Applying Model-Based Techniques to the Development of UIs for Agent Systems

Sebastian Ahrndt*, Dirk Roscher, Marco Lützenberger, Andreas Rieger, and Sahin Albayrak

Abstract. To counter difficulties of user interface (UI) development, model based techniques became firmly established over the last years. The basic idea of model based user interface development (MBUID) is to formally specify a UIs appearance and behaviour by means of several models. Especially for distributed multi-agent systems, the appliance of MBUID can be most promising. Agent applications involve many different execution platforms and heterogeneous devices and perfectly fit for Ambient Assisted Living landscapes due to their innate characteristics of distribution and autonomy. When it comes to agent systems, one always has to consider the fact that humans have to communicate with agents in the end. It is our opinion that most approaches neglect this fact and thus cut the dynamics and the capabilities of distributed multi-agent systems. Hence in this work, we present an approach for the development of UIs for software agents which applies model based techniques and also retains all degrees of freedom for the underlying multi-agent system.

1 Introduction

Ambient Assisted Living (AAL) is strongly facilitated by the vision of ubiquitous computing, where smart interacting devices are integrated into the everyday life. As a matter of fact, the importance of AAL services increases over time as a result of demographic changes. In order to maintain the quality of life – especially for the elderly – technologies are required which support a living at home in many aspects, such as autonomy, security and health.

Over the last years, *Agent Oriented Software Engineering* (AOSE) has evolved as suitable technique for the development of AAL systems [6]. The reason for this is that multi-agent systems (MAS) perfectly fit for AAL landscapes due to their

Sebastian Ahrndt · Dirk Roscher · Marco Lützenberger · Andreas Rieger · Sahin Albayrak
DAI-Labor, Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany
e-mail: sebastian.ahrndt@dai-labor.de
*Corresponding author.

J.M.C. Rodríguez et al. (Eds.): Trends in PAAMS, AISC 157, pp. 1–8.
springerlink.com

innate characteristics of distribution and autonomy. In fact, agent-based systems are able to match many requirements of Ambient Assisted Living. However, it is generally agreed, that the success of software applications is not only founded by the capability of the application itself, but also by the quality of its handling and also by its usability. When it comes to AAL, one always has to consider, that the target audience is usually the elderly. This raises many challenges for software developers as elderly people are not as experienced in handling software as younger people are [7]. Further, the situation is aggravated, as different device types and many interaction modalities (such as voice-, mouse-, touch- and gesture-based interaction) have to be taken into account. To sum up, in order to support users in the spirit of AAL, developers have to provide users with intuitive and yet non-intrusive control mechanisms.

However, serving multi-modal interaction possibilities and also supporting different device types results in countless UI variations and even more configuration options. To counter this problem, user interfaces for similar application areas are frequently developed in compliance with the *Model-Based User Interface Development* (MBUID) paradigm. The basic idea of MBUID is to formally specify a user interface's appearance and behaviour by means of several models from which executable code can be derived. Further, interpreter-based *Model-based User Interfaces* (MBUI) have the ability to manipulate their models at runtime and to dynamically adjust to the current execution context.

In this paper, we argue that the combination of software agents and MBUIs is a sophisticated way to increase the comfortability when developing AAL services. We start with a survey on related approaches (see Section 2). Afterwards, we present an approach that enables the development of agent-applications with MBUIs that are interpreted at runtime in order to provide a holistic user experience for AAL environments (see Section 3). Subsequently, we will illustrate a proof-of-concept implementation of an AAL service which we currently present in the showroom of our research institute (see Section 4). We proceed by discussing practical experiences we have made thus far and finally wrap up with a conclusion (see Section 5).

2 Related Work

Prior to our development, we performed a survey on existing approaches. The HCI community provides an established body of works regarding MBUIs and MBUI development environments [11, 14]. However, these works do not contribute to the integration of MBUIs into the agent domain. As a matter of fact, this area of research is only sparely covered. The agent community for instance tries to counter the complexity of UI development by web-based solutions [1, 15]. As these approaches are not directly comparable with our architecture, we identified some others which are described next.

Braubach et al. [4], for instance, introduce *Vesuf*, a development environment for MBUIs. Vesuf was not streamlined for agent applications, however, the

framework was tested in real life, in an urban hospital facility[1], where it demonstrated its capability to generate adaptable UIs for software agents. In their work, the authors emphasise the difficulties in developing UIs for agent systems and argue, that interpreter-based MBUIs are capable to overcome most of the mentioned problems.

Eisenstein and *Rich* [8] propose an architecture which is based on task-models and which facilitates the development of collaborative interface agents. The authors apply task-models to control the behaviour of agents and also as foundation for the UI. Development is done in compliance with the underlying task-model and supported by a set of editors, each one geared towards a specific part of the application.

Tran et al. [18] present an approach which applies MBUID for data systems. The authors present an agent-based framework, that allows for the automated generation of database UIs and application code, which is based on a combination of task-, context-, and domain model. As the different models have different roles, agents are used for the code generation as well.

Pruvost and *Bellik* [16] present a framework for multi-modal interaction in ubiquitous systems. The framework is a part of the *European ATRACO* project[2]. One interesting aspect of this work is that agents negotiate on how to render the MBUIs.

Braubach et al. [4] impressively demonstrate the capabilities of merging MBUIs with agent systems, although their approach was not intended for agent systems in the first place. As a result to this design decision, their architectural presentation model lacks depth. In our opinion, the used presentation model does not provide enough information. Hence, it has to be extended with modality-dependent informations, which leads to one UI descriptions for each supported modality. Further, although *Vesuf* is an interpreter-based MBUID environment, there is no context-model available. Hence, the UIs cannot adapt to the actual context-of-use at runtime. The other examinees focus on particular aspects and disregard the bigger picture of a holistic user experience. Nevertheless, *Pruvost* and *Bellik* present interesting ideas, which gives us visions for future extensions of our work, as for instance agents which negotiate about the most adequate way of interaction.

To sum up, our survey shows, that although there are many approaches to develop MBUIs, only a few of them have been applied and tested in conjunction with agent systems. Yet, it is our believe, that a multi-agent systems and MBUIs is a promising combination for AAL environments.

3 Approach

As mentioned above, the development of multi-modal UIs is a complex task. Interpreter-based MBUID can be used as it counters many difficulties and also suits well for the realm of AAL. Based on our survey we can state that current MBUI technology has, as yet, not found its way into the agent domain and vice versa. It is

[1] MedPAge (Medical Path Agents), see `http://vsis-www.informatik.uni-hamburg.de/projects/medpage`

[2] Adaptive and Trusted Ambient Ecologies, see `www.uni-ulm.de/in/atraco`

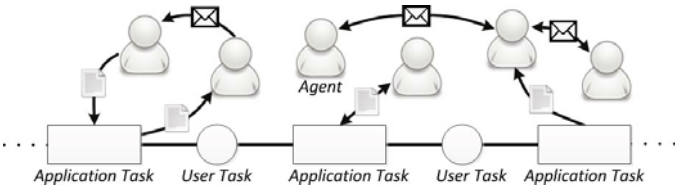

Fig. 1 Abstract illustration of the approach, enabling agents to perform the application tasks of the task-model.

our objective to narrow the gap between both technologies and in the following we describe our way to achieve this goal. We start by outlining the target system and proceed by getting granular on our approach. Subsequently, we introduce applied technologies and finally, we argue on how the presented systems works together and fosters the interplay between MBUIs and multi-agent system technology.

3.1 AOSE meets MBUID

MBUI development applies several models in order to ensure device independence, multi-modal interaction and context-awareness. Each model encapsulates particular information on some part of the application as a whole. Runtime systems interpret these models and derive UIs which are optimised for a given execution context.

However, although there are several different models available, one is involved in the majority of MBUID environments – the task-model [5]. The task-model formalises the general workflow[3] of the application and distinct between tasks of the user and tasks that belong to the application's logic. Task-models can be described by using many languages, and reach from static to dynamic and executable ones.

Agents on the other hand are usually compelled to some application goal and manage the application's logic accordingly. In order to enable MBUIs for multi-agent systems we have to ensure that the application's tasks can be interpreted and performed by the agents. Figure 1 illustrates the principle.

In the example, the task-model is represented as a chain of application- and user tasks. Whenever the runtime system detects an application task, a referenced back-end service should be called – in our case an agent. Further, required data should be forwarded to the agent, yet, as MBUIs and multi-agent systems are usually based on different technologies and have different conflicting properties (straight definition vs. degrees of freedom), this task is not easily accomplished. In order to foster communication between MBUIs and multi-agent systems, we developed the *Human Agent Interface* [2] (HAI). HAI was designed to facilitate the integration of user interface technologies into agent applications. During runtime, HAI acts as a gateway between UIs and MAS, hiding particular UI details from the agent application and vice versa. Thus, to convert and deliver UI messages to the agent world and to forward responses from the agent application back to the user interfaces is HAI's

[3] A workflow is considered to be the tasks that can be reached.

main purpose. We designed HAI to be independent from any specific UI technology and also as extension to the Model-View-Controller (MVC) architecture. Due to its characteristics, HAI constitutes a suitable foundation for the problem we address in this work.

3.2 From Theory to Practice

Before presenting a system which takes AOSE and MBUID into account we want to provide a short outline of the applied technologies. Although HAI is not restricted to a particular agent framework, we frequently used HAI in combination with the *Java-based Intelligent Agents Componentware* [10] (JIAC V). JIAC V is a Java based agent framework which has been developed at the Technical University of Berlin since 1998. It combines service-oriented with agent-oriented concepts and offers conformity to FIPA standards[4].

Using model-based development to implement user interfaces provides many advantages. Nevertheless, as an objective of our work, we want to prevent UI- and agent developers from affecting each other. In order to do so, we applied the *Multi-Access Service Platform* [3] (MASP), as it allows model-based development and clearly distinct between UI and application. MASP task-models are based on the widely accepted *ConcurTaskTree* notation [13] (CTT). CTT separates task-models into four types of tasks: User tasks, application tasks, interaction tasks and abstract tasks. User and interaction tasks are performed by the user. Application tasks are executed by the system and abstract tasks are complex actions which can not be expressed by the other ones. In order to provide information on the execution sequence (e.g. parallel, step-by-step) and interdependencies between them, the tasks are ordered by means of LOTUS operators. Although CTT is a good foundation for user-centric design, it neglects some requirements for AAL sceneries and had to be extended in some aspects [9]. To start with, AAL environments – especially those with agents – continuously collect sensor data and may identify situations in which user interaction tasks have to be triggered or disabled. Classical CTT do not support this kind of behaviour and therefore prevents proactive agents to adjust the user interface to the latest set of environmental data.

Figure 2 illustrates the architecture of the implemented system including MASP, HAI and the MAS. Once an application task occurs, the additional backend service is executed. In order to assure that agents are able to manage the respective application tasks, we have implemented the *HAIService*, which manages the mapping process. After the HAIService was called a HAI service message is generated and send to the HAI (1). This message contains additional data (e.g. name, required capabilities or supported input/output parameters) and an identifier for the designated agent, which is used by HAI to establish a permanent connection to the responding agent for further UI requests. Subsequently, HAI converts the UI message into an agent message (in compliance with the FIPA ACL standard) and forwards the message to the agent system (2). The agent system now processes the incoming message

[4] FIPA – The Foundation for Intelligent Physical Agents – see `www.fipa.org`

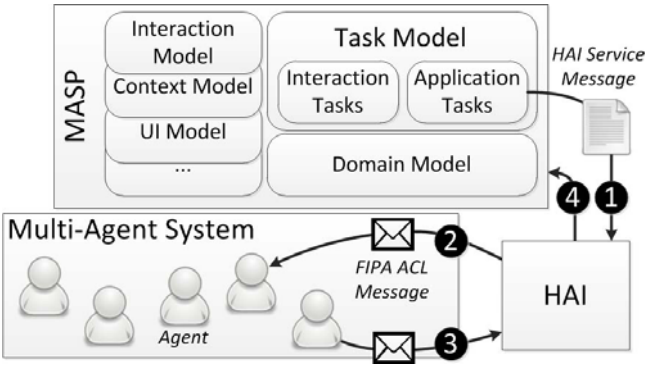

Fig. 2 Architecture of the system, consisting of three top-level tiers: MASP, HAI and the MAS.

and responds by a message as well (3). As a result, HAI notifies the interpreter that the task has been accomplished and updates the data model if necessary (4). Finally, the runtime system of MASP forces the view to apply to the potential changes.

4 Proof of Concept

As we strongly believe, that a combination of MBUID and AOSE facilitates the development of AAL environments, we proceed by describing an agent-based AAL service which we developed within the *SmartSenior* project[5]. As security is an inherent part of the AAL vision we recently implemented an agent-based assistant which is able to detect anomalies in an AAL environment and to inform a user about them by different interaction modalities [12]. The assistant comprises three logical components: sensor-, analysis- and reaction unit. While the sensor unit for itself is not agent-based, the others are. The analysis unit is a multi-agent system consisting of a situation recognition agent, that collects sensor data, a rule evaluation agent and an anomaly detection agent to manage situations which are not covered by rules. While the rule evaluation agent reacts on same event types always in the same fashion, the anomaly detection agent is able to learn a user's regular behaviour and reacts when discrepancies to this regular behaviour are identified using statistical methods. Once an event occurs, it is forwarded to the reaction component. The reaction component is an agent that determines the best interaction possibility to inform the user. This process is straightforward and makes only use of the localisation data gathered from the sensor unit. MASP was utilised to develop and furnish three different UIs [17]. Once the user is close to a screen, a popup will provide informations and options about an occurring event. In case the user is not in scope of a screen but at home, a bracelet the user wears informs about the event. Finally,

[5] SmartSenior – longer independence for senior citizens, see `www.smart-senior.de`

in case the user is currently not at home, the users smart phone is used as interaction device.

5 Conclusion

In this paper, we introduced an approach that facilitates the development of device- and modality independent UIs for agent applications. In order to do this, we argued that MBUID is a suitable foundation for the implementation of agent applications for AAL environments. Furthermore, we emphasised that this area of research is only sparely covered and that existing solutions have severe shortcomings, on either the agent- or on the MBUI side. We further described how the task-model, that is available in most MBUID environments, can be utilised to apply model driven techniques for the development of UIs for agents. Subsequently, we provided an outline of our example system, that makes use of MASP as MBUID environment and JIAC as agent framework. In order to exploit the MBUIs of the MASP for JIAC agent systems, we made use of HAI. Due to HAIs independence from specific UI technologies or agent frameworks, our approach is easily adaptable to other MBUID environments and agent frameworks. After presenting our approach, we illustrated a proof-of-concept implementation of an agent-based AAL service.

It is our opinion that, as sophisticated model driven UI techniques are, the support for the dynamics and the capabilities of distributed multi-agent systems for AAL environments often fall short. To counter this issue, we extended the CTT in a way that allows agents to act proactively. We also used the MVC paradigm to strictly separate between agent and UI specific parts. Nevertheless, currently there are unsolved issues regarding our approach. While agent technology offers capabilities for coordinated and orchestrated the increasing number of applications in AAL environments, a mechanism is required to ensure a reasonable UI. Hence, a first step is to find a more sophisticated possibility to negotiate about the best way of interaction for a given situation, even for the whole AAL environment.

Evaluating design-oriented approaches is usually a tedious task. In the future, we want to demonstrate that a combination of MBUID and AOSE facilitates the development of AAL environments. In order to measure the impact on the development effort of AAL services, we intent to compare different developer teams with equal capabilities performing the same task – some using the presented approach, the others not. We will present the results of this evaluation in a succeeding work.

References

1. Agent Oriented Software Pty. Ltd.: JACK Intelligent Agents – WebBot Manual, 5.3 edn. Agent Oriented Software Pty. Ltd., Victoria, Australia (2009)
2. Ahrndt, S., Lützenberger, M., Heßler, A., Albayrak, S.: HAI – A Human Agent Interface for JIAC. In: Klügl, F., Ossowski, S. (eds.) MATES 2011. LNCS, vol. 6973, pp. 149–156. Springer, Heidelberg (2011)

3. Blumendorf, M., Feuerstack, S., Albayrak, S.: Multimodal user interfaces for smart environments: The multi-access service platform. In: Bottoni, P., Levialdi, S. (eds.) Proceedings of the Working Conference on Advanced Visual Interfaces. ACM (2008)
4. Braubach, L., Pokahr, A., Moldt, D., Bartelt, A., Lamersdorf, W.: Tool-supported interpreter-based user interface architecture for ubiquitous computing. In: P. Forbrig, Q. Limbourg, B. Urban, J. Vanderdonckt (eds.) Interactive Systems - Design, Specification, and Verification, pp. 89–103. Springer (2002)
5. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A unifying reference framework for multi-target user interfaces. Interacting with Computers 15(3), 289–308 (2003)
6. Corchado, J.M., Perez, J.B., Hallenborg, K., Golinska, P., Corchuelo, R. (eds.): Workshop on Agents for Ambient Assisted Living. AISC, vol. 90. Springer, Heidelberg (2011)
7. Dutton, W.H., Blank, G.: Next generation users: The internet in britain. Oxford Internet Institute, University of Oxford (2011)
8. Eisenstein, J., Rich, C.: Agents and guis from task models. In: Proceedings of the 7th International Conference on Intelligent User Interfaces, pp. 47–54. ACM (2002)
9. Feuerstack, S., Blumendorf, M., Albayrak, S.: Prototyping of Multimodal Interactions for Smart Environments Based on Task Models. In: Mühlhäuser, M., Ferscha, A., Aitenbichler, E. (eds.) AmI 2007 Workshops, CCIS, vol. 11, pp. 139–146. Springer, Heidelberg (2008)
10. Hirsch, B., Konnerth, T., Heßler, A.: Merging agents and services – the JIAC agent platform. In: Bordini, R.H., Dastani, M., Dix, J., Amal, E.F.S. (eds.) Multi-Agent Programming: Languages, Tools and Applications, pp. 159–185. Springer, Heidelberg (2009)
11. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V.: USIXML: A Language Supporting Multi-path Development of User Interfaces. In: Bastide, R., Palanque, P., Roth, J. (eds.) DSV-IS 2004 and EHCI 2004. LNCS, vol. 3425, pp. 200–220. Springer, Heidelberg (2005)
12. Mustafić, T., Clausen, J., Messerman, A., Chinnow, J.: Concept of a sensor based emergency detection in a home environment. In: Ambient Assisted Living 2010, p. 4. VDE Verlag (2010)
13. Paterno, F., Mancini, C., Meniconi, S.: Concurtasktrees: A diagrammatic notation for specifying task models. In: Howard, S., Hammond, J., Lindgaard, G. (eds.) Proceedings of Interact 1997. Human-Computer Interaction Conference. Chapman and Hall (1997)
14. Paterno, F., Santoro, C., Spano, L.D.: MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. ACM Transactions on Computer-Human Interaction (TOCHI) 16(4), 1–30 (2009)
15. Pokahr, A., Braubach, L.: The Webbridge Framework for Building Web-Based Agent Applications. In: Dastani, M., El Fallah Seghrouchni, A., Leite, J., Torroni, P. (eds.) LADS 2007. LNCS (LNAI), vol. 5118, pp. 173–190. Springer, Heidelberg (2008)
16. Pruvost, G., Bellik, Y.: Ambient multimodal human-computer interaction. In: Proceedings of the Poster Session at The European Future Technologies Conference, pp. 1–2 (2009)
17. Raddatz, K., Schmidt, A.D., Thiele, A., Chinnow, J., Grunnewald, D., Albayrak, S.: Sensor-based detection and reaction in ambient environments. In: Ambient Assisted Living 2012. VDE Verlag (to appear, 2012)
18. Tran, V., Kolp, M., Vanderdonckt, J., Wautelet, Y., Faulkner, S.: Agent-Based User Interface Generation from Combined Task, Context and Domain Models. In: England, D., Palanque, P., Vanderdonckt, J., Wild, P.J. (eds.) TAMODIA 2009. LNCS, vol. 5963, pp. 146–161. Springer, Heidelberg (2010)