Adaptive User Interfaces for Smart Environments with the Support of Model-Based Languages

Sara Bongartz[1], Yucheng Jin[1], Fabio Paternò[2], Joerg Rett[1], Carmen Santoro[2], and Lucio Davide Spano[2]

[1] SAP AG, Darmstadt, Germany
{sara.bongartz,yucheng.jin,joerg.rett}@sap.com
[2] CNR-ISTI, Pisa, Italy
{fabio.paterno,carmen.santoro,lucio.davide.spano}@isti.cnr.it

Abstract. This article presents a solution for supporting adaptive user interfaces in work environments. Its architecture is built upon the concept of model-based UI design extended by context aware and adaptive features. Model-based languages provide the software development process with useful support for, building design prototypes and actual implementations for devices with various interaction resources. The proposed architecture is able to adapt to selected aspects of the context during run-time by communicating with a context server and applying the specified adaptation rules. In order to show the possibilities of the proposed solution, we report on its application in the development of an adaptive user interface prototype to be used in a warehouse picking system.

Keywords: Adaptive service front-ends, Context-aware user interfaces, model-based user interface languages, Warehouse picking system.

1 Introduction

Work environments are continuously becoming richer in sensors and devices and it is important that users are efficient and perform well in such contexts, without having to spend too long in understanding how to interact with the system. Adaptive interfaces can be useful for this, because they can provide the required information in the most suitable modality by taking into account the current context of use. For this purpose, it is useful to consider various contextual aspects, including the user-related aspects (tasks to accomplish, personal preferences and knowledge, etc.), the aspects related to the technology (available interaction resources, connectivity support, etc.) and the environmental aspects (level of noise, light, etc.).

In order to show the possible application of our methods, tools and languages; we consider a specific application domain: warehouse picking, which is a part of a logistics process often found in retail and manufacturing industries. Warehouses store the goods and products incoming from suppliers until they are collected and shipped to the stores or customers. The process of picking items from a shelf, collecting them in some sort of container and bringing them to certain locations is usually conducted by

F. Paternò et al. (Eds.): AmI 2012, LNCS 7683, pp. 33–48, 2012.

one or more persons. This leaves a maximum degree of freedom for the enterprise to change or rearrange the environment when needed. The costs of maintenance and the down-times are lower compared to a fully automated system based on e.g. conveyor belts. In addition, only people can react and adapt to unforeseen situations.

Since the beginning of warehouse picking, people have thought on how the pickers can be supported in their task. Technological solutions have been provided, like Voice-directed warehousing (VDW). VDW refers to the use of the voice direction and speech recognition software in warehouses and distribution centres. In a voice directed warehouse, workers wear a headset connected to a small wearable computer, which tells the worker where to go and what to do using verbal prompts. Workers confirm their tasks by speaking pre-defined commands and reading confirmation codes printed on locations or products throughout the warehouse.

A significant drawback of VDW is that the information is volatile. Once information, such as the amount of items to be picked has been given, the system might not be able to repeat it. If the picker forgets such information, its retrieval might become laborious. Thus, visual UIs have gained importance in the task of supporting the picker. In previous work [1] the authors report on a 12 participant within-subjects experiment, demonstrating the advantages of a head-mounted display based picking chart over traditional text-based pick lists, paper-based graphical pick charts, and mobile pick-by-voice systems. No multimodal solution was investigated in that study. A different study [3] aimed at comparing various design solutions for head-mounted displays with different levels of information. The order was not only presented on a graphical UI (GUI) but also supported by some kind of sensors installed in the environment. This system exhibited a kind of Ambient Intelligence by detecting the picker's action of reaching into a shelf and comparing the respective box and its containing item with the order from the backend system. Such systems based on HMDs can be extended to support Augmented Reality, as shown in [2]. One limitation of these contributions is that they provide solutions that are implemented with ad hoc techniques and thus cannot be easily generalised to other similar applications. The use of model-based languages provides designers and developers with a general vocabulary for describing their solutions that can be refined in concrete terms for various interaction platforms, and then used to obtain implementations in a variety of languages, even for various combinations of interaction modalities. Thus, this model-based approach can be interesting in this application domain in which the combination of vocal and visual interaction has not been investigated so far. The level of multimodality, i.e. emphasising one of the two modalities, should depend on the context of use. For instance, in a noisy environment the interaction should rely mainly on the GUI. In addition, also the user's preferences and the capabilities of the platform should be considered. If the environment is fully intelligent, the picker can receive support for navigating through the location and picking the right items. Ideally, the system should be able to receive this context information and adapt the UI accordingly.

In this paper we present a solution consisting of an adaptive, context-sensitive UI which is based on an architecture for context-sensitive service front-ends. The solution is based on the use of model-based languages for interactive application descriptions in order to facilitate the possibility of deriving versions adapted to various

contexts of use, particularly in terms of interaction device resources. Such languages are currently under consideration for standardisation in W3C, because of their useful support in creating versions of interactive applications that adapt to different interactive devices. However, they have been mainly used in academic environments, with very few cases of use in real world applications.

In the paper, after discussing related work, we provide some background information related to the approach we have developed and how it has been extended in order to better support adaptive multimodal user interfaces for smart environments. Next, we describe the example application considered and how it varies depending on the context of use; followed by a description of the architecture supporting the adaptation of the multimodal service front ends. Then, we introduce some adaptation rules for the application considered, which are formalised in terms of *event*, *condition*, *action* (ECA) rules. We then show the corresponding prototype and report on an early user test focusing on the adaptation rules considered. Lastly, we draw some conclusions and provide indications for future work.

2 Related Work

Mobile applications often require highly-focused visual attention, which poses problems when it is inconvenient or distracting to continuously look at a screen (e.g., while walking). Aural interfaces support more eyes-free experiences, as users can primarily listen to the content and occasionally look at the device. However, designing aural information architectures remains a challenge. For example, recent studies [15] have highlighted that backward navigation is inefficient in the aural setting, as it forces users to listen to each previous page to retrieve the desired content. Thus, they have introduced topic and list-based back navigation strategies in order to enhance aural browsing and improve the navigation experience, reducing the perceived cognitive load. This shows the potential of multimodal interfaces in ubiquitous scenarios, but also the need for some specific design solutions, which depend on the interaction modalities exploited.

The problem of designing user interfaces that are able to be rendered on multiple types of platforms, including multimodal and vocal ones, has been addressed in some previous work, but still needs more general, better engineered solutions. Damask [10] includes the concept of layers to support the development of cross-device (desktop, smartphone, voice) UIs. Thus, the designers can specify UI elements that should belong to all the user interface versions and elements that should be used only with one device type. However, this approach does not consider the support of multimodal user interfaces. XFormsMM [8] is an attempt to extend XForms in order to derive both graphical and vocal interfaces. The idea is to specify the abstract controls with XForms elements and then use aural and visual CSS respectively for vocal and graphical rendering. However, aural CSS have limited possibilities in terms of vocal interaction and the solution proposed requires a specific ad-hoc environment. For this purpose we propose a more general solution which is able to derive implementations in various languages.

Obrenovic et al. [11] have investigated the use of conceptual models expressed in UML, in order to derive graphical, form-based interfaces for desktop, mobile or vocal devices. However, since UML is a software engineering standard, aimed at supporting the specification of the internal software application functionalities, it seems unsuitable to capture the specific characteristics of user interfaces. A different approach to multimodal user interface development has been proposed in [9], which aims to provide a workbench for prototyping UIs using off-the-shelf heterogeneous components. In that approach, model-based descriptions are not used and it is necessary to have an available set of previously defined components, which are able to communicate through low-level interfaces; thus making it possible for a graphical editor to easily compose them.

Sottet and others [13] have presented a set of general principles relevant for supporting model-based adaptation while in our case we present a software architecture supported by engineered tools that can be applied in real world applications.

Octavia et al. [12] have considered the use of a model-based approach to facilitate adaptation in virtual environments, also using the event-condition-action paradigm, we provide a more general architecture for this purpose able to support adaptation involving various interaction modalities.

To summarise, we can say that the few research proposals that have also considered multimodal interaction have not been able to obtain a suitable engineered solution in terms of logical descriptions and corresponding software architectures and provided limited support in terms of the generation of the corresponding user interface implementations. For example, in [14] the transformations were specified using attributed graph grammars, whose semantics is formally defined but have considerable performance limitations.

3 MARIA and Its Support for Multimodal Interaction

We exploit the MARIA model-based framework [5] for obtaining adaptation able to better support various interaction modalities. The framework provides a language for the abstract description (the so-called “Abstract User Interface” level, in which the UI is defined in a platform –independent manner) as well as multiple platform-dependent languages (which are at the level of the so-called “Concrete User Interface”), which refine the abstract language depending on the interaction resources at hand. Examples of platforms are the graphical desktop, the graphical mobile, the vocal platform, etc.

At the abstract level, a user interface is composed of a number of presentations, it has an associated data model, and can access a number of external functions. Each presentation is composed of a number of interactors (basic interaction elements) and a set of interactor compositions. There are four types of interactor composition operators: grouping, relation, composite description and repeater. These composition operators support the structuring of the elements inside a presentation. A *grouping* is a type of interactor composition used when a logic composition of interactors is needed. Therefore, grouping basically represents a generic group of interactor elements. A *relation* is an interactor composition which expresses a relation between an interactor

(or an interactor composition) and other interactors (or interactor compositions). A *composite_description* represents a group aimed to present contents through a mixture of only_output elements (namely: description/object/feedback/alarm) with navigator elements, while a repeater is used to repeat the content according to data retrieved from a generic data source. Each presentation is also associated with a dialogue model, which describes how the events generated by the interactors can be handled. With respect to previous languages in this area a number of substantial features have been added, such as a data model, a dialogue model, the possibility to specify typical Web 2.0 interactions, and the support to access Web services.

In its current version, MARIA consists of a set of languages: one for abstract user interface descriptions and a set of concrete refinements of such language for various target platforms (Vocal, Desktop, Smartphone with touch, Mobile, Multimodal desktop, Multimodal mobile). Moreover, user interface generators for various implementation languages are available starting with such concrete languages. The multimodal concrete language provides the possibility to indicate how to distribute the user interface elements across modalities through a simple and intuitive vocabulary that can be applied at various granularity levels. While previously this multimodal concrete language was associated with a generator of X+V implementation [6], in this work we consider a new generator able to create HTML 5 applications with multimodal features obtained using the Google support. Such support allows sending the user's utterance to a remote vocal recogniser in order to determine the corresponding input value, together with a Google Chrome extension that provides Text-to-Speech (TTS) access directly from JavaScript code. An HTML5 page developed with this extension consists of two parts: the graphical and the vocal one. These parts are linked by a rule applied to the id attributes of the HTML elements: the id of the vocal element is obtained adding "_vocal" at the end of the id of the graphical object. This allows us to obtain different implementations for the same interface element in the two modalities. Once a graphical element gets the UI focus, the extension retrieves its corresponding vocal element and invokes the TTS engine. The synthesis properties (e.g. speech, break, emphasis) are represented by a set of pre-defined CSS classes. If the graphical element is an input, the extension also starts the possibility of recording the voice and, when it detects a long silence after the vocal input, it invokes the ASR passing it the user's utterance. The result of the ASR is then used for filling the corresponding graphical element, or simulating a link or button click.

4 Example Application

In this work we want to exploit the model-based approach in supporting adaptation in real world applications relevant in the ambient intelligence domain. Thus, we provide some further detail on the application considered and how interaction can vary in it depending on the ambient intelligence available. The example application is situated at a distribution centre of a supermarket chain in the domain of retail industries. The task of the so-called pickers is to collect items from the shelves in the warehouse and place them into containers. One collection belongs to an order issued by a specific store of the supermarket chain.

The process starts when the picker signs-up for an order. Fig. 1 represents the scenario in Business Process Modelling Notation [4]. The simplified model for the warehouse picking process consists of Events of types: message (circle with envelope), Gateways of the type data-based exclusive (diamond with X), Tasks (rounded rectangles) processed by whether the system (scroll) or the picker (people), Data objects (document) and Text annotations (square bracket).

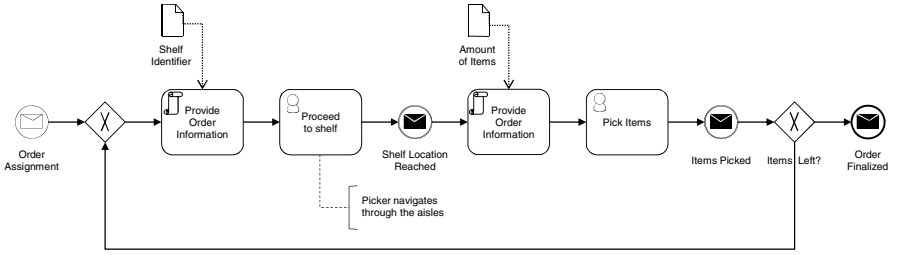

Fig. 1. Scenario for a warehouse picking process expressed in Business Process Modelling Notation [4]

After the Order Assignment, the system will provide the shelf identifier to the picker so they know where to find the items. The picker will then proceed to the shelf by navigating through the aisles. Depending on the initial location and the destination, s/he may pass through several halls. The system needs to be informed that the picker has reached the shelf location (i.e. a message needs to be sent). The system will then provide the amount of items to be picked and the picker can begin picking them. After this, the system needs to be informed about the completion of the picking. As long as there are still some items left, the process will loop back to the first Task. The process ends when the order is completed, i.e. when all items have been picked.

Now, we are going to consider the example application in two different contexts of use, with and without ambient intelligence. In the first situation (A) we assume that the warehouse is equipped with a kind of sensor which could support Ambient Intelligence. Concerning the example scenario described in this article, we assume that the system supports:

- the input of vocal prompts by the picker,
- tracking the position of the picker, e.g. through indoor navigation,
- identification of the actions of the picker, e.g. reaching into a shelf or,
- tracking the location of the items, e.g. by means of RFID tags.

In situation (A) the environment will trigger the event. In our example scenario the event “Shelf Location Reached” can be triggered by the module that tracks the position of the picker. The event “Item picked” can be triggered by a module that tracks the location of an item. In the second situation (B) we assume that the system can only support the input of vocal prompts by the picker. In this case the picker needs to trigger the event, i.e. issue messages to the system. In our example scenario, the event “Shelf Location Reached” can be triggered by the picker through a vocal interface. It is common that the picker reads a number from a sign which is attached to the shelf. The event “Item picked” can be triggered in a similar fashion. The picker then repeats the amount of items that have been picked.

It is clear, that the situation at a specific warehouse might be somewhere in between situation A and B. For example, one warehouse might have a tracking system for the picker but not for the items and vice versa. Thus, it would be desirable to have an interactive application that could adapt to the specific situation of the warehouse environment.

5 Architecture

Our architecture shows how we can provide support for adaptation through the use of model-based descriptions of interactive applications. At design time the various initial versions of the applications are developed in terms of the three concrete descriptions (vocal, graphical, and multimodal) specified according to the MARIA language. In addition, the relevant adaptation rules are specified in terms of events, conditions, and actions according to a language for adaptation rules that will be described later on. Such adaptation rules are triggered by contextual events, which can depend on various aspects (user preferences, environmental changes, application-related events, etc.). The impact of the adaptation rules can have various granularities: complete change of user interface (e.g. from vocal to graphical if the environment becomes noisy), change of some user interface parts (e.g. change from map view to order view), and even change of attributes of specific user interface elements (e.g. change of font size).

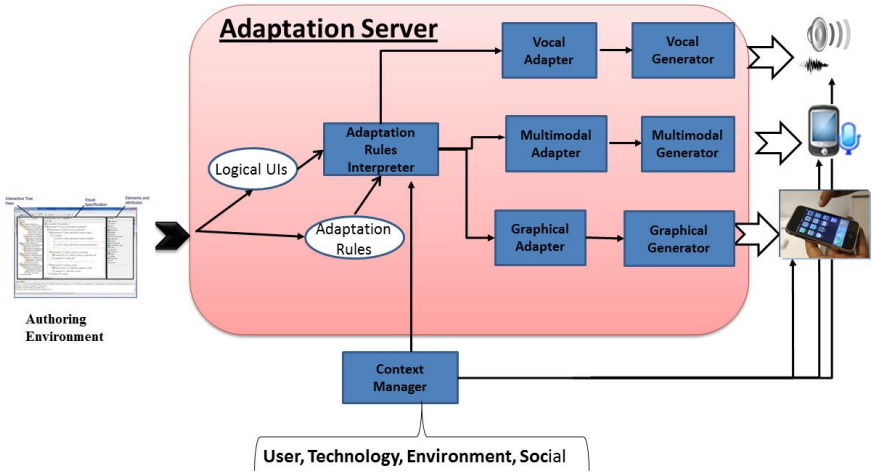

Fig. 2. The software architecture supporting the adaptive application

Thus, at design-time it is possible to specify the relevant logical descriptions of the interactive application versions and the associated adaptation rules.

At run-time we have an adaptation server that is able to communicate with the context manager server in order to receive information on subscribed events and the interactive devices available in order to update the application according to the adaptation rules. The context server is also able to communicate with the applications, which can manage directly some contextual events according to their specifications.

More specifically, the adaptation rules interpreter has access to the list of the relevant adaptation rules. Changes in the context communicated by the context manager can trigger some of them. The adaptation rule interpreter considers the action part of the rules and depending on its content it can either trigger the activation of the application in a different modality (which means to trigger a new application generator for the most relevant modality in the new context of use) or indicate some change to perform to the adapter associated with the current modality (in this case the adapter will then request the corresponding generator to update the interactive application accordingly). The languages to specify the adaptation rules and the interactive applications are distinct, so that it is possible to modify one without having to change the other one, but with clear relations defined among them so that the actions of the adaptation rules can be specified in terms of required modifications to the model-based descriptions of the interactive applications. Thus, the logical description of the interactive application can dynamically change from the version that was initially provided by using the authoring tool.

6 Adaptation Rules

We developed a XML-based high-level description language intended to declaratively express advanced adaptation logic defining the transformations affecting the interactive application when some specific situations occur both in the context (e.g. an entity of the context changes its state), and in the interactive application (e.g. an UI event is triggered). In particular, the three parts of the language are: *event*, *condition*, *action* (ECA). The *event* part of the rule should describe the event whose occurrence triggers the evaluation of the rule. This part could specify elementary events occurring in the interactive application, or a composition of events. The *condition* part is represented by a Boolean condition that has to be satisfied in order to execute the associated rule action(s). The condition part is optional. In the *action* part there might be 1 to N simple actions occurring in the interactive application or even 1 to N other adaptation rules. In practise, the action part often contains indications on how the concrete description of the interactive application should change in order to perform the requested adaptation. Event Condition Action is an approach that was originally introduced for the structure of active rules in event driven architecture and active database systems, and has already been used for supporting adaptive user interfaces (see for example [12]). In our case, we have structured it in such a way to easily connect it to the events generated by the context manager and the interactive application specification.

Below there is a list of example adaptation rules supported by the prototype. For each rule we provide a title with a brief explanation/rationale and the three key parts of its specification (event, condition, action).

- *Fragile object* - The rationale of this rule is that when the worker is about to pick a fragile object, the multimodal UI should switch to only-vocal modality in order not to distract the user while picking the item.
    - *Event*: the right shelf has been reached

  - *Condition*: the worker has to pick a fragile item and the current modality is not only-vocal
  - *Action:* Switch from multimodal to only-vocal modality,
- *Picking timeout* - The user has just reached the destination shelf of the item but there is no confirmation of the actual item picking. The application then assumes that the worker is distracted/confused and/or not able to recognize the item to pick, then it provides again info on the item, both graphically and vocally.
  - *Event*: the user has reached the destination shelf
  - *Condition*: there has not been confirmation of the item picking and the user interface is multimodal
  - *Action*: the application visualizes an image representing the item to pick, and simultaneously repeats the item name vocally.
- *Order visualization for experienced workers* - If the user has good knowledge of the warehouse shelf organisation, there is no need to show associated path information: the application adapts accordingly.
  - *Event*: beginning of a session with the HMD
  - *Condition*: the user is a warehouse expert
  - *Action*: the application hides the information about how to reach the different shelves.
- *Traffic Jam* - There are multiple workers who are expected to approach the same path at the same time: the application adapts in order to minimise the risk of workers to wait for other people before picking the items.
  - *Event*: order completed
  - *Condition*: multiple pickers are expected to approach the same path at the same time and the path optimization preference is selected.
  - *Action*: the application shows the blocked path suggesting a different route.
- *Noisy environment* – The environment gets noisy, then the multimodal application switch to 'only-graphical' modality.
  - *Event*: the environment gets noisy
  - *Condition*: the application is using both the graphical and vocal modality for interacting with the user.
  - *Action*: the application switches to the only-graphical modality

We show how it is possible to express such adaptation rules through our high-level description language with one example. We consider (Fig. 3) the rule for the order visualization for experienced workers. When the interaction starts (the presentation raises the *onRender* event), if the current user is an expert one (represented as an attribute in the 'user' part of the context model), the application hides the path to reach the shelf, which is represented by an interactor with id *path_to_shelf*, setting its *hidden* attribute to *true*. A symmetrical rule manages the case of inexperienced workers.

```
<rule>
    <event>
        <simple_event event_name="onRender"
            xPath="/interface/"
            externalModelId="uiModel"/>
    </event>
    <condition operator="eq">
        <entityReference xPath="/context/users/user/@experience"
            externalModelId="ctxModel"/>
        <constant value="high" type="string"/>
    </condition>
    <action>
        <update>
            <entityReference
                xPath="/interface[@current_presentation]/interactor[@id = 'path_to_shelf']/@hidden"/>
            <value>
                <constant value="true" type="boolean"/>
            </value>
        </update>
    </action>
</rule>
```

Fig. 3. Formalization of the Order visualization for experienced workers rules

7 Adaptive Application

We start with a description of the graphical version of the interactive application considered. The GUI consists of four views (Order, Map, Task and Statistics), for the sake of brevity only the Order view and the Map view are discussed. The Order view, shown in Fig. 4, mainly contains information on the previous (i.e. shelf 433), the current (i.e. shelf 473) and the next (i.e. shelf 481) items to be picked. This sequence of picks is represented in three rows starting with the previous pick and having the current pick highlighted (i.e. inverted) and magnified.

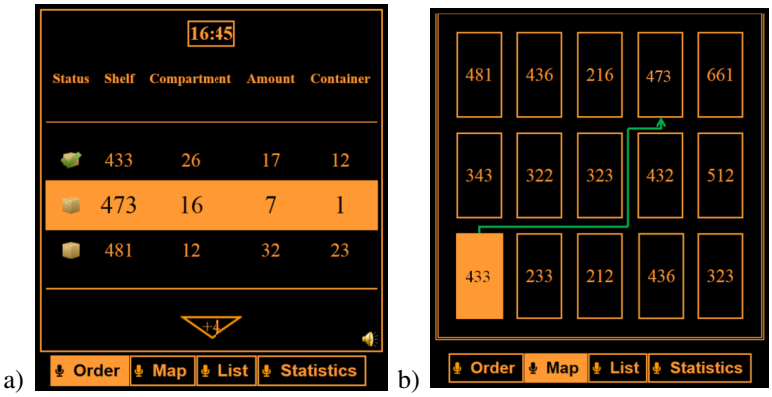

Fig. 4. Design of the graphical user interface (GUI). a) Order view b) Map view

The columns reflect the types of information available for the pick (status, shelf, compartment, amount and container) while only the status of the pick (e.g. open), the shelf identifier (e.g. 473) and the amount of items to be picked (e.g. 7) are relevant here. The active view is reflected as a highlighted tab in the bottom area. The main

information in the Map view is a simplified representation of the location of the shelves (in Bird eyes view) showing the current location of the picker (i.e. the previous shelf), the destination shelf (i.e. 473) and a suggested route (line with arrow). In general it is possible to navigate between the screens by using voice commands, but this functionality is not used in the actual setting.

Based on a list of requirements for the prototype a Head-Mounted Display (HMD) and a wearable computer are used to access the application. The UI generated from the MARIA specification is implemented in HTML5, JavaScript and AJAX. The navigation route in the Map view is drawn using the canvas label of HTML5. Speech recognition is realized using the speech input label of HTML5 and calling a respective API. The architecture of the application implementation is shown in Fig. 5.

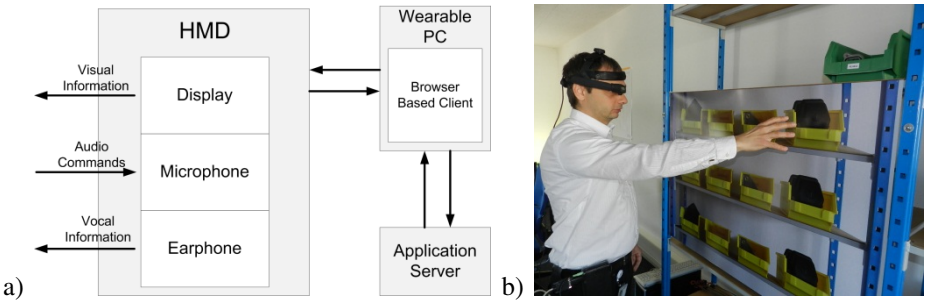

Fig. 5. a) Architecture of the prototype. b) Picking from a shelf using a Head-Mounted Display.

The adaptation server sends the updated data to the wearable computer after a change in the context has triggered the execution of an adaptation rule. The display is used for the visual output, the earphone for the vocal output and the microphone for the vocal input of the user. Some changes might be triggered by the smart environment (e.g. tracking of the picker's position or the item's location). Table 1 lists the five variations of the context and its consequences for the interaction modalities with respect to the basic interaction flow. The variations are based on the *Condition* and the consequences derive from the *Action* stated in the respective adaptation rule.

Table 1. Variations of the context and its consequences for the interaction modalities

Context variation	Interaction consequence
The items to be picked are fragile	After vocally confirming the arrival at the destination by the picker, the visual output will be switched off, only vocal remains.
The route is blocked by other pickers.	The Map view marks the blocked path and suggests an alternative route.
The picker is experienced	The Map view is omitted.
The environment is noisy	The vocal input and output is switched off, only visual output remains
The picking is not performed due to some confusion or distraction	An image of the item to be picked is shown, the vocal output is repeated.

Finally we present the basic interaction sequence (i.e. the basic interaction flow) with an example for an adaption in Fig. 6: the picker is presented with three screens and two vocal outputs (upper balloons) and needs to perform two vocal inputs (lower balloons). Assuming that a picker, who is experienced, i.e. has been working for a long time in the warehouse environment and thus should know by heart the location of the shelves, and the Map view can be omitted. We assume that an indicator of the experience level is stored within the profile of the picker and is added as context information at run-time during the log-in procedure.

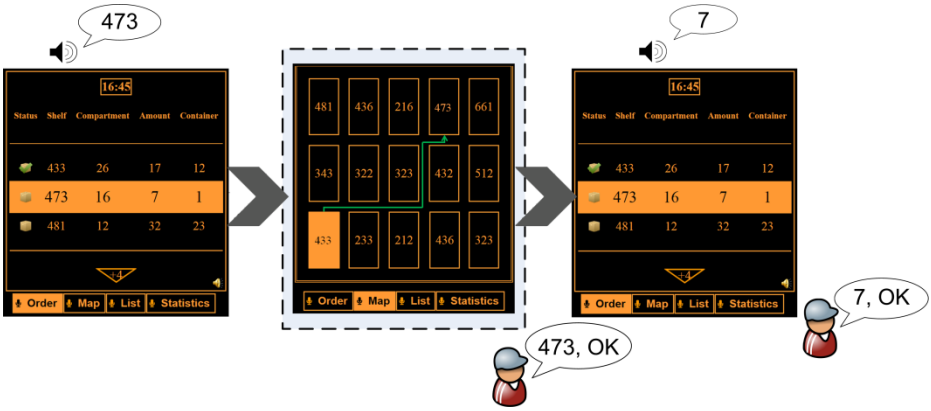

Fig. 6. Basic interaction flow with adaptation: the execution of the rule for an experienced picker omits the appearance of the Map view (dotted line)

8 User Feedback

We have conducted a first user study in order to evaluate the five adaptation rules from the end-users point-of view. The study aimed at evaluating the applicability and usefulness of the adaptation rules, specified as described in Section 6 through an XML-based ECA style, by assessing the quality of the adaptation rules as subjectively perceived by the participants. The general concept "quality" was operationalized by several more specific constructs, e.g. usefulness, comprehensibility or simplicity, which were assessed by a questionnaire.

To address such issues, the five adaptation rules were the independent variables. We had a within-subject design, meaning that every participant was confronted with every adaptation rule. The dependent variables were the subjectively perceived quality of the adaptation rule as assessed in a 9-item questionnaire. The questions originated from a list of non-functional requirements for the prototype identified in user studies in the beginning of the project and aimed at assessing the following aspects: the user's awareness for the adaptation rule, its appropriateness and comprehensibility, its effectiveness with respect to performance and usability, its error-prevention, continuity, intuitiveness, and general likeability.

Participants were company staff or students of the local university. A total of 10 participants took part in the study, 9 were male and 1 was female. The average age of participants was 24 years (SD = 1.82). The technical set-up consisted of an HMD with earphone worn by the participants. The device presented the GUI and the vocal output as shown in section 7. The sequence of the interaction was controlled by the moderator simulating the change of context and the execution of the adaptation rule.

Participants were first introduced into the scenario and the interface, i.e. getting familiar with the hypothetical situation in the warehouse and learning how to interact with the interface. Participants were asked to play through a "basic interaction flow" which started with the systems request to pick items from a certain shelf, required the user to hypothetically walk to that shelf and ended with the user's confirmation that he picked a certain amount of items. Participants were asked to comment their hypothetical actions, e.g. by saying "I walk to the shelf 473 now" or "I pick 7 items from the shelf". After ensuring that the participants understood the basic interaction flow of the interface, the study started by introducing the first alternative flow. All alternative flows (flows containing adaptation rules) were applied to the same scenario as practiced in the basic flow. Prior to playing through the alternative flows, participants were informed about the condition of the adaptation rule (e.g. "imagine you are now in a noisy environment"), but not about the actual rule (i.e. the action of the rule). All five rules were played through and the sequence of the adaptation rules was permutated to avoid order effects. After each rule, the 9-item questionnaire was filled out.

Since most of the scales of the questionnaire were not normal-distributed, we applied non-parametric tests for the data analysis. We calculated the Friedman test for every single questionnaire scale and the aggregated overall rating from all 9 scales (Bonferroni-corrected) to assess differences between the five adaptation rules. In case of significance, we calculated a post-hoc Wilcoxon signed-rank test for each pair of adaptation rule (Bonferroni-corrected as well).

The Friedman test revealed significant differences for the aggregated overall rating over all 9 scales ($\chi^2(4) = 18.74$, $p = .001$) and for 4 of the subscales: Appropriateness ($\chi^2(4) = 19.26$, $p = .001$), Performance ($Z = -2.69$, $p=.007$), Error-Prevention ($\chi^2(4) = 22.73$, $p = .000$), Intuitiveness ($\chi^2(4) = 22.31$, $p = .000$) and General Likeability ($\chi^2(4) = 18.92$, $p = .001$). Only these significantly different scales are regarded in detail here. Post-hoc tests revealed a significant difference in the rating between the rules Fragile Objects and Traffic Jam ($Z = -2.60$, $p = .009$) and Experienced Worker and Traffic Jam ($Z = -2.70$, $p=.007$). The significant differences in the subscale Appropriateness are between the rules Fragile Objects and Traffic Jam ($Z = -2.62$, $p = .009$) and Fragile Objects and Pick Timeout ($Z = -2.69$, $p = .007$). For the subscale Error prevention, the significant differences can be found between the rules Fragile Object and Pick Timeout ($Z = -2.71$, $p = .007$), Traffic Jam and Experienced Worker ($Z = -2.81$, $p = .005$) and Pick Timeout and Experienced Worker ($Z = -2.68$, $p = .007$). Intuitiveness shows significantly different values for the rules Fragile Objects and Traffic Jam ($Z = -2.69$, $p = .007$). Finally, although the Friedman test revealed significant differences between the rules for the scales: general Likeability and Performance; direct pairwise comparison failed reaching significance due to Bonferroni correction.

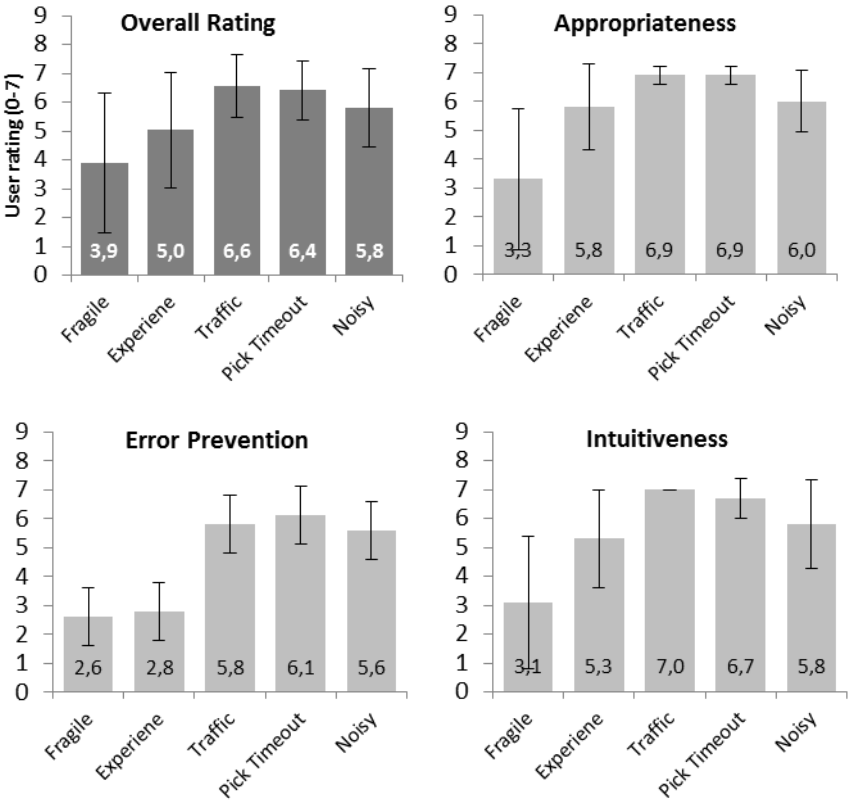

Fig. 7. Overall rating and the subscales Appropriateness, Error-Prevention and Intuitiveness

The big picture of the results (see Fig. 7) shows a clear trend: all quality aspects of the Fragile Object rule are consistently rated the worst, and the Traffic Jam and Pick Timeout rule are consistently rated best. This pattern can be observed for all quality scales, indicating a clear and coherent preference pattern. Traffic Jam and Pick Timeout are consistently and undoubtedly preferred by the users (with very good overall ratings of 6.6 and 6.4 on a scale from 0-7). Alongside the good rating of these two rules, the standard deviation is very small, indicating a very high agreement between the participants. However, the Fragile Object rule, as the worst rated one, shows the highest variance in the ratings between the subjects. This indicates that there is no strong agreement between the subjects, yet still most of the subjects gave comparably low ratings for that rule. A possible explanation for this finding can be drawn from the subject's comments. While all subjects gave a positive opinion about the idea to support the process of picking a fragile object, most of the subjects noted that the actual realisation of that rule was poor. Turning off the display was irritating and non-intuitive to the subjects. The abrupt darkness in the HMD was perceived as a

break-down of the system and therefore caused confusion. Rather, subjects had wished to receive a short warning message before turning off the display.

We found similarities between those rules that were ranked well and those that were ranked poor. The group of poorly ranked rules was omitting information like the visual output and the Map view with regard to the Basic Interaction Flow. The Fragile rule takes a prominent position as a very strong modality, the visual channel, is shut off. Those rules that were ranked well however delivered additional information like the blocked path or the image of the item. This noticeable difference between the adaptation rules is presumably the reason for the striking difference in the preference ratings. It is worth investigating the role of adding vs. removing information as well as amount of information in the course of interface adaptation.

9 Conclusions

Work applications often need intelligent environments able to provide adaptive user interfaces that change the interaction modalities taking into account contextual aspects. In this paper we have reported a solution exploiting the use of model-based descriptions of interactive applications that facilitate the development and the dynamic update of versions that depend on the interaction resources available. The models allow the generation of different versions of the interactive application that exploit the different modalities according to policies defined in adaptation rules, which are separated from the UI definition and can be modified as a separate aspect. The solution proposed is able to support adaptation at various granularity levels ranging from changing the interactive application since the interaction modality has changed to small modifications of the current version. We have discussed its application to a warehouse picking case study, indicating how a set of integrated tools (authoring environment, adaptation server, dynamic interactive application generators) have been exploited, and we have also reported on an early user test related to the adaptive rules considered.

The result of the user test showed in general that adaptive rules are received well if they trigger the delivery of additional information. Omitting some information without notification, however leads to some usability problem.

The MARIA authoring environment including the interactive application generators used in this work are publicly available at http://giove.isti.cnr.it/tools/MARIAE/home

Future work will be dedicated to further engineering the solution proposed and apply it to other case studies. In addition, we plan to do some studies targeting designers and developers of adaptive interactive applications in order to better assess how their work is facilitated through a model-based approach.

This work has been supported by the SERENOA EU ICT Project, http://www.serenoa-fp7.eu/

References

1. Weaver, K.A., Baumann, H., Starner, T., Iben, H., Lawo, M.: An empirical task analysis of warehouse order picking using head-mounted displays. In: 28th International Conference on Human Factors in Computing Systems (CHI 2010). ACM, New York (2010)
2. Schwerdtfeger, B., Klinker, G.: Supporting Order Picking with Augmented Reality. In: 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, pp. 91–94 (2008)
3. Ali, S., Lewandowski, A., Rett, J.: A SOA based context-aware order picking system for warehouses using Laser Range Finder and wearable computer. In: 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1–8 (2011)
4. Object Management Group (OMG): Documents Associated with Business Process Model and Notation (BPMN) Version 2.0, `http://www.omg.org/spec/BPMN/2.0/`
5. Paternò, F., Santoro, C., Spano, L.D.: MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. ACM Trans. Computer-Human Interaction 16(4), 1–30 (2009)
6. Manca, M., Paternò, F.: Supporting Multimodality in Service-Oriented Model-Based Development Environments. In: Forbrig, P. (ed.) HCSE 2010. LNCS, vol. 6409, pp. 135–148. Springer, Heidelberg (2010)
7. Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., Young, R.: Four Easy Pieces for Assessing the Usability of Multimodal Interaction: the CARE Properties. In: Proceedings INTERACT 1995, pp. 115–120 (1995)
8. Honkala, M., Pohja, M.: Multimodal interaction with XForms. In: Proceedings ICWE 2006, pp. 201–208 (2006)
9. Lawson, J., Al-Akkad, A., Vanderdonckt, J., Macq, B.: An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components. In: Proceedings ACM EICS 2009, pp. 245–254 (2009)
10. Lin, J., Landay, J.A.: Employing Patterns and Layers for Early-Stage Design and Prototyping of Cross-Device User Interfaces. In: Proc. CHI, pp. 1313–1322 (2008)
11. Obrenovic, Z., Starcevic, D., Selic, B.: A Model-Driven Approach to Content Repurposing. IEEE Multimedia, 62–71 (January, March 2004)
12. Octavia, J., Vanacken, L., Raymaekers, C., Coninx, K., Flerackers, E.: Facilitating Adaptation in Virtual Environments Using a Context-Aware Model-Based Design Process. In: England, D., Palanque, P., Vanderdonckt, J., Wild, P.J. (eds.) TAMODIA 2009. LNCS, vol. 5963, pp. 58–71. Springer, Heidelberg (2010)
13. Sottet, J.-S., Ganneau, V., Calvary, G., Demeure, A., Favre, J.-M., Demumieux, R.: Model-Driven Adaptation for Plastic User Interfaces. In: Baranauskas, C., Abascal, J., Barbosa, S.D.J. (eds.) INTERACT 2007. LNCS, vol. 4662, pp. 397–410. Springer, Heidelberg (2007)
14. Stanciulescu, A., Limbourg, Q., Vanderdonckt, J., Michotte, B., Montero, F.: A Transformational Approach for Multimodal Web User Interfaces based on UsiXML. In: Proc. ICMI, pp. 259–266 (2005)
15. Yang, T., Ferati, M., Liu, Y., Ghahari, R.R., Bolchini, D.: Aural Browsing On-The-Go: Listening-based Back Navigation in Large Web Architectures. In: Proceedings ACM CHI 2012, pp. 277–286. ACM Press (2012)