Physical Hypermedia: a Context-Aware approach

Cecilia Challiol[1], Andrés Fortier[1,2], Gustavo Rossi[1,3], Silvia Gordillo[1,4]

[1] *LIFIA. Facultad de Informática. UNLP. La Plata, Argentina.*
{ceciliac, andres, gustavo, gordillo}@lifia.info.unlp.edu.ar
[2] *DSIC, Universidad Politécnica de Valencia. Valencia, España.*
[3] *CONICET*
[4] *CICPBA*

Abstract. In this paper we describe an original architecture for building and deploying physical hypermedia applications, i.e. those applications in which digital and real world objects are linked together using the well-known navigational metaphor of the World Wide Web. We also explain how our architecture evolved from a substrate for implementing location-based services to a powerful and open basis for supporting different navigation semantics and for building travel assistance services according to the current user's context. After motivating our research and describing the architecture, we illustrate our ideas with some simple examples and compare our research with other related work in the subject.

1 Introduction

Physical Hypermedia (PH) extends the Hypermedia paradigm by allowing navigation through physical and digital objects in a hypermedia network. In PH, physical objects are enhanced with digital information, allowing the mobile user to get that information through a web browser as he explores the real (physical) world [8,9]. In these systems a minimal sensor network is assumed to be set up, so that the system is notified when the user is in front of a digitally enhanced object. As the response to such notification, the system presents the user information about that physical object and a set of links (e.g. in the form of urls) to other objects. These links can be either physical or digital, each one with its own navigation semantics. Digital links behave just like conventional hypermedia links and thus navigation is atomic (i.e., the act of clicking on a link means issuing an http request and navigating to a web page without any intermediate steps). On the other hand, clicking on a physical link expresses the user's intention to physically walk from the current physical object to the link's target. However, there is no guarantee that the user will effectively complete this task, which implies making the decision to navigate and then "walking the link" [10] to the target object. Only when the user arrives to the target, we can say that he has effectively navigated the physical link.

The main problem in PH is that physical navigation is not atomic and it heavily depends on the user's activity and will. He can get lost in his way, he can be distracted by other attractions and make a detour, or even cancel the trip to the

original target and choose a new destiny. The aim of our current research in PH is to find ways to improve the user's experience in navigating a digitally enhanced world [4,15]. In our research we found four major and relevant issues:

- The importance to incorporate a contextual model to adapt the application to the user's needs. As an example, consider that the user is in a digitally enhanced museum and chooses to make a hypermedia trip whose estimated completion time is an hour. If the museum closes in half an hour and the user is not aware of this, he will start the activity and interrupt if due to the closing time of the museum. If the system is aware of the user's preferences and the museum schedule, it can warn him about the situation and present a list of different activities to perform, ranked according to his preferences and the time left.
- The need to offer different (perhaps not hypermedia) services to the user. Physical objects should be enhanced to provide not only information but also services. As an example, consider an art exposition where people can buy any artwork in exhibition. When a person stands in front of an artwork the object might offer the user a service to make a bid. At the end of the day those who made the best bids receive the corresponding notification to end the buying process.
- The importance of changing the role (behavior) of physical objects according to the current user's activity. For example, if the user decides to physically navigate from A to B and he gets lost in his way, some objects should play more the role of active guides than just offer their usual information. While in the "helper guide" role, the object should indicate the user that he is lost and show him how to arrive to the intended target object. In this case the basic physical object's services have been extended with the specific role behaviors. Notice that the same object might play different roles for different users at the same moment.
- The relevance of assuring system's modularity. Even though this should be a rule of thumb for any software development, having a flexible and scalable architecture is crucial in this case, since we expect new enhancements to be delivered to the user as soon as possible. As an example consider adding new context features (e.g. the means of transportation the user is employing for traversing a physical link), adding new role types or behaviors to physical objects, as for example an alert or warning push service when some unexpected event happen (e.g. a road is blocked because of an accident).

In this paper we describe a software architecture and an associated framework for building PH applications. This framework, which is an evolution of our previous research on context-aware systems [5,16], provides a platform to enrich physical objects such that they can present information, physical and digital links, and a collection of services according to the actual role they are playing. The framework supports a scalable context model, allowing to dynamically add and remove relevant contextual features, and to provide personalized user assistance. Finally, it offers an extensible model of roles to be played by physical objects, depending on the user context (such as activity, location, preferences, etc). The framework also supports an important variety of sensing mechanisms and its modularity allows graceful evolution

and improvement of sensing policies. In this paper, however we will ignore sensing issues which can be read in [7].

The two main contributions of the paper are the following:

- We describe a physical hypermedia-aware architecture supporting different kind of navigation and assistance functionality.
- We show how a modular, object-oriented architecture gracefully evolved from supporting location-based services to include (physical) navigational contexts and roles.

The rest of the paper is structured as follows: in Section 2 we present an example of a PH application for tourists in the city of La Plata. In Section 3, we describe our previous work in context-aware services. In Section 4, we present the architecture to develop PH applications. Finally in Section 5 we compare our work with other research projects in the field and in Section 6 we discuss some further work we are pursuing.

2 Motivation: A Physical Hypermedia Example

Through the rest of the paper we will use an example of a PH tourist guide for the city of La Plata, to illustrate the kind of problems we need to solve. Similar to a conventional hypermedia the user navigates through a network of nodes (using a Web browser), which comprise information on points of interest (buildings, squares, museums, etc) and related material (history, traditions, etc). Nodes are connected with links, which reflect meaningful relationships between nodes. There are two main differences between conventional and physical hypermedia; first, some of the nodes have their physical counterpart in the real world (e.g. a building), and they are open when the user is in front of the physical object; besides, some links "point" to physical objects and navigation is intended to be physical, meaning that the user must move to the target to get it opened. We next concentrate on the behavior of physical objects and physical links.

Let us suppose that the user is standing in front of the Natural Sciences Museum in La Plata, and as a consequence the browser shows information about the museum (history, opening hours, etc), and a collection of digital and physical links. Some physical links may refer to other related buildings (e.g. which were built in the same period) such as La Plata's Cathedral. When the user clicks on this physical link, he gets as a result a map describing the itinerary from his current position to the target's location. Figure 1 shows an example of this behavior.

Once the map is presented, the user can choose to go to the Cathedral or cancel the requested itinerary. If the user follows the recommended path, at some point he will walk by the Theater, which is another point of interest. Since the user is walking in the correct way (from the point of view of the activity he started when he clicked the physical link), the Theater should take the role of an *information guide*; playing this role, the Theater informs the user that he is in the right way and gives information about the estimated time of arrival to his destiny. Also some of the Theater services (selling tickets for example) are not displayed, since that would distract the user from his current activity. Finally, the Theater could even present information related to the

Cathedral, since that is the target of the user's activity (for example, how many tickets are left to access the cathedral elevator to see a panoramic view of the city). Figure 2.a depicts this situation.

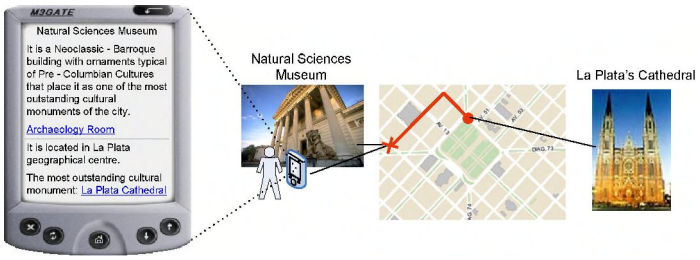

Fig. 1. As a result of clicking on a physical link, the system presents a map showing the path that the user should follow to reach his destiny.

At a certain moment, the user may get lost and walk in other direction, without following the planned route. When he walks by another point of interest (for example, the Government House) he will receive information based on the actual role played by the building. Since the user is not in the right way, the Government House could take the *helper guide* role and advise the user that he is walking in the wrong way with respect to his intended task, (i.e. to reach the cathedral). It also gives the user information about how to go back to his intended path. This situation is depicted in Figure 2.b.

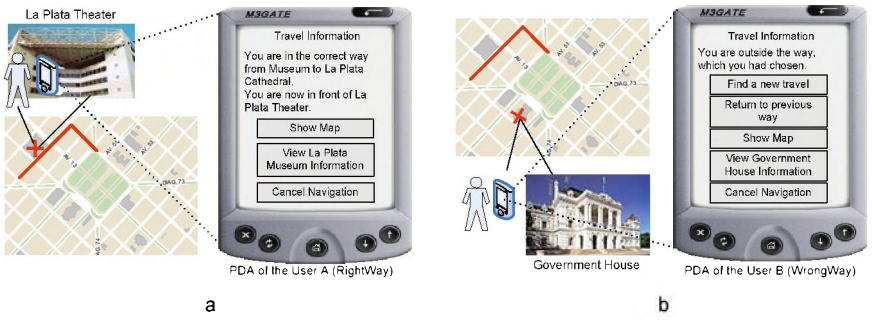

Fig. 2. Examples of physical object's role according to the user's activity and current location.

In case the user in Figure 2.b follows the instructions presented while being in front of the Government House, he will end up reaching the Cathedral. At this moment the user finishes his current physical navigation, because he has arrived to the final target and gets as a result a Web page with information about the Cathedral, similar to the one presented in Figure 1. Alternatively, the user could have made

another choice when he found out that he was outside the intended path and asked the system to cancel his current navigation.

Although we have been describing physical objects which are evident components of a tourism hypermedia network, there are also other types of useful physical objects which can help to guide the user. These objects are usually found in the physical space the user is traversing (like traffic lights, public phones, ATMs, corners, etc.) but are not considered to be points of interest from the tourist point of view. These objects may also provide physical information and physical links.

The same ideas presented before can be also used in an indoor setting. For example, if the user decides to visit the museum before going to the Cathedral, he would find "live" geological objects displaying their characteristics, exhibiting links to digital information or recommending a visit to another part of the museum where the user can find other objects of interest (for example, in the paleontology room) of the same geological era. A well designed system should make the outdoor-indoor transition seamless to the user, keeping the same paradigm and form of interaction. In the following section we briefly describe our context aware framework and next we show its evolution to support the previously described PH features.

3 The Underlying Framework

The PH framework is a step forward in our previous research work [5,16] aimed at providing support for deploying dynamically varying services according to the user's context. The framework has been generalized in order to accommodate different customization hot spots, in particular the ones required by physical hypermedia applications.

We will first focus on how to provide location-based services for a tourist system similar to the one presented in the previous section (but without hypermedia features). For this purpose we assume that there is an underlying software which already provides the basic information about the city and the relevant points of interest (like museums, churches and so on); this system is used for example by potential visitors. Our goal is to enhance the system so that it can be accessed by means of a mobile device (like a PDA or a Smartphone), and to allow information and services to be presented according to the user's context. Initially, the system shows a set of services, which are present all the time (for example, a map showing the city and the user's current position) while others are context aware (for example a list of nearby restaurants is enabled at lunch time, ranked by distance and user's food taste). The system can also be enhanced by taking into an account the context of other objects, such as the activity in a given building. Suppose that the user whishes to visit the city town hall, which is closed due to unscheduled maintenance operations. If the system can be aware of the town hall context, the user could have been warned about it or the trip planned in a different way.

Due to space constraints we will only explain the architecture's main components and the process of configuring the framework to support location-based services. The most important abstractions we defined are the following:

Aware Object: An aware object is a standard application object (e.g. a building, person, etc) that has been enhanced with the ability to keep track of the context features that are relevant for it at a given moment in time. In the tourist example, the user itself is represented by an aware object which manages relevant context features, like his location or his food preferences. Other example of aware object could be a building which manages context features like its state (closed, opened, etc). An aware object is an observer [6] of every context feature he owns, so that any change in its context can be captured and processed.

Context Feature: Aware objects rely on a collection of *context features* which, as their name suggest, model the relevant context. A context feature holds its current "value", which is usually a high level view of the information gathered by sensors (either hardware sensors such as a GPS or software sensors as will be shown later). Following the tour example, the aware object that represents the user will have a location feature which holds his position in a given location system [13] (the connection between sensing devices and context features is out of the scope of this paper; the interested reader can consult our previous work in this field [5,7,16]). In case the user is moving outdoors and has a GPS in his PDA, each time a new location is gathered the location feature gets its current situation updated and a context-change event is triggered so that it can be captured by the aware object.

Adaptation Environment: Our framework provides a set of hot-spots to use it as a platform that provides different (customized) behavior responses. One of these hot-spot is the *AdaptationEnvironment* class, used to create different types of adaptation mechanisms. In the guided tour case, a *ServicesEnvironment* is created by sub-classing *AdaptationEnvironment* and adding the required behavior to keep track of available services. If we want to provide groupware services (e.g. a messenger whose contact list are the users present in a room), we should define a *GroupwareServicesEnvironment* that would be in charge of setting network connections and discovering available devices. The aware object should be registered in one or more environments depending on the adaptation it needs. In the guided tour case, the aware object which represents the user should be registered in the *ServicesEnvironment.*

Event Handlers: When a context feature changes, it triggers a context-change event, which is captured by the aware object. As a response to this event, the aware object forwards it to all the environments to which he is registered, so that they can perform their specific behaviors. An event handler that is registered in an environment will receive a message each time a context event is fired. An event handler may specify to be called only when a certain aspect triggers the event (e.g. location change) and/or when it is fired by a specific aware object (this is useful when many aware objects share a common environment, like a groupware one). By decoupling the context model from the possible customization based on context changes, adding new capabilities to the application is straightforward. In the guided tour example, a services event handler should be registered to the *ServicesEnvironment* to determine if new services should be added or active ones removed, according to the user location.

In Figure 3 we show a simplified class diagram of the framework. In Figure 4 we show a class diagram with the service classes. The basic framework classes that are extended appear grayed out.

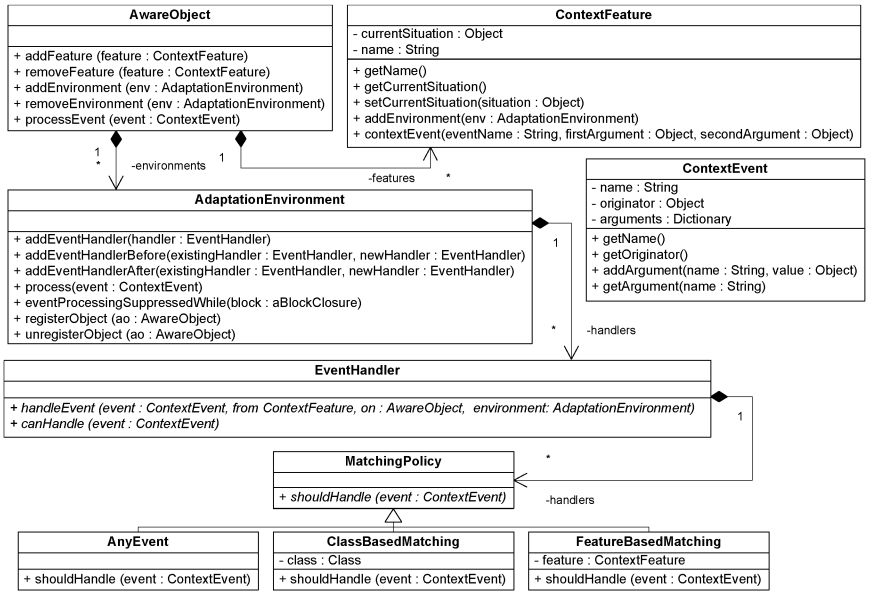

Fig. 3. A simplified class diagram of the framework.

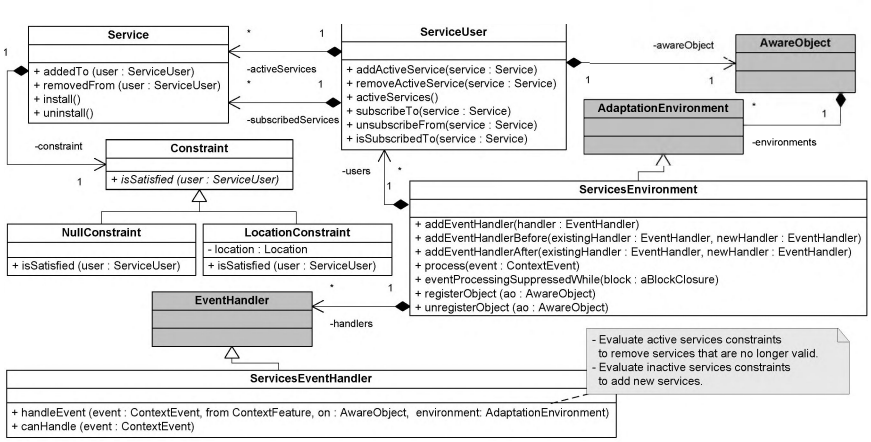

Fig. 4. A class diagram showing the extensions needed to achieve location-based services.

4 Supporting Physical Hypermedia Features

In this section we show how to use the described framework to instantiate PH applications like the one described in Section 2. Since PH applications usually require handling "fine grained" physical objects, we assume that points of interest are tagged with beacons, which can be Bluetooth [11] or Infrared [17]. With this configuration, when the user stands near the physical object his device will receive a signal that identifies that particular place and thus let the system know where he is standing. Though similar behaviors can be achieved by using GPS or a combination of both GPS and beacons, the tradeoffs of using these different location-sensing devices are outside the scope of this paper.

In order to support the PH concepts, we modified an existing PDA-based web browser to extend nodes and links semantics. To explain its behavior, in Figure 5 we show the typical information displayed when the user is standing in front of a point of interest. The web page contains five main categories:

- Digital information, explaining the place's features.
- Digital links, which are standard hypermedia links.
- Physical information, about the point of interest location.
- Physical links, which can be used to start a walking activity.
- Services provided by the point of interest.

Fig. 5. The standard information web page, divided in five categories: digital information, digital links, physical information, physical links and services.

When the user clicks on a digital link, only the digital information is changed, showing the contents of the Web page requested by the user, but the physical links and services remain the same. In other words, even though the user is digitally navigating if he is still standing in front of the same physical object, physical information should be kept unchanged. On the other hand, when the user clicks on a physical link he indicates that he wants to reach a physical location; therefore a map (or other multimedia representation), showing the path he should follow, must appear; only when the user reaches his destination the complete information about that place (digital and physical information, physical and digital links and services) are shown.

Since our architecture adheres to the MVC [12] paradigm, the modified web browser is embedded in the final application just as any other view would be. As we

show in the following sub-section, the browser is an aware object which acts as a view of one specific context feature that models the navigation history. When the user clicks a links, the context feature is changed and as a result the web page is updated. Notice that in the MVC paradigm, views are aware objects with respect to models.

4.1 Main Components

In this section we show how we derived the basic functionality of a physical hypermedia application using the framework components. As in the tourist example, the user is represented by an aware object which "observes" a collection of context features that are relevant to configure the application behavior. These context features are:

- **Location**: Indicates the physical object in front of the user. In case the user is not in front of a physical object (for example, while walking a physical link), his location is represented as *anywhere*. Locations are not "just" strings or latitude/longitude pairs, but full fledged objects. Objects may be hypermedia nodes (i.e. physical objects that have their digital counterparts, with both physical and digital links) or plain physical objects that can only have physical links. When we later show how to add role support to the PH application, we will demonstrate that the location feature will hold a physical object wrapped with the role it is playing.
- **User Activity**: It indicates the activity that the user is actually performing, in terms of the PH application. The most important activities related to the example are *in front of* a point of interest, walking a link *in the right way* or walking a link *in the wrong way*. The framework is open-ended regarding activities, but for the sake of comprehension we emphasize the previously mentioned one. It is important to notice that, when using beacon-based sensing, we only know the precise user's location when he is in front of a physical object (either a node in the hypermedia or other physical object like a traffic light). In case we use a GPS this would still hold, but we could additionally have information about the user's position in latitude/ longitude coordinates, and therefore we could be able to show him in real time the path he is following.
- **Navigation**: Keeps track of the current url the user is visualizing, and of all the previous links the user has requested (either physical or digital). When the user clicks on a link, the url requested by the user is set as the feature's current situation. Also, when the user senses a new physical object, the navigation feature situation is set to a url describing the physical object's information. Since the web browser is set as an observer of this context feature, an http request will be triggered and the display will be updated.
- **Digital Navigation**: Keeps track of the current digital object and of all the previous digital objects the user has visited. Notice that the user can digitally navigate while standing in the same physical location, thus the digital navigation feature can change its current situation while the location feature remains unchanged.

Figure 6 depicts the relationship (according to the MVC paradigm) between the views (the main GUI and the enhanced web browser) and their corresponding models (the aware object and the navigation context feature respectively). The browser is a component of the view, which acts as an Observer [6] of the navigation context feature. When the navigation feature changes, the browser receives a notification and generates an http request with the url provided by the current situation of the navigation context feature. The browser therefore is only updated when it receives a notification that the navigation feature has changed.

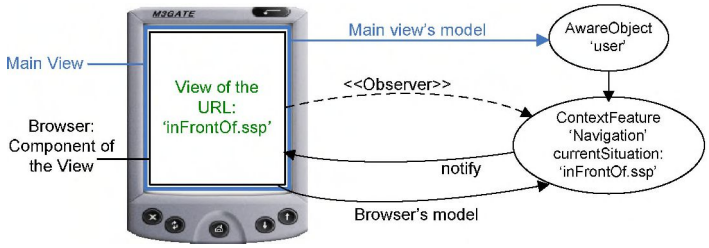

Fig. 6. The enhanced web browser as a view of an aware object.

When the user's device senses a physical object's signal, it changes his state to the *in front of* activity, displaying the objects information. Unless explicitly specified by the user when clicking a physical link, he is supposed to stay in the same activity. The information displayed in his browser is updated as he walks by different physical objects. In case the user clicks a physical link the system is optimistic and assumes that he will be walking the right way unless a point of interest out of his path is found. Once the user reaches his destination, his activity goes back to *in front of*. In order to implement this behavior, activities are modeled as first class objects whose responsibility is to process transitions by returning the new activity the user should switch to. Activities are modeled as a variant of the State [6] pattern.

As a final remark, it should be noticed that all transitions except two are triggered by changes in the user's location. The first exception is the transition from the *in front of* activity to *walking right way*, which is triggered when the user clicks a physical link. The second one can happen while the user is walking and he explicitly cancels his trip. As we will see in the next section, this is important to determine which handler should perform this transition.

To provide navigation assistance we designed a role model inspired in [18] and which is intended to let physical objects exhibit different services according to the user's state, as explained in Section 2. In our architecture, each role is modeled by a class, which acts as a wrapper of the physical object to which it is assigned. This class is used to modify or extend (depending on the situation) the basic services that the user receives when he stands in front of the physical object. To do so, a group of services are associated with each role; these services can be generated dynamically according to the needs of the user in that moment and therefore the services provided

by a given role can vary according to the user's context. As an example we detail some possible roles a physical object may play:

- **Navigation Guide**: this role is relevant when the user has chosen to walk a physical link. Once he has started to walk, he can be either walking in the right way or in the wrong way according to the presented path. Therefore, we have two different roles that represent this situation:
    - **Information guide**, which represents the role to be played by a physical object if the user is in the correct way, giving the user confidence that he is in the path to reach his target.
    - **Helper guide**, which is the role which corresponds to the situation in which the user is outside of the way, warning him about this and providing a set of alternative services to resume his trip. If the user wants to return to his previous path, the system offers a service showing a map to do so (*Return to previous path* service). On the other hand, the system can find a new path from the user's position to his original destiny (*find a new path* service).

  Notice that both roles could have common services such as *Show Map* to provide a map with the user's location, *Cancel Navigation* to stop the current physical link navigation and so on. These are shared services since they are related with the activity of walking a link itself and are independent of whether the user is in the correct way or not. Another shared service is *View Information*, which provides digital links and information to the object that is being sensed by the user.
- **Query guide**, which can be used to perform searches in the PH network, for example searching for nearby points of interest, finding paths and so on. The service provided by this role is the analog of performing a search inside a web site, but adapted to the PH paradigm.
- **Alert guide**, which is used to alert the user about critical events. For example, the user can receive a notification about a fire taking place a few blocks from his position. Notice that in this case we expect the alert guide to suppress any other irrelevant service that may disturb the user, so this role not only adds services to the wrapped object but also inhibits others.
- **Attention guide**, which is used to capture the user's attention to inform him about interesting things. For example, this guide can let the user know that in that day the access to museum is free. Notice that this role is different from the previous one in two main aspects: it doesn't inhibit other services and it is used for non-critical notifications.

4.2 Handling Context Changes

Once the context features have been defined, we need to design the different handlers to respond to changes in the user's context. For the sake of clarity we decided to use a naming convention for handlers with the form <DependentContextAspect>-<ContextAspectToChange>. As an example, a Location-Activity handler is triggered when there is a change in the location context feature and acts on the activity feature.

Location-Activity Handler: this handler is registered to be triggered when the location feature changes. As a response, the user's activity is re-evaluated according to his new location. As explained in the previous section this is achieved by asking the current activity to process the new location, which returns the new user activity.

Navigation-Activity Handler: when the user clicks a link in the browser, the url is set as the current situation for the navigation feature. As a response to this change, the navigation-activity handler is triggered. In case the selected url is a physical link (i.e., the user asks for physical navigation), the user's activity is simply set to walking a link in the right way and the *travel* variable corresponding to this activity is set with the itinerary from his current position to his destiny (the target of the link). This itinerary is showed when the browser is updated (as a response of the fact that the navigation feature has changed). On the other hand, if the selected link is a digital one, the digital navigation feature is updated with the digital object representing the target of the link. The browser is updated (because the navigation feature has changed) and it shows the information of the digital object that has been stored in the digital navigation feature.

Location-Navigation: The user's location has changed (e.g. new data has been gathered from a sensor). Depending on the user's current and previous activity, the url of the digital navigation feature is updated according to the following rules:

- If the user is in front of an object, the url is set to *inFrontOf.ssp*[1], that displays the object's information as was shown in Figure 5.
- If the user is walking (either in the right or wrong way) and has reached his target, the url is set to *inFrontOfTarget.ssp*.
- If the user is walking (either in the right or wrong way) and has passed by a point of interest that is not the target, the url is set to *passedBy.ssp*.

It should be noticed that, as a result of the process made by this handler, the navigation feature is changed, which means that the browser will be updated.

Once the basic handlers have been defined, we can proceed to handle context changes in order to assign a role to the current physical object in front of the user. This role is assigned to the physical objects dynamically, according to the user's context (so far we only take into an account the user activity, but we plan to extend this to other context aspects in the future). When a physical object has been sensed, its role is determined to better suit the user's needs. Since the role is assigned when the user changes his position, we created a new handler named **Location-Role,** which is registered to be triggered when the location feature changes (since the object's role depends on the users activity, this handler is registered to be triggered after the Location-Activity handler processes the location change event). As a response to a change in the user's location the control is passed to an instance of the *RoleBuilder* class which acts as a Facade [6]. The *RoleBuilder* performs a double-dispatching between the user and his current activity in order to determine the physical object's role. In Figure 7 we show an interaction diagram depicting the collaborations between the handler, the role builder and the activity in order to get the physical object's role.

Notice that the role assignment process starts as a response to a change in the current location (i.e. the physical object that is in front of the user), and ends up adding a role to the physical object. As a result, the location context feature will now

[1] *ssp* stands for Smalltalk Server Pages, the Smalltalk counterpart of *jsp* pages

hold a role (which wraps the physical object), indicating that the user is standing in front of a physical object, that has been reshaped to adapt its behavior to the user needs.

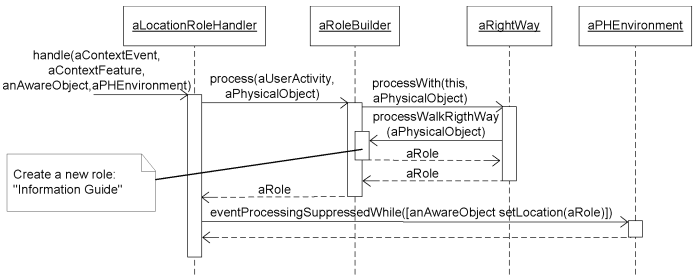

Fig. 7 An interaction diagram depicting the creation of a role based on the user's activity.

5 Related Work

The term Physical Hypermedia was coined in [8]. In [9], the authors present a comprehensive framework (HyCon) whose goal is to extend the Hypermedia paradigm with the manipulation of real world objects. This framework supports not only context-aware physical navigation but also allows users to create their own linking structures and trails. Another related approach is the HyperReal framework [14], an object-oriented framework which allows mixing the real and the digital world. Finally, Harper et al [10] extended the metaphor of links with the idea of "walking" a link as they augment Hypertext with physical relationships present in the real world.

Our research has been certainly inspired by these seminal research projects; we further characterize real-world objects according to the role they can play to assist the user's travel. From an architectural point of view, we chose not to build a software substrate from scratch but to extend a service-oriented architecture with hypermedia services; in this way we can integrate hypermedia behaviors with other context-aware services not directly related with navigation. We do so by composing relevant context features, which describe the user's situation and a set of handlers to respond to changes in the context model.

We found our model of context to be quite similar to the one presented in [3]. While in most approaches, context is viewed as a collection of data that must be specified at design time and whose structure is supposed to remain unaltered during the lifetime of the application, [3] proposes a phenomenological view of context. In this approach, context is considered as an emergent of the relationships and interactions of the entities involved in a given situation. In our approach, the intended behavior is the combination of the context features (which together make up the whole context) and of the interaction between the aware-objects, the context features

and the handlers. In addition, we do not assume a pre-defined context shape, and allow for run-time changes on the context model.

From an architectural point of view, our work has been inspired in [2]: the sum of our micro-architectural decisions (such as using dependencies or decorators) also generates a flexible, evolvable architecture. Finally, we adapted part of the ideas in [18] to define the set of roles which physical objects may play in the context of the user's journey.

6 Concluding Remarks and Further Work

We have described an architecture for building context-aware physical hypermedia applications; the architecture and its associated software framework supports different kinds of browsing behaviors. In particular, it allows building navigation assistance by providing a model for assigning behavioral roles to physical objects according to the user's navigation activity. One of the main strengths of our approach is that it allows incremental development of this kind of software systems; for example new services or object's roles can be added dynamically, therefore supporting Abowd's view on ubiquitous computing [1]. The framework and the PH prototype were developed in VisualWorks, a Smalltalk-based environment, which is cross-platform and can be executed in PDAs. Although we used a web browser written entirely in Smalltalk, any other browser that supports a callback facility to intercept link selection can be used.

We are now researching in several areas: first we are devising a finer grained strategy for assigning roles to physical objects, taking into an account several context features, such as the user's history or his preferences. To improve the construction of PH applications we are also developing a user friendly tool. We are studying how other metaphors of hypermedia navigation can be mapped to PH (e.g. what is the counterpart of the *back* button for a physical link). Finally, we are developing a set of widgets to generate a better GUI; our main concern is how to present a large amount of services provided by a physical object in small displays such as the ones found in mobile devices. In this regard, our hypotheses is that zoomable interfaces are well suited for this situation, since they allow to structure a large amount of items in a small area, with a smooth interaction and a bigger nesting depth that can be applied in the same situation with standard menus.

7 References

1. Abowd, G. D.: Software Engineering Issues for Ubiquitous Computing. Proc. 21st Int'l Conf. Software Engineering, ACM Press, 1999, pp. 75-84.
2. Beck, K., Johnson, R. E.: Patterns Generate Architectures. ECOOP 1994: 139-149.
3. Dourish, P.: What we talk about when we talk about context. Personal and Ubiquitous Computing 8(1): 19-30 (2004).
4. Challiol, C., Gordillo, S., Rossi, G., Laurini, R.: Designing Pervasive Services for Physical Hypermedia Applications. Proceedings of the IEEE International Conference on Pervasive Services. Lyon, June 2006.

5. Fortier, A., Rossi G., Gordillo, G.: Decoupling design concerns in location-aware services. In Mobile Information Systems II, pages 187–202, 2005.
6. Gamma, E., Helm, R., Johnson, J., Vlissides, J.: Design Patterns. Elements of reusable object-oriented software, Addison Wesley 1995.
7. Grigera, J., Fortier, A., Rossi, G., Gordillo, S.: A Modular Architecture for Context Sensing. To be presented in "The Second IEEE International Symposium on Pervasive Computing and Ad Hoc Communications (PCAC-07)", Niagara Falls, Canada, May 21-23, 2007
8. Gronbaek, K., Kristensen, J., Eriksen, M.: Physical Hypermedia: Organizing Collections of Mixed Physical and Digital Material. Proceedings of the 14th. ACM International Conference of Hypertext and Hypermedia (Hypertext 2003), ACM Press, 10-19.
9. Hansen, F., Bouvin, N., Christensen, B., Gronbaek, K., Pedersen, T., Gagach, J.: Integrating the Web and the World: Contextual Trails on the Move. Proceedings of the 15th. ACM International Conference of Hypertext and Hypermedia (Hypertext 2004), ACM Press. 2004.
10. Harper, S., Goble, C., Pettitt, S.: proximity: Walking the Link. In Journal of Digital Information, Volume 5, Issue 1, Article No 236, 2004-04-07. http//jodi.ecs.soton.ac.uk/Articles/v05/i01/Harper/.
11. Huang, A., Rudolph, L.: A privacy conscious bluetooth infrastructure for location aware computing,.http://people.csail.mit.edu/albert/pubs/2004-albert-infrastructure-forlocation-aware-computing.pdf.
12. Krasner, G., Pope, S.: A Cookbook for Using Model-View-Controller User Interface Paradigm in Smalltalk-80, Journal of Object Oriented Programming, August/ September, 1988, 26-49.
13. Leonhardt, U.: Supporting Location-Awareness in Open Distributed Systems. Ph.D. Thesis, Dept. of Computing, Imperial College London, May 1998.
14. Romero, L., Correia, N.: HyperReal: A Hypermedia model for Mixed Reality. Proceedings of the 14th ACM International Conference of Hypertext and Hypermedia (Hypertext 2003), ACM Press, 2-9.
15. Rossi, G., Gordillo, S., Challiol, C., Fortier, A.: Context-Aware Services for Physical Hypermedia Applications. OTM Workshops (2) 2006: 1914-1923
16. Rossi, G., Gordillo, S., Fortier, A.: Seamless Engineering of Location-Aware Services. Proceedings of CAMS 2005, 2nd Workshop on Context-Aware and Mobile Services, Ciprus, October 2005, Springer Verlag.
17. Want, R., Hopper, A., Falcao, V., Gibbon, J.: The Active Badge Location System. ACM Trans. Information Systems, Jan. 1992, pp. 91-102.
18. Yesilada, Y., Stevens R., Goble, C.: A foundation for tool based mobility support for visually impaired web users. In Proceedings of the Twelfth International Conference on World Wide Web, pages 422–430, 2003.