 'language' in this case. They also need to provide a method to allow Personalisation to tell the application what the value of the parameter should be, in this case either German or English.

PSM's control of Service Discovery and Composition is also evident in the scenario. The most obvious example of composition is when the Information Service recomposes from a Traffic Information Service to an Airport Service. The service itself is composed of an 'InformationGUI' and a 'TransportInformationService'. On the car, PSM composes the BMWInformationGUI with the TrafficInterpreterService. On reaching the airport, a re-composition occurs based on a context trigger. PSM listens for events from the rules triggered and will cause the re-compose accordingly, swapping the TrafficInterpreterService, that was listening to data coming off the dvb-t, for the AirportInterpreterService that listens to data coming off the airport WLAN. PSM also controls the service transfer, which is seen in two places in the scenario; when Bart leaves the house for the car, the newscast follows him to the car, and when the Airport Information Service follows him from the car to the PDA. Again, all this is possible by having the applications provide some simple interface methods that the PSP can then use to make the service pervasive.

This particular scenario was demonstrated on the current prototype at a demonstration in December 2005. In addition to the PSP, the applications and the PSP itself were integrated with lower layer network components, also being developed inside Daidalos. The context triggers were simulated; however, the rest of the demonstration was real. The test site included a BMW with a display monitor, a dvb-t network for the traffic information and several lower level components responsible for such things as QoS, billing, authentication and network handover. This prototype has shown how services can be made pervasive in a simple way by the PSP platform. Additionally, this has been shown on a real network with all the issues associated with a network accounted for. These were real services giving real value in a pervasive manner.

Pervasive Feature	Evidence of Feature in Prototype
Context aware	• Collection of Barts's context attributes • Collection of Applications's context attributes • Behaviour triggered as a result of context
Predictive	• Transfer of Newscast from Home to Car • Pausing of Newscast when busy • Resuming of Newscast when free • Redirection of VoIP call to Bart's current device (boss did not call the device, but just Bart in general) • Re-Composition of Traffic Information to Airport Information • Transfer of Airport Information from Car to PDA
Invisible	• Automatic transfers of Newscast from Home to Car, of Airport Information from Car to PDA • Little required user intervention (though that possibility has not been removed)
Mobile sensitive	• Bart moves about his environment, while the system continuously monitors him and tries to fulfil his requirements with the knowledge it has for example, putting the Newscast in sound-only mode when driving
Continuously adaptive	• Constant context monitoring is apparent. No state of finality is reached in the scene. Bart's services follow him around as he moves through his environment, the adapt their behaviour depending on what Bart is doing for example, talking on the phone or driving.

Figure 3 Table of Pervasive Features Displayed in Prototype of December 2005.

As can be seen in Figure 3, the prototype demonstrated that services built on the PSP demonstrate characteristics that would deem them pervasive.

In this section the progression of Daidalos has been illustrated, in particular the progression of the Context, Personalisation and PSM components. Other aspects of

the platform such as the Privacy and Security component have been ignored as outside the scope of this paper. It has been shown that it is not only possible to create a pervasive environment of services using the PSP, but also that the PSP makes this task an amenable option for Service Providers.

The project continues to address more and more pervasive issues. The first demonstration in 2004 was very simple. In 2005, the prototype exhibited a higher degree of complexity. This project has another three years to develop a platform that addresses increasing amounts of the complex issues of pervasiveness and that will truly allow Service Providers to create pervasive services.

7 Conclusion

This paper has described a dynamic method of service composition that uses the capabilities of personalisation to achieve its requirements in pervasive environments. It argues that this goes one step further than the static service composition that is described in many research papers. Continuous monitoring of the user and their environment enhances the user's experience of services and networks by truly minimising the need to interact with them. The implementation of a prototype has shown that the ideas set forth in this paper are indeed feasible. Although the scenarios are merely a slice of the potential of a pervasive network, they have given us a peek at its potential.

However, there are a number of challenging problems remaining. In the second phase of Daidalos we will be addressing three of these:

(1) Monitoring user behaviour and inferring user preferences. It is essential that the system itself should monitor the user and infer user preferences as far as it is able in order to build up user preferences without requiring too much of the user. Although provision has been made for this in the architecture, development has not yet started on this. The problem is a challenging one and yet essential if the system is to provide a useful service with minimal user intervention.

(2) Handling security and privacy. The issues here are clearly understood but unril now assumptions have been made to enable us to concentrate on the task of dynamic personalised service composition and re-composition. In the second phase the security and privacy issues will be addressed too.

(3) Incorporation of full context awareness. In the first phase context information and triggers, such as location information or location changes, were simulated to verify the ideas. In the second phase we will move to a fully integrated system in which such information and triggers are provided by the underlying infrastructure.

Other interesting issues raised by the research and which need to be addressed include the development of appropriate context-aware user preferences. The first (and simplest scenario) illustrated several of these. How does the system know that when the professor approaches it should switch back to using the PDA? How do we know that the professor's proximity is relevant – he/she may be a few metres away but there is a wall between them? How does the student get information on the professor's location without violating the professor's privacy? And so on.

These ideas will continue to be explored and further developed and prototyped over the next three years as the project progresses in its second phase. Due to the integrated nature of the whole project, these developments will add value to the overall results produced. For example, new more innovative service discovery technologies are to be identified, new methods of inferring preferences and predicting required behaviour are to be investigated. In this phase, we have combined the steam and the pistons to provide forward motion. The next phase will focus on finding out how to put more power in the engine itself.

References

1. Huang, C., Garlan, D., Schmerl, B., Steenkiste, P.: An Architecture for Coordinating Multiple Self-Management Systems. Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture (WICSA-4), Oslo, Norway, June 12-15. (2004)
2. Davy, A.: Task Driven Service Composition for Pervasive Computing Environments. M-Zones White Paper, http://www.m-zones.org. (2004)
3. Huang, A.-C., Steenkiste, P.: Building Self-configuring Services Using Service-Specific Knowledge. The Thirteenth IEEE International Symposium on High-Performance Distributed Computing, Honolulu, Hawaii USA, June 4-6. (2004)
4. Hirschfeld, R., Kawamura, K.: Dynamic Service Adaptation. The 4th International Workshop on Distributed Auto-adaptive and Reconfigurable Systems, Tokyo, Japan, IEEE Computer Society, March 23-26. (2004)
5. Filman, R. E., Friedman, D. P.: Aspect-Oriented Programming is Quantification and Obliviousness. Proceedings of the ECOOP 2001 Workshop on Advanced Separation of Concerns, Budapest, June 17-18. (2001)
6. Araniti, G., De Meo, P., Iera, A., Ursino, D.: Adaptively Controlling the QoS of Multimedia Wireless Applications through User Profiling Techniques, IEEE Journal on Selected Areas in Communications, Vol. 21(10). (2003) 1546-1556
7. Lewis, D., O'Donnell, T., Feeney, K., Brady, A., Wade, V.: Managing User-Centric Adaptive Services for Pervasive Computing. Int. Conf. on Automatic Computing (ICAC'04), New York, May 17-18 (2004) 248-255
8. Wagner, M., Balke, W.-T., Hirschfeld, R., Kellerer, W.: A Roadmap to Advanced Personalization of Mobile Services. In Proceedings of the 10th Int. Conf. on Cooperative Information Systems (CoopIS) Industry Program 2002, Irvine, CA, USA, October 30 – November 1 (2002)
9. Maamar, Z., Mostefaoui, S. K., Mahmoud, Q. H.: Context for Personalized Web Services. Proceedings of the 38th Annual Int. Conf. on System Sciences (HICSS'05), Big Island, Hawaii, January 3-6 (2005)
10. Daidalos. Daidalos EU Framework Programme 6 Integrated Project, http://www.ist-daidalos.org. (2005)
11. Chakraborty, D., Yesha, Y., Joshi, A., "A Distributed Service Composition Protocol for Pervasive Environments", WCNC 2004 - IEEE Wireless Communications and Networking Conference, vol. 5, no. 1, March 2004 pp. 2579-2584
12. Tosik, V., Pagurek, B., Esfandiari, B., Patel, K., "Management of compositions of E- and M- business web services with multiple classes of service", NOMS 2002 – IEEE/IFIP Network Operations and Management Symposium, vol. 8, no. 1, April 2002 pp. 935-938
13. Casati, F., Ilnicki, S., Jin, L. et al (2000) : Adaptive and Dynamic Service Composition in eFlow. HP Labs 2000 Technical Reports, www.hpl.hp.com/techreports/2000/

14. Sheng, Q. Z., Benatallah, B., et al, Enabling Personalized Composition and Adaptive Provisioning of Web Services. The 16th International Conference on Advanced Information Systems Engineering (CAiSE), Riga, Latvia, June 7-11, 2004.
15. IBM. Tivoli Personalized Services Manager, Version 1.2. ftp://ftp.software.ibm.com/software/pervasive/info/tech/tpsm_ss.pdf, 2002.
16. Brar, A. and Kay, J. Privacy and Security in Ubiquitous Personalized Applications. User Modelling Workshop on Privacy-Enhanced Personalization, Edinburgh, UK, 25 July, 2005.