

# Using Workflows to Coordinate Web Services in Pervasive Computing Environments

Anand Ranganathan  
University of Illinois at Urbana-Champaign  
ranganat@uiuc.edu

Scott McFaddin  
IBM T.J. Watson Research Center  
mcfaddin@us.ibm.com

## Abstract

*Pervasive Computing Environments augment physical spaces with a large number of devices and services that help users perform different kinds of tasks. Users in these environments interact with one or more web services using various devices to achieve their goals. One of the problems in these environments is discovering and coordinating different web services for achieving the user's goals. Users may not be aware of which services and devices are available in an unfamiliar environment and how to interact with them in order to achieve their goals. In order to simplify a user's interaction with the environment, we present a novel approach of modeling and managing a user's interaction with the environment based on workflows. We have built a prototype, using the popular business workflow language(BPEL) that models various processes in pervasive environments as workflows. We found that this approach improves the usability of these environments. It also increases flexibility in changing the model of interaction without having to touch individual services and applications. This approach is particularly useful in helping visitors in public spaces like malls, museums, supermarkets and hospitals.*

## 1. Introduction

Pervasive Computing Environments are physical environments saturated with computing and communication, yet gracefully integrated with human users. They feature a large number of devices (sensors, computers, user-interface devices, etc.) and services (such as web services) that help users in performing various kinds of tasks.

One of the uses of pervasive computing is in helping users perform tasks in new environments. Visitors in unfamiliar environments (especially in public spaces like malls, museums, airports, supermarkets and hospitals) may not be familiar with the operation and organization of these spaces and how to perform various kinds of tasks there. In this paper we show how such public spaces can be enhanced with web services and various kinds of devices to help visitors function efficiently in these environments.

Our vision is to allow mobile users to walk into any pervasive computing environment and interact with it to achieve their goals. The environment has a number of web-services performing various kinds of tasks like providing location or map information, giving product or exhibit information, receiving payment, sending advertisements, etc. Most of these web services have a local scope – i.e. they are useful only for users who are in the same environment. Users can interact with web services in the environment using various handheld or wearable devices, using kiosks, touch screens and other devices available in the environment or through speech. Additionally, the interaction can also occur through a multi-modal combination of these mechanisms.

In this paper, we propose the use of business processes and workflows for modeling the way a user should interact with the environment and to coordinate different web services in the achievement of user goals. A workflow breaks down a complex task involving inter-dependent interactions with multiple services into a sequence of simpler subtasks. Each subtask involves a user interacting with a single web service.

A workflow[6] is the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules. Workflows have been used successfully in enterprise settings such as in processing loan applications. We have found that workflows are also a good way of simplifying and customizing a user's interaction with a pervasive computing environment. The use of workflows offers various advantages. It provides flexibility in changing the underlying business process (i.e. the model of the user's interaction with the environment) without having to change the individual services and applications. It also allows integrating and composing different services easily.

We have developed a prototype system that uses workflows to manage how a user interacts with various web services in a pervasive environment. The prototype is based upon a simple interchange protocol between a user and the environment. When the user enters the environment, a task selection step occurs which determines the overall goal of the user ("purchase an item", "return an item", "repairs", etc.). In the next step, additional information is gathered about the user's

preferences (e.g., how does the user like to pay for purchases), and the current context of the environment. The system then generates a customized workflow that describes how various services should interact with one another as well as with the user. For example, if the user is in a pervasive shopping environment and wants to buy a camera, the workflow generated would include steps like guiding the user to the correct aisle (using a store layout web service), allowing him to get more information about the available cameras (using a camera information web service), allowing him to select items to buy (by interacting with a cart web service) and finally receiving a payment from the user (using a payment web service). Finally, the workflow is executed by making calls to the various web services and also choosing the modalities of interaction with the user – services can interact with the user through handheld or wearable devices, using available displays in the store, specialized kiosks, etc. The user, thus, achieves his goal by interacting with various specialized web services and the workflow describes how the interactions should proceed.

A key feature of our system is that the workflow can not only make use of services available in the pervasive environment itself, but also the user's own personal services running on his handheld or other wearable devices. This allows further customization of the user's interaction with the environment.

The use of web services and workflows in commercial public spaces like malls and supermarkets can help enhance customer experience and provide new opportunities for e-commerce and m-commerce. The system can, for instance, provide targeted, location-based advertisements (since it knows what the user is looking for). In Section 2, we describe the basic design elements of workflow systems. Section 3 introduces a reference scenario which uses workflow in a pervasive shopping mall. Section 4 describes our architecture. Section 5 addresses the specific functionality of incorporating on-device services. Sections 6, 7 and 8 describe our experiences and related and future work.

## 2. Workflows in pervasive environments

Our pervasive computing environment consists of a number of web services, each performing specialized tasks. These services could be either part of the environment or brought into the environment by the user on his mobile devices. These services interact with one another and with the user to achieve various goals. A workflow specifies the potential execution order of operations from a collection of web services, the data shared between them, and how exceptions are handled.

Workflows offer many advantages in modeling and executing user interactions with the environment:

1. Workflows are a natural way of coordinating a number of disparate services to achieve a certain

goal. It allows breaking a complex task into a set of simpler subtasks that can be handled easily by individual services. It also allows enforcing ordering constraints between various subtasks

2. It enables structured interaction between the user and the environment. These interactions are guided rather than strictly reactive. It also helps integrate multiple user-interface devices.
3. It provides flexibility in changing the model of the underlying process of interaction without having to alter individual web services. Also, workflows can be easily reused across different environments
4. It improves scalability of application development and execution. This is because flows can be composed; portions of application and user interactions can be developed and tested before being incorporated into a larger production version.
5. Flows may be dynamically generated based on context information and user preferences, allowing customized user-environment interaction
6. Workflows can handle exceptions and faults elegantly. If any segment of the sequence of interactions fails, the workflow can specify ways of undoing previous operations and going back to a legal state from where either another path can be taken to reach the goal or the process can be terminated. This is useful in pervasive environments, which are characterized by uncertainties and faults, but, which may offer multiple ways of reaching the same goal.
7. Various tools exist to model the interaction and execute it – this makes development of pervasive computing scenarios amenable to the same tooling technologies used in enterprise settings.

We describe our workflows using the Business Process Execution Language (BPEL)[7], which is emerging as a standard for describing workflows. BPEL, which uses XML syntax, allows describing complex processes by creating and wiring together different activities like performing Web services invocations, manipulating data and throwing faults. Essentially, BPEL allows us to model the interaction among various services and the user as a state machine or a flowchart. The BPEL script is then executed on an application server to actually perform these interactions.

Each step in a BPEL process is called an activity. Primitive activities include invoking an operation on some Web service (<invoke>), waiting to be invoked by someone externally (<receive>), generating the response of an input/output operation (<reply>), copying data from one place to another (<assign>), indicating that something went wrong (<throw>), etc. These primitive activities can be combined into more complex activities like executing an ordered sequence of steps, branching, looping and executing activities in parallel.

### 3. Scenario

To illustrate how workflows can be used to tailor user interactions with the environment, we describe a scenario. Let us assume that the user enters a pervasive clothing retail store environment. This environment has a number of web services running in it and performing specialized tasks. These include a login service (where users can identify themselves to the environment), a store layout service (which describes the layout of the store), various individual product services (for example, a shirt service that has information about all the shirts in the store), a cart service (where users can add items that they want to buy), a payment service (where users can pay for what they have selected) and so on. The environment has displays acting as kiosks at various places. There is also an indoor location service using 802.11 access points.

The user enters the store carrying a handheld device. His handheld device may also have various services running in it (for example, services that store his personal details and clothing preferences). When he enters the store, his handheld opens up a web page for him where he can indicate what he wants to do in the store. Let us assume that the user wants to buy a shirt. He is then presented, on his handheld, with a list of subtasks that have to be performed for buying a shirt. This list could look like:

1. Go to Shirt Aisle
2. Browse shirts (and, possibly, try them on)
3. Select shirts (by adding to cart)
4. Go to available cashier
5. Pay for shirts
6. Specify packing and delivery options

The above list is the outline of the workflow that was generated for the user to achieve his goal of buying a shirt. Each step in the process makes use of a web service. For example, the first step makes use of the store layout service to generate a route from the user's current location to the shirt aisle. If the user clicks on this step on his handheld, he is shown a map of the store and the route he has to take. If the user clicks on the second step, he is taken to the web interface of the shirt web service where he can get more information about the shirts he sees on the racks (as well as other shirts in the warehouse). The shirt service can get the user's clothing preferences from a personal service running on his handheld and can thus show him appropriate shirts. The user can also be guided to a nearby display which has more real estate for showing shirt related information and he can continue to interact with the shirt service from there. He can try on and then select shirts he wants to buy and his selections are conveyed to the cart service. Since his selections are conveyed to the cart service, he is freed from the burden of carrying his purchases all around the store

before he pays for them. The cart service would get in touch with the store warehouse service that would deliver the selected item at the cashier and the user can pick them up after paying for it.

Once the user has finished selecting items to purchase, he can then make use of the store layout service to get a route to an available cashier on his handheld. Clicking on the 5'th step would allow him to interact with the payment service; and finally clicking on step 6 would allow him to interact with the packing and delivery web service to specify those options.

The workflow process running at the background coordinates the various web services in this activity. Depending on the task, it can allow the user to interact with the web services using either his handheld or other available touch screens. It also includes fault handling procedures. For example, if the payment operation failed, the workflow can undo any previous steps (like returning the shirts to the available racks so that other shoppers can buy them). Finally, the workflow may be more complex than the simple sequence of activities in this example – it could have branching, looping and other complex constructs as well.

This entire workflow process is designed to enhance the user's shopping experience. It tries to augment the physical world (the store, the shirts, etc.) with the virtual world (the handheld, the services, etc.) to make it easier for the user to shop, or more generally, to perform any normal activity.

### 4. Architecture

This section describes the architecture of our prototype system as illustrated in Fig 1. The job of this architecture is to carry out the core steps (task selection, context collection, flow generation and flow execution steps) of the basic interchange protocol with a user, as introduced in section 1. The main elements of this architecture are the task planner service and the BPEL runtime engine, as seen at the right of Fig 1. These elements respectively generate and execute workflows within the pervasive environment. The Task Planner and Runtime Engine are architecturally separated so that it is possible to substitute different logics for the generation and the execution of workflows.

The Task Planner service is responsible for ascertaining the goal of a user and then converting that goal into a workflow. When the user enters the pervasive environment, he interacts with a Task Selection service to specify his goal (Step 1 in Fig.1). The Task Selection service informs the Task Planner service about the goal (Step 2). In the current prototype, the Selection service presents the user with various goal choices in a form on a web-page and the user clicks his goal. Once the goal is determined, the Task Planner gets the appropriate high level workflow template from

a Template Service (Step 3). A workflow template is essentially a full BPEL script from which certain concrete information is missing, such as the particular service against which individual steps are bound. These templates broadly indicate what actions should be performed for achieving different goals. For the retail store example, the Template Service is loaded with templates for goals like buying items, returning items,

ordering items or finding out information about certain items. Table 1 illustrates some of the elements of a template for buying a shirt (involving steps such as guiding the user to the correct aisle, letting him interact with a Shirt Service that provides information about shirts and so on). The left column has BPEL-like fragments of code and the right column explains in brief what the code is trying to do.

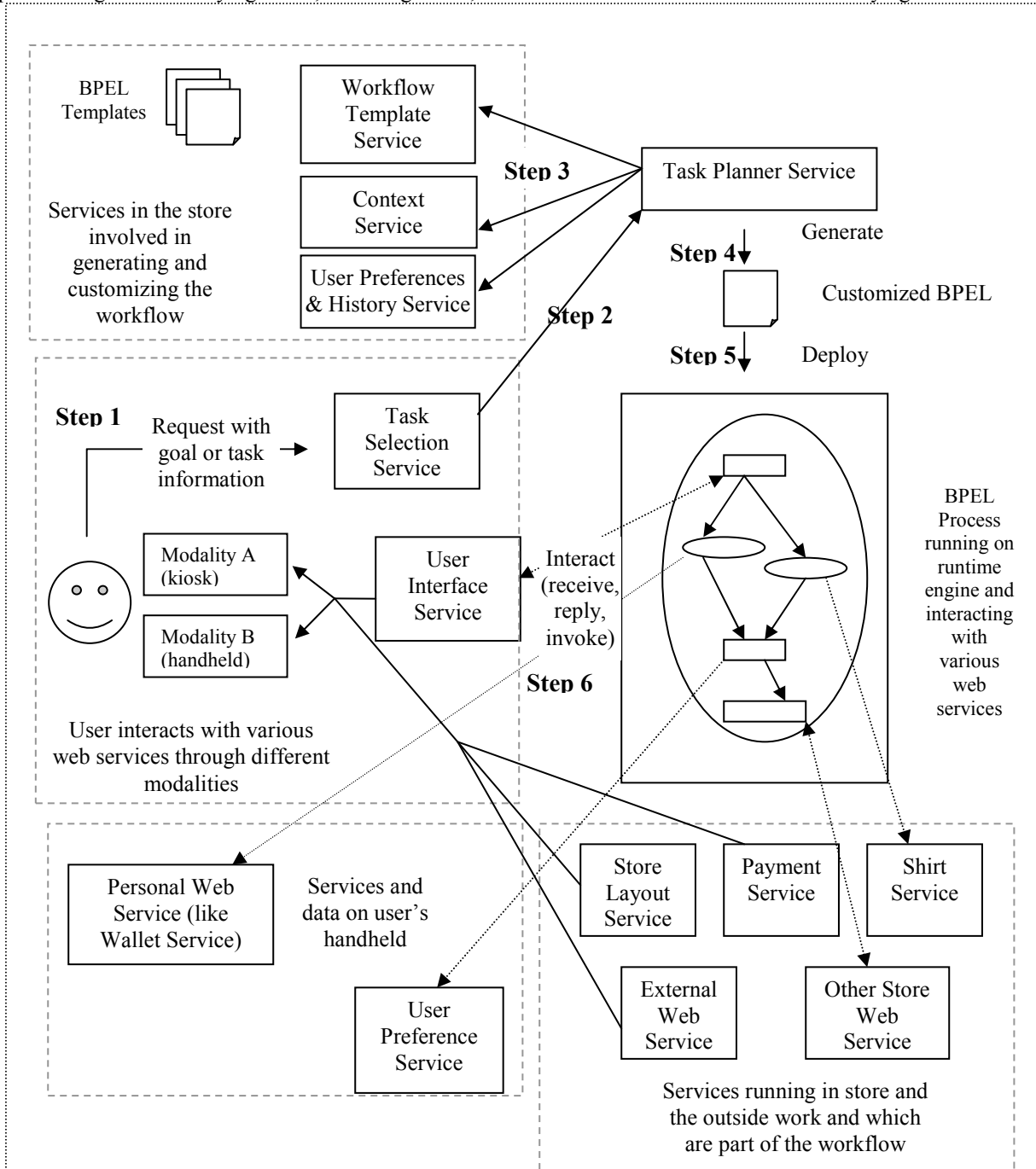


Figure 1 Architecture for workflow-based interaction

**Table 1. Illustration of a Workflow Template as provided by the Template Service**

<pre>&lt;invoke partner="TrackerService" operation="getUserLocation" inputContainer=username outputContainer=userLocation&gt;</pre>	<p>Get user's current location from Tracker Service (by calling the getUserLocation method with the parameter userName)</p>
<pre>&lt;invoke partner="StoreLayoutService" operation="getRoute" inputContainer=routeParams&gt;</pre>	<p>Get route from user's current location to shirt aisle from Store Layout Service and display to user.</p>
<pre>&lt;invoke partner="UserInterfaceService" operation=setLink inputContainer=shirtServiceURL&gt;</pre>	<p>Provide user with reference to Shirt Service to allow him to browse shirts</p>
<pre>&lt;receive partner="ShirtService" operation="setSelections" container="selections"&gt; &lt;invoke partner="CartService" operation="setSelections" inputContainer="selections"&gt;</pre>	<p>After the user had finished selecting the shirts, get selections from Shirt Service and give this information to the Cart Service</p>
<pre>&lt;invoke partner="TrackerService" operation="getUserLocation" inputContainer=username outputContainer=userLocation&gt; &lt;invoke partner="StoreLayoutService" operation="getRoute" inputContainer=routeParams&gt;</pre>	<p>Get route from user's current location (got from Tracker Service) to available cashier (if his cart is not empty)</p>
<pre>&lt;invoke partner="UserInterfaceService" operation=setLink inputContainer=paymentServiceURL&gt;</pre>	<p>Provide user a reference to Payment Service</p>
<pre>&lt;receive partner="PaymentService" container="payment"&gt;</pre>	<p>Wait for confirmation from Payment Service</p>
<pre>&lt;switch &gt;   &lt;case condition= payment=success&gt;     &lt;invoke partner="PackingService" operation="specifyOptions"&gt;     &lt;invoke partner="DeliveryService" operation="specifyOptions"&gt;</pre>	<p>If payment was successful, then allow user to interact with the Packing and Delivery service</p>
<pre>&lt;case condition= payment="failure"&gt;   &lt;invoke partner="cartService"&gt;</pre>	<p>If payment was not successful, then return all items in cart to appropriate services like the Shirt Service.</p>

The next step is the customization of the selected template based on the user's preferences and the surrounding context. The task planner gets the current context from an infrastructure context service (which has information like the number of people in the store, any special events or promotions taking place in the store, and exceptional conditions like certain parts of the store being closed). It gets the user's preferences from either the user's history maintained within the store or from personal services running on the users handheld. From this the BPEL script is further refined according to various rules (Step 4). These rules can help decide exactly which service is to be used in the workflow. They can also add or remove activities or tailor the way certain activities are performed. They essentially modify the XML tree structure of the BPEL template. These rules are currently defined in a proprietary XML format. We are looking at expressing them in a standard format like RuleML[13]. Table 2 has examples of such rules (written in natural language).

Upon completion of the customization and generation of the BPEL, the Task Planner service

deploys the customized BPEL script on an application server (Step 5). This running BPEL script invokes or receives invocations from other web services as per the workflow (Step 6).

Users interact with web services in the environment through web pages generated by them. They may view these web pages either on a handheld or on a kiosk in the environment. For example, the Payment Service mentioned above generates an account selection web page on the handheld or on a kiosk. The web page allows the user to select a mode of payment followed by a confirmation screen against the proposed payment ("Confirm the charge of \$231.66 to credit card number XYZ for the items below...").

Users can also interact with the BPEL workflow itself to specify details about the goals, make choices at decision points in the workflow or get information about the various subtasks involved in achieving the goal. Traditionally, workflows do not have mechanisms for interacting with users. In our system, we let the user interact with the workflow through a specialized User Interface web service. This service generates a web

interface for the workflow. It generates forms whenever the workflow needs some input from the user (such as more details about the goal). It also outlines the main subtasks in the workflow and also indicates at which stage the user is at currently. Additionally each subtask in the workflow also has a link which takes the user to a web page where he can interact with a specialized web service for accomplishing that subtask. The service also decides what device (for example, a handheld, a touch screen in the environment, a speech based interface, etc.) should be used for the user to interact with the workflow or with any web service.

**Table 2: Rules for customizing a BPEL template**

1.	If user is male and big and tall, guide him to the male plus section (by invoking Store Layout Service appropriately). Also replace ShirtService with PlusShirtService
2.	If user likes a particular designer or brand, guide him to the appropriate rack. Also inform Shirt Service about this particular preference so that it can show appropriate shirts to the user.
3.	If user normally pays by credit card or store credit, ask Payment Service to automatically charge him
4.	If store is busy, insert invocations to load-balancing services into various stages of workflow. These services guide the user to the least busy payment aisle or to a trial room with smallest line
5.	If user has allowed it, insert invocation to Advertisement Service to give the user various advertisements or offers at suitable times.

## 5. Incorporation of personal services

One of the key design goals of our system is that users can bring in their own personalized services running on their handheld or other wearable devices and have it be part of their workflow-based interaction with the environment. This allows further customization of their experience with the environment and can also automate various mundane tasks on behalf of the user[12]. For example, a personal details and preference service keeps a list of the preferences of the user, including personal details like height and weight, information like shirt size and shoe size, and clothing preferences. A wallet service maintains a list of payment instruments -- checking accounts, credit cards, digital money -- available to the user much as a physical wallet does the same function using plastic artifacts. If the Task Planner detects the presence of a Wallet Service running on the users handheld, it customizes the script it generates to allow the user to use the Wallet Service for paying as well.

A challenge in incorporating user-provided services is that workflow systems have traditionally been designed in enterprise settings where interfaces and binding information are known at compile-time or

deployment time. In a mobile setting, which is designed to accommodate impromptu relationships with mobile devices, this approach must be extended. In our prototype we have used a battery of existing proxy modules against which the workflow may be statically compiled and bound. The proxies then bind at run time to interfaces and personal services. The discovery of these services involves inspection and cross comparison of service descriptions (using WSDL and WSIL) at run time, when the user presents the device and its services to the environment.

The workflow service and the services on the user's device discover each other through a pluggable discovery framework[15]. This discovery framework factors away the specific discovery protocol that is used in the environment (UDDI, inspection-based[15], multicast-based[14], etc.) and presents a common interface to the services.

## 6. Prototype and Experiences

To experiment with the idea of using workflows to tailor user interaction, we have built a prototype pervasive retail store environment. This environment consists of a number of web services like a Store Layout Service, a Shirt Service, a Cart Service, a Payment Service, a Return Item Service and so on. A user can enter the environment with an IPaq that runs his personal web services (like a Personal Preference Service and a Wallet Service). The user can choose from a number of possible tasks like "Buy Shirt", "Return Item" and so on. Depending on the user's goal, the Task Planner Service generates and deploys a customized BPEL process that guides the user through a sequence of interactions with various Web Services. The user can interact with the User Interface Service and other services either through his handheld or through a kiosk (which is a touch-screen) in the environment. The web services were built using EJBs and they generated customized web pages for interacting with the user using JSPs. The web services and the BPEL runtime engine were deployed on top of IBM's Websphere Application Server[10].

An important aspect of our prototype is that it provides the user both a global as well as a local view of the process he is following. The global view consists of the major steps of the workflow process (like "Go to aisle", "Select Shirt", etc.). This global view has a number of elements. It shows him which steps he has successfully completed, which step he is currently performing and what the future steps are. It also gives him a brief description of the current step and a link to the local view for the step. The BPEL process calls the User Interface Service to describe the contents of the various elements. The User Interface Service then renders the global view to the user in a manner that is

appropriate to the device the user is using. So, it generates different web pages for handhelds and kiosks. Fig 2 shows a screenshot of a global view as a web-page on a kiosk for the process of buying a shirt.

The local view of the process allows the user to perform the current subtask (like selecting a shirt) and is generated by the web service he is currently interacting with (like the Shirt Service). Fig 3 shows a web page generated while the user was interacting with a Shirt Service to search for appropriate shirts.

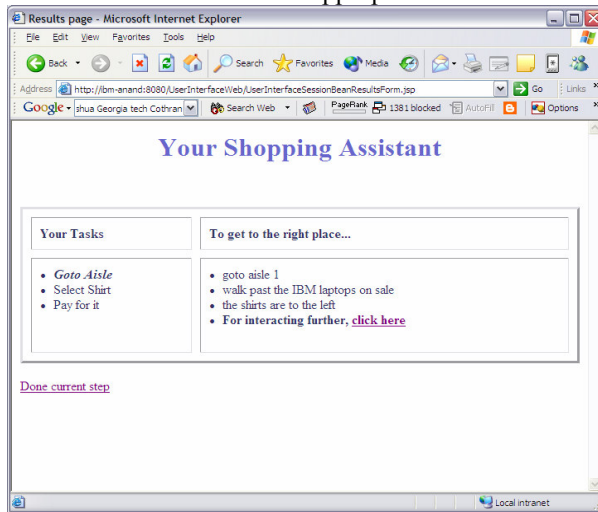


Figure 2 Global view of workflow

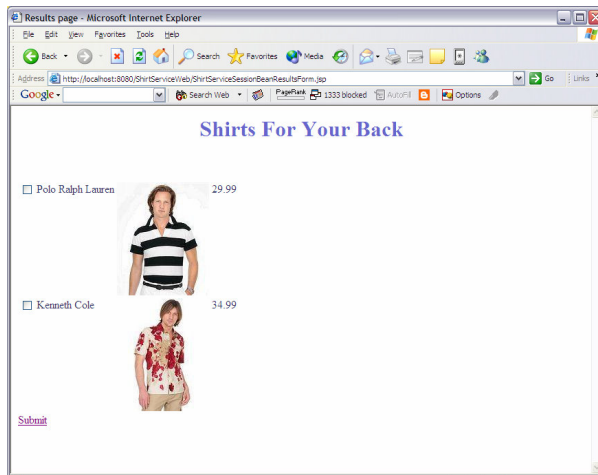


Figure 3 Local view of shirt web service

We imagine that templates of flows as well as the customization rules will be created by a store manager. These templates are in XML. When the flows are to be deployed, one has to specify the references to the web services it calls. This is one of the limitations of BPEL, since it only supports static bindings and references currently. In the case of mobile devices, whose addresses aren't known beforehand, we use proxies.

We believe that workflows do indeed help users in pervasive environments based on the following our observations of users who used our system in the shopping scenario:

1. A single web service helps a user perform a certain task in the physical world
2. A workflow helps the user perform a complex task that involves various subtasks

The first observation followed from the way users interacted with various web services. The Shirt Service, for example, individually, helped users get more information about shirts and also provided an alternative way of browsing the shirts available (in addition to physically browsing through the various racks). This helped the user find and select shirts quicker.

The second observation came from the way that the workflow process helped customize the interaction of the user with individual services as well structure the user's interaction with multiple services. For example, the workflow got the clothing preferences of the user from his personal service on the handheld and provided this information to the Shirt Service so that it can guide the user to the shirts he may want. Also, the workflow process guided users through the various services he had to interact with for buying a shirt.

A key issue is the lack of dynamism of workflows. A workflow is very good for modeling processes which will not change frequently. However, if the user changes his goal midway through a workflow, or if the context of the environment changes, the workflow cannot adapt. A new workflow would have to be generated and deployed to let the user achieve his new goal or take account of the change in context.

Workflows are useful for modeling user interactions when there is a well defined sequence (or more generally, a flowchart) of actions that the user can take to achieve his goals. This is especially true in hospitals, airports and other environments where a user would have to perform various activities in different parts of the building in order to board a plane, get a vaccination, buys a specific item, etc. They do not, however, allow spontaneous interactions, where the user does not have a clearly defined goal and wants to experiment or try different things. Further research is needed to investigate how spontaneity can be allowed by workflows.

## 7. Related Work

There has not been much prior research on the use of workflows for tailoring user interactions in pervasive environments. There has, however, been work done on the composition of web services. [1] describes the use of an augmented Golog interpreter, which is based on situation calculus, to determine a sequence of semantic

web services to be executed for achieving a certain goal. The Sword toolkit[2] uses a rule-based expert engine for determining how to construct a composite web service from existing services. [3] and [5] propose planners that use STRIPS or SHOP2 to compose a plan, given the goal and DAML-S descriptions of a set of basic services. However, all these approaches are targeted towards performing actions on the internet and not towards helping users perform tasks in pervasive environments. They do not allow users to interact with the individual services using one or more devices. They do not have any feedback mechanisms to show users what the plan is, how it is progressing and what stage of the plan is currently being executed. Most of them also do not have fault-handling mechanisms or ways of customizing the plans based on context. Finally, they do not allow new services or sub-plans to be brought in by the user and dynamically incorporated in the plans.

[4] describes a softbot that uses actions schemas describing information-providing and world-altering actions to plan how to achieve a goal on the internet. It, however, is based on actions like ftp and gopher, rather than web service invocations. Also, it does not have ways for users to be involved in the execution of the plan. The SAHARA architecture[11] allows the creation, placement and management of services for composition across independent providers. It, however, does not deal with how users can interact with a sequence of one or more services using various devices to perform a certain task.

## 8. Future Work

As part of future work, we would like to incorporate security into the framework. The services in the store and the user's device must authenticate each other. Also, the user should be assigned roles that constrain the kinds of activities that he (and the services he brings along with him) can perform. We also want to extend our framework to allow a user to simultaneously engage in multiple workflows. This would allow him to achieve disparate goals at the same time. We would also like to allow multiple users to engage in their own workflows at the same time. This would cause coordination problems where different workflow scripts may compete for the same services and devices, and may perform conflicting actions.

Another area of future work is to investigate ways of making the scripts more dynamic and responsive to changes in context and the user's goals. This would involve monitoring the execution of the workflow and generating new plans or workflows when the context or the user's goals changes. We are also investigating the use of planning approaches to determine the sequence of operations needed to be taken for a certain goal.

## 9. Conclusion

We have found that workflows are of great use in guiding users towards their goals in unfamiliar and complex environments. They also help in customizing the user's interaction with the environment by making use of the user's preferences and the current context. A workflow is better than a simple script that calls the various web services since it is inherently based on a flow-chart like model of user interaction and has transaction semantics (including fault-handling and rollback capabilities). Web services and workflows help in, at least partly, achieving one of the goals of pervasive computing – viz. to integrate physical and virtual environments seamlessly.

## 10. References

- [1] S. McIlraith and T. Son. Adapting Golog for Composition of Semantic Web Services. In International Conference on Knowledge Representation '02, Toulouse, France, 2002.
- [2] S. R. Ponnekanti and A. Fox. SWORD: A Developer Toolkit for Web Service Composition. In Proc. of the Eleventh International World Wide Web Conference, Honolulu, HI, 2002.
- [3] M. Sheshagiri et al. A Planner for Composing Services Described in DAML-S. In Workshop on Planning for Web Services, ICAPS, Trento, July 2003
- [4] O. Etzioni and D. Weld. A softbot based interface to the internet. Comm of ACM, July 1994
- [5] J. Hendler et al. Automating DAML-S Web Services Composition using SHOP2. In 2nd International Semantic Web Conference (ISWC2003) Florida, October 2003
- [6] Workflow Management Coalition, The Workflow Reference Model, Document Number TC00-1003
- [7] "Business Process Execution Language for Web Services Version 1.0," BEA, IBM and Microsoft, August 2002: <http://www-106.ibm.com/developerworks/library/ws-bpel/>
- [8] G. Pingali, et al. Steerable Interfaces for Pervasive Computing Spaces In: IEEE International Conference on Pervasive Computing and Communications - PerCom'03. Dallas-Fort Worth, Texas. March 2003
- [9] M. Weiser . The computer for the 21st Century. Scientific American 265(3): 66-75, 1991
- [10] IBM Corporation. IBM WebSphere Application Server. <http://www-4.ibm.com/software/webservers/>
- [11] B. Raman et al. The SAHARA Model for Service Composition across Multiple Providers, Pervasive Computing, August 2002
- [12] S. Berger, et al., Web Services on Mobile Devices – Implementation and Experience, 5<sup>th</sup> IEEE Workshop on Mobile Computing Systems and Applications, Oct. 2003.
- [13] J.K. Lee and M. M. Sohn. The eXtensible Rule Markup Language, Communications of the ACM, Volume 46, Issue 5, pp. 59-64, May 2003
- [14] Multicast DNS, <http://www.multicastdns.org/>
- [15] S. Berger, et al, Towards Pluggable Discovery Frameworks for Mobile and Pervasive Applications. MDM'04, Berkeley, CA, Jan 2004