A Multi-Agent System Architecture for the Adaptation of User Interfaces

Víctor López-Jaquero, Francisco Montero, José P. Molina, Pascual González, Antonio Fernández-Caballero

Laboratory on User Interaction & Software Engineering (LoUISE)
University of Castilla-La Mancha, 02071 Albacete, Spain
{ victor | fmontero | jpmolina | pgonzalez | caballer }@info-ab.uclm.es

Abstract. Nowadays the design of user interfaces has become a discipline of great importance in Software Engineering, mainly due to the increasing impact that a high quality user interface has in the success of a software product. However, the growing diversity in interaction devices and techniques has raised a big expectation for the design of both methods and architectures able to cope with context of use heterogeneity issues in an intelligent way. Multi-agent systems jump into scene as an alternative to design the adaptation capabilities required to cope with this problem in a natural manner.

1 Introduction

Nowadays the design of user interfaces has become a discipline of great importance in Software Engineering, mainly due to the increasing impact that a high quality user interface has in the success of a software product. To face the challenge of designing once and running in many different contexts of use (for instance, in different devices: PC, PDA, …), model-driven architecture (MDA) appears as a solution, where the application is derived from a series of models describing both the dynamic and static aspects of an application. For the last decade, MDA has been a really active thread in UIs design research community, in the form of Model-Based User Interface Development Environments (MB-UIDE) [9]. In the design of a general technique that supports adaptivity [1] in a flexible manner, where knowledge can be reused and integrated with a UI design method that provides the required formalism to build UIs in a systematic way [7][6], a software architecture able to cope with all these requirements is needed. However, this software architecture needs to be able to make decisions about which adaptations should be applied, when they should be applied, etc. System decisions about adaptation should be grounded in the UI model developed at design time, along with the information about the context of use that the system gets from the environment. In this paper the use of the multi-agent paradigm in the design of a software architecture to support adaptative behaviour in UIs is proposed. A set of agents collaborate in a multi-agent system (MAS) to achieve the final goal of adaptation, by receiving through their sensors the changes in the environment (context of use) where they are involved.

[1] This work is partly supported the Spanish PBC-03-003 and CICYT TIN2004-08000-C03-01 grants.

2 The Adaptation Process

Within adaptivity there is a wide range of possibilities where several actors (usually the system and the user) can take the initiative in the different stages carried out in order to perform the adaptation. Thus, this adaptation is not preformed automatically, but semiautomatically. The stages needed to perform adaptation according to [3] are: (1) **initiative:** one of the actors involved in the interaction suggests its intention to perform an adaptation. The main actors are usually the user and the system, (2) **proposal:** if a need for adaptation is detected, it is necessary to make proposals of adaptations that could be applied successfully in the current context of use for that need for adaptation detected, (3) **decision:** as we may have different proposals from the previous stage we need to decide which adaptation proposals best fit the need for adaptation detected, and even if it is worth applying any of them, and (4) **execution:** finally, the adaptation chosen will be executed.

3 A Multi-Agent Architecture For Adaptive User Interfaces

The multi-agent system perceives the changes in the context of use by means of sensors. Then, a set of adaptations will be chosen among the feasible adaptations taking into account the expected benefit evaluation that each feasible adaptation would produce to the user if it would be applied. Finally, the selected adaptations will be applied following a transformational approach.

At *Initiative stage* the adaptation process is fired. This can be achieved mainly in two different ways: (1) the user explicitly expresses his intention to perform an adaptation, (2) the system or a third-party agent detects that an adaptation might be helpful or needed. In this second case, this stage can be subdivided into two smaller sub-stages. On the one hand, the system needs to guess the current goal the user is pursuing, and on the other hand it should guess which needs the user has with respect to the detected current goal. In the multi-agent system proposed for the adaptation of the UI this stage is performed by means of *AgentContextPlatform*, *AgentContextEnvironment*, *AgentContextUser*, and the user itself. The three agents receive any change in the context of use perceived by the sensors and take advantage of the data collected during design and the data perceived to figure out whether the adaptation process should be fired or not.

Within agent design paradigm, just as in human reasoning model, the possible actions that an agent can use to face a situation (a change in the context of use in our case) are the plans. Thus, *AgentAdaptationProcess* agent has a plan for each possible adaptation that can be applied. Adaptations are represented as adaptivity rules at design time, which are translated later into agent's plans, following an approach based on *Prometheus* [8] method plan specification. The meta-model for an adaptivity rule is specified in terms of the context-of-use events that trigger the adaptivity rule, the sensors that produce those events, the data the rule accesses (read/write), the transformations of the UI needed in order to apply the rule, and the context precondition. The context precondition specifies the required conditions that the current context of use must meet in order for the adaptivity rule to be applicable. The transformation specifies the "real" adaptation of the UI. These transformations modify

a graph representation of the usiXML [10] specification of the running UI. To modify the representation of the graph an attributed graph grammar engine is used.

Decision stage is performed by *AgentAdaptationProcess*. This agent will use the selected selection policy to choose the adaptations that should be applied. Notice that the user can also decide which adaptations to apply among the adaptations that match the current changes in context. We have two different policies available for the selection of the adaptations that best fit a context of use change. The first one is the simpler one. When this first adaptation selection policy is chosen, the first rule that could be applied to the current situation is selected. The selection order is the same as the order in which the rules were fired. Therefore, following this first policy no meta-planning method is required. However, it will yield unpredictable results in many cases, making adaptation a useless feature. The second policy is a little more sophisticated. It selects the rules taking into account usability criteria evaluation.

Execution stage is also performed by *AgentAdaptationProcess* agent. In this stage, there are three main sub-steps: (1) Get an up-to-date copy of the UI expressed in terms of usiXML language, (2) Apply the adaptations. The adaptations are applied performing the transformations specified for each chosen plan. And, (3) restore the UI out of the newly generated one. The adapted UI will be shown to the user, restoring interaction to the state it was before adaptation took place. An agent called *AgentStimuliGenerator* has been added to the architecture to make debugging and evaluation easier. This agent simulates the arrival of data from the sensors, following a pattern of events specified by the designer.

4 Implementing The Adaptive Architecture

In this section we will show an overview of the technologies used in the implementation of the architecture. For the MAS implementation we have used JACK Intelligent Agents™ [2]. To maximize platform independence we have wrapped the multi-agent java based system within an HTTP server interface. The HTTP server interface allows any platform capable of networking using TCP/IP protocol to access the adaptation engine. This HTTP server has been implemented as a servlet (server side applet) that runs on top of a TOMCAT server.

usiXML UI description language is able to describe a UI in a manner independent from the platform where it will run on. Therefore, a renderer is needed so the user can visualize the UI. For this purpose, a renderer for the concrete UI level of usiXML has been written for XUL language. This renderer translates a usiXML CUI specification into a XUL language specification that can be visualized by the user. XUL is an XML-based UI language that can be visualized in any Internet browser based on Mozilla engine (http://www.mozilla.org). At this moment, we have implemented the embedded sensors needed to capture the data from the interaction using JavaScript. JavaScript allows the implementation of the dynamic behaviour of the UI. The engine to execute the transformations associated to the adaptations uses the API from AGG [5] tool to perform the transformations. The engine transforms a usiXML specification expressed as an attributed graph into a new usiXML specification transformed according to the adaptation rules selected. In [5] a detailed description of the transformation process can be found.

5 Final Remarks

There have been many works related to the adaptation of UIs, especially in the field of intelligent tutoring systems. However, most of that research has led to solutions where the adaptations were hardcoded within the system, making it very difficult to modify the way adaptations are made, or to reuse the solution from one application to another. In this paper a MAS is proposed that is able to cope with the adaptation process in a flexible way, and where the same language is used for the specification of both the UI and the adaptation rules. Furthermore, the system detects the context of use by means of a set of sensors that modify the context model included inside the agents' beliefs, making the MAS react to accommodate the UI to the different situations produced by the changes in the context of use detected by sensors.

References

1. Benyon D., Murray D.. Developing adaptive systems to fit individual aptitudes. IUI 1993, pp. 115-121, Orlando, Florida, United States, ACM Press, 1993.
2. Busetta, P., Ronnquist, R., Hodgson, A. and Lucas, A. Jack intelligent agents - components for intelligent agents in java. AgentLink News Letter, January 1999. White paper.
3. Dieterich, H., Malinowski, U., Khme, T. and Schneider-Hufschmidt, M. "State of the Art in Adaptive User Interfaces". In: Schneider-Hufschmidt, M., Khme, T. and Malinowski, U., eds.: Adaptive User Interfaces: Principle and Practice. Amsterdam, Holland, 1993.
4. Fernández-Caballero, A., López-Jaquero, V., Montero, F., González, P. Adaptive Interaction Multi-agent Systems in E-learning/E-teaching on the Web. International Conference on Web Engineering, ICWE 2003. Springer Verlag, pp. 144-154. 2003.
5. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V. USIXML: a Language Supporting Multi-Path Development of User Interfaces. Proc. of 9th IFIP Working Conference on Engineering for HCI jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems (Hamburg, July 11-13, 2004). LNCS, Vol. 3425, Springer-Verlag, Berlin, 2005, pp. 207-228.
6. López-Jaquero, V., Montero, F., Molina, J.P., Fernández-Caballero, A., González, P. Model-Based Design of Adaptive User Interfaces through Connectors Design, Specification and Verification of Interactive Systems 2003, DSV-IS 2003. Springer Verlag, 2003.
7. López-Jaquero, V., Montero, F., Molina, J.P., González, P., Fernández-Caballero, A. A Seamless Development Process of Adaptive User Interfaces Explicitly Based on Usability Properties. Proc. of 9th IFIP Working Conference on Engineering for HCI jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems (Hamburg, July 11-13, 2004). LNCS, Vol. 3425, Springer-Verlag, Berlin, 2005.
8. Padgham, L., Winikoff, M. Prometheus: a methodology for developing intelligent agents. AAMAS 2002: 37-38
9. Paternò, F. Model-Based Design and Evaluation of Interactive Applications. Springer Verlag, 2000.
10. usiXML specification. Available at http://www.usixml.org
11. Wooldridge, M., Jennings, N.R. Agent Theories, Architectures, and Languages: A Survey, ECAI-Workshop on Agent Theories, Architectures and Languages. Wooldridge, M.J. and Jennings, N.R. (eds.), 1994.