CASi – A Generic Context Awareness Simulator for Ambient Systems

Jörg Cassens, Felix Schmitt, Tobias Mende, and Michael Herczeg

Institute for Multimedia and Interactive Systems, University of Lübeck,
Ratzeburger Allee 160, D–23562 Lübeck, Germany
{cassens,schmitt,herczeg}@imis.uni-luebeck.de

Abstract. In this paper, we present CASi (Context Awareness Simulator), a software system for the simulation of context-aware computer systems and environments. CASi provides an abstract framework of components for simulating smart world applications like a smart office or house with ambient sensors and actuators. Agents moving through these application worlds are tracked by sensors and their actions are influenced by actuators, both of which can be programmed to resemble the actual peripherals of the tested system. CASi allows testing ambient, context aware computer systems even in early stages of development without the need for expensive prototyping or real world deployment.

Keywords: Ubiquitous computing, ambient intelligence, context, simulation.

1 Introduction

When designing and building ambient, context aware computer systems, extensive testing will be necessary due to the complexity, the distributed nature and the richness of human interaction of such systems. Deployment of a prototype to its actual target domain is one obvious way to handle this phase of the development process. This approach, however, has severe drawbacks: The deployment of the system consumes time during which development resources are occupied, and the provision of actual sensor and actuator hardware is expensive. Furthermore, human subjects will need some time until they interact with such a system in a natural way. Changes to the system design and even regular development iterations require updates to the system's peripherals and their installation. These problems can be mitigated by implementing virtual sensors, actuators and users in a simulation environment.

MACK [11] is a framework developed by our research group on top of which application systems for different domains can be built. Key requirements for a simulating environment to be used with the development of MACK are:

- applicability across different domains,
- adaptability to new system architectures,
- options to run simulations faster than real time, and
- platform independence.

F. Paternò et al. (Eds.): AmI 2012, LNCS 7683, pp. 421–426, 2012.

In the following sections, we first review and discuss other existing solutions, pointing out their respective advantages and shortcomings. Afterwards, we introduce CASi's design and architecture by describing its key components and the rationale behind it. At the end, first evaluation results are presented before we point out the roadmap for future development of CASi.

2 Related Work

Of particular interest in the field of ubiquitous systems is a special class of deterministic, discrete simulators known as object-oriented simulators [9] and here in particular individual-based or agent-oriented simulators [1]. Basically, an agent-based simulation will be comprised of different simulated actors and their interaction with each other and the world. Instead of describing the probably very complex behavior of the system as a whole, individual agents are described in simpler terms. These agents act simultaneously, and the behavior of the system as a whole is seen as an emergent phenomenon. This approach has been chosen for the CASi simulator. Other approaches and partial solutions for the requirements described can be found:

Siafu [6] is a simulation tool that implements many of the characteristics we demanded in the previous section. The system offers a graphical user interface for visualization purposes which makes it easy for human users to monitor the simulation. Siafu's architecture is rather limited with regard to adaptability and generalizability. Another issue with Siafu is the missing support for plug-in components modeling sensors and actuators. Siafu also requires advanced programming skills not only for implementing new types of sensors, actuators, and protocols, but even for the definition of simulations themselves.

DiaSim [3,4] forms the simulation part of a bigger environment for developing pervasive computing applications. Other components include the DiaSpec language for describing functionality of sensors and actuators in simulated as well as real world systems, and the DiaGen component, which generates an outline of the to-be-written program code for the actual implementation of simulated and real-world components.

In DiaSim, entities like agents, sensors, and actuators are not modeled on an individual level. It uses mathematical density functions to model the probability of persons being present in a particular area of the simulated world and to determine probabilities of relevant actions with regard to the modeled system. The areas from which simulated sensors may record data or across which simulated actuators may convey their output are also described mathematically in an indirect way.

CASS [8] is mainly targeted at detecting rule conflicts in rule-based AI systems for smart-home applications. It focuses on one particular element of ambient intelligent systems, namely a specific kind of reasoning subsystem.

ISS [7], the Interactive Smart Home Simulator, not only models the environment, its inhabitants and various sensors and actuators. Its focus lies on the simulation of home appliances for smart home environments. ISS includes the reasoning components within the simulator itself. Since the reasoners trying to derive context from sensor inputs form a crucial part of any ambient intelligent system, we believe that

close ties between simulator and AI lead to reduced flexibility with regard to setting up different testing scenarios.

Similar to ISS, **CAST** [5] also focuses on smart home scenarios. Within its domain CAST aims at simulating data exchange between an ambient system's components not on a functional, but rather on a network and security engineering level.

3 CASi: Concept and Vision

Our development of CASi, the Context Awareness Simulator, is grounded in our work on MACK [11] and MATe [10]. MACK (Modular Awareness Construction Kit) is a framework for constructing ambient intelligent computer systems. MATe (MATe for Awareness in Teams) is an application system built on top of MACK. It aims at supporting team members in knowledge-intensive work environments by fostering awareness of their colleagues' activities and interruptibility status.

One key feature of MACK and MATe is a message passing infrastructure that connects the different actuators and sensors with a hub that does the actual processing, deriving contexts and instigating context-sensitive behavior. Another key feature is the flexible and extensible reasoning subsystem, which can make use of different knowledge representations and reasoning paradigms.

Due to the perceived limitations and different foci of the existing solutions outlined in the previous section, we decided to develop our own simulator. The main focus was set on modularity and encapsulation, since the framework character of MACK demands great flexibility with regard to application domains and, thus, changes in (simulated) peripherals, logics, and communication infrastructure. Generic interfaces between CASi's components also allow modifications of the simulation engine itself.

4 Design and Architecture

In order to develop a highly customizable simulator, we designed an architecture which avoids close bindings between the components. The simulator has been implemented in Java since it is flexible, wide-spread, and platform independent.

The world is a bundle of objects which are needed to describe an environment. The most important element in the world object is a collection of rooms (or, more generally, spaces). Rooms form the map on which agents are able to perform actions. A room is enclosed by multiple walls, which can contain doors. The world also contains a collection of agents and a collection of components like sensors and actuators. It is also possible to embed custom objects. Sensors, actuators, and custom objects can be positioned anywhere on the map, e.g. in any room. Sensors can only monitor a restricted area. Actions and agents outside this area do not influence the component. Actuators in turn cannot influence agents which are outside of their reach.

In CASi, agents are virtual persons who interact with each other and with sensors, actuators and other elements in the simulation independently. They can be used to trigger events in sensors and can be influenced by actuators. Every agent has its own action handler scheduling actions that should be performed next. Actions can be

defined with a deadline by which they have to be completed, an earliest start time, duration and a priority. Different action handlers are able to use these parameters to select the most relevant next action for an agent. This results in a dynamic goal stack that guides an agent's activity.

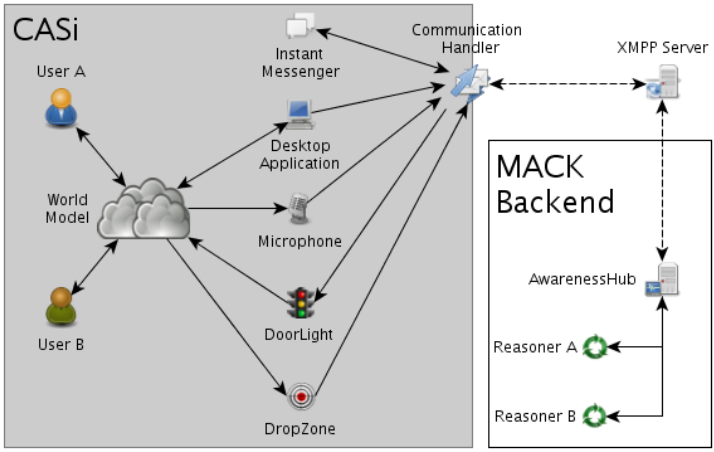

Fig. 1. CASi interfacing with a MACK-based system. CASi simulates the different agents as well as sensors and actuators of the MACK frontend whereas context processing takes place in the actually installed backend. Solid lines denote API calls, dashed lines XMPP connections.

In general, the concept for action handlers defines three different collections. The to-do list contains actions that should be performed once. The real world equivalents are meetings and special tasks. The second collection is a pool containing actions to be performed several times. An example could be short coffee breaks. Furthermore, action handlers may contain an interrupt list containing actions that have to be performed at the next opportunity. This can be used for synchronizing agents, e.g., when they perform an action that involves other agents, like talking. The agent who starts an action schedules another action on the other agent's interrupt list. This prevents the other agent from continuing with other actions before the joint action is completed.

Actions can be postponed for a later execution, e.g. if an agents has to talk to another agent, it may be not allowed to interrupt the agent if it is occupied. The first agent can then schedule the action for a later try.

The communication handler represents connections to external systems, e.g. a network interface which connects to a context middleware or the context aware application itself. It sends sensor values, and receives new values for actuators. Currently, we use the XMPP-based MACK protocol for communication between the simulated sensors and actuators and the MACK backend which contains, amongst other things, reasoners for different contextual aspects (like interruptibility, user activity).

A simple GUI acts as a visual representation of model states. In addition, the GUI allows scaling the time in the simulation and the entire simulation can be paused. Agents, sensors and actuators can be selected to display their state.

An example simulation based on MATe has been developed in order to test modifications and extensions to the architecture on the fly. This simulation models a typical office scenario and consists of several individual office rooms, a meeting room, a coffee kitchen, and restrooms (see Fig. 2). Agents move freely according to their current action list. Virtual sensors and actuators are deployed mimicking a real installation of MATe. Agents have a DropZone on their desk to put a personal token into, a Desktop Activity Analyzer reports on open applications and typing frequency, audio sensors detect the number of persons talking in a room, the state of doors is sensed, and an interactive device called the Cube allows the users to give corrective feedback to the system's reasoners. All these sensory inputs are sent to the MATe system's AwarenessHub and its reasoners (see Fig. 1). The resulting output is forwarded to the simulation again, so the agents are confronted with changing states of actuators like the DoorLight, which signals an office occupant's interruptibility to others, and the Cube, which shows the activity recognized by MATe on its top surface. These actuatory outputs influence the agents' behavior, e.g., if an agent has a talk-to action first on its list, but its counterpart's DoorLight signals non-interruptibility, the agent will postpone the action until later and resume with the next action on its list.

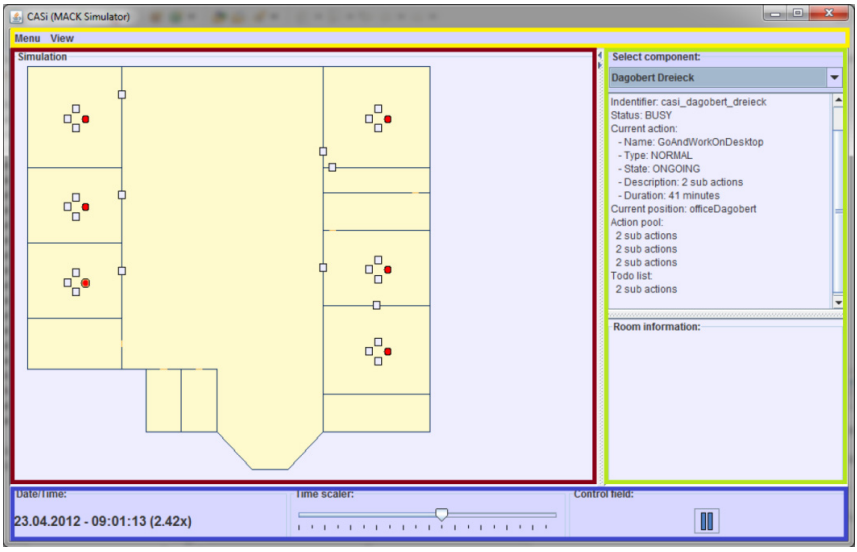

Fig. 2. CASi's SimpleUI visualization interface. Note the detailed information about one agent in the side bar on the right as well as the time scaling slider and the pause button on the bottom.

In addition to the example simulation, which tests the simulator against a known working system, we also ran tests after new components had been added to the system under test. For example, a new case-based reasoner (CBR) for user activities was added to MATe. Simulations were run which included agent activities modeled after results from a field study. Examination of the knowledge base and the system output showed that the reasoner worked as intended. The simulation took only minutes per run, similar data acquisition would have taken half a day each in the real-world.

5 Conclusion

In this paper, we have presented an individual-based simulator for context aware and ambient intelligent systems. While the simulator is customized to interface with the MACK framework, it lends itself to adaptation for other flavors of context middleware. First tests within an ambient application environment have shown the usefulness of CASi for rapid testing of new reasoning components and for testing the effects of adding or removing sensors. In the future, CASi will be tested in further ambient applications and it will be deployed as a modular component to be included in other ambient systems development environments.

References

1. Bonabeau, E.: Agent-based modeling: Methods and techniques for simulating human systems. Proceedings of the National Academy of Sciences (2002)
2. Bratman, M.E.: Intention, Plans, and Practical Reason. CSLI Publications, Stanford (1999)
3. Bruneau, J., Jouve, W., Consel, C.: DiaSim: A Parameterized Simulator for Pervasive Computing Applications. In: Proceedings of the 6th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, IEEE (2009)
4. Jouve, W., Bruneau, J., Consel, C.: DiaSim: A parameterized simulator for pervasive computing applications. In: Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications, pp. 1–3. IEEE Computer Society (2009)
5. Kim, I., Park, H., Noh, B., Lee, Y., Lee, S., Lee, H.: Design and Implementation of Context-Awareness Simulation Toolkit for Context learning. In: Proceedings of the IEEE In-ternational Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing. Workshops, vol. 2, pp. 96–103. IEEE Computer Society (2006)
6. Martin, M., Nurmi, P.: A Generic Large Scale Simulator for Ubiquitous Computing. In: Annual International Conference on Mobile and Ubiquitous Systems, pp. 1–3 (2006)
7. Nguyen, T.V., Nguyen, H.A., Choi, D.: Development of a Context Aware Virtual Smart Home Simulator. CoRR abs/1007.1274 (2010)
8. Park, J., Moon, M., Hwang, S., Yeom, K.: CASS: A Context-Aware Simulation System for Smart Home. In: Proceedings of the 5th ACIS International Conference on Software Engineering Research, Management & Applications, pp. 461–467. IEEE Computer Society (2007)
9. Roberts, C.A., Dessouky, Y.M.: An Overview of Object-Oriented Simulation. Simulation 70, 359–368 (1998)
10. Ruge, L., Kindsmüller, M.C., Cassens, J., Herczeg, M.: How About a MATe for Awareness in Teams? In: Proceedings of the Seventh International Workshop on Modelling and Reasoning in Context, pp. 58–69 (2011)
11. Schmitt, F., Cassens, J., Kindsmüller, M.C., Herczeg, M.: Mental Models of Ambient Systems: A Modular Research Framework. In: Beigl, M., Christiansen, H., Roth-Berghofer, T.R., Kofod-Petersen, A., Coventry, K.R., Schmidtke, H.R. (eds.) CONTEXT 2011. LNCS, vol. 6967, pp. 278–291. Springer, Heidelberg (2011)