A Multi-Agent Approach to Professional Software Engineering

Marco Lützenberger, Tobias Küster, Thomas Konnerth, Alexander Thiele, Nils Masuch, Axel Heßler, Jan Keiser, Michael Burkhardt, Silvan Kaiser, Jakob Tonn, Michael Kaisers, and Sahin Albayrak
firstname.lastname@dai-labor.de

Technische Universität Berlin, DAI-Labor

Abstract. The community of agent researchers and engineers has produced a number of interesting and mature results. However, agent technology is still not widely adopted by industrial software developers or software companies—possibly because existing frameworks are infused with academic premises that rarely apply to industrial settings. In this paper, we analyse the requirements of current industry-driven software projects and show how we are able to cope with these requirements in the Java Intelligent Agent Componentware agent framework, JIAC V. We argue that the lack of industry-grade requirements and features in other agent frameworks is one of the reasons for the slow acceptance of agent technology in the software industry. The JIAC V framework tries to bridge that gap—not as a final solution, but as a stepping stone towards industrial acceptance.

Keywords: Agent Framework, MAS Development, Industrial Adoption

1 Introduction

The concept of Agent Oriented Software Engineering, or AOSE, dates back as far as 1997, when Michael Wooldridge published his seminal article [40] and established an entirely new branch of research. Fifteen years later, we face numerous theories, methodologies, tools and frameworks, each supporting the development of agent-based software applications in the one or the other aspect. Yet, despite intensifying research on AOSE, it is far from being audacious to say that the agent community has as yet failed to convince the industry to adopt their ideas [1]. The problem of agent-technology to gain foothold in industrial processes has been discussed vigorously [29,33,38,39] and some reasons for this problem were already identified. These reasons include poor awareness of industrial needs, disconnection from conventional software engineering, immature technology, and a focus on research issues that are not necessarily required by industry [39].

The Java Intelligent Agent Componentware (JIAC) was developed under the premise to narrow the discrepancy between research and industry. In this paper,

we present the fifth incarnation of our agent framework, namely JIAC V [17]. We are well aware that, after fifteen years of research, the enthusiasm for 'yet another agent framework' can only be moderate. Yet, JIAC does not fall into this category. We argue that—as opposed to well known and established agent frameworks—JIAC was neither explicitly developed as a research framework (cf. Jason [5]), nor streamlined towards the requirements of individual industrial stakeholders (cf. JADE [3]). JIAC was developed under the premise to cover a wide spectrum of requirements and to further the industrial adoption process. The development was focused on the objective to provide a robust communication infrastructure, even beyond the borders of homogeneous computer networks. The modular assembly of JIAC agents allows for multi-agent system (MAS) solutions that are tailored to the application context. JIAC currently offers modules for migration, rule interpretation, persistence, scripting languages, load measurement, OSGi-integration and human-agent communication, to name but a few.

We do not consider JIAC to be an ultimate solution, but rather a step towards industrial acceptance. We argue that, so far, state-of-the-art frameworks were not able to convince the industry of the elegance of the agent paradigm and alternative approaches are strongly required should AOSE ever gain foothold in industrial processes.

We begin this paper with a brief description of different projects in which JIAC was used and respectively mention features that were required for a successful appliance (Section 2). Based on this analysis, we examine the capabilities of well established agent frameworks to deal with the previously collected requirements (Section 3). We use this analysis to substantiate our thesis, that certain features are not sufficiently covered in state-of-the-art approaches. We proceed by presenting the JIAC framework in more detail (Section 4), including a description of standard features, the most relevant extension as well as development tools. Subsequently, we describe selected appliances of JIAC in more detail and respectively underline the technical integration of required framework features (Section 5). Finally, we discuss the role of JIAC within the pool of well established agent frameworks and the agent community and wrap up with a conclusion (Section 6).

2 The Case of the JIAC V Framework

In this section we compile a list of requirements that were derived from application projects which we concluded over the last couple of years. Following Weyns at al. [39], there are many reasons that hamper industrial adoption of multi-agent technology. These reasons include poor awareness of industrial needs, disconnection from conventional software engineering, immature technology and a focus on research issues that are not necessarily required by industry [39].

To avoid these pitfalls we used industry-funded projects to develop and extend JIAC and discussed our intentions with our industry partners. In the following we give a short description of the projects in which JIAC V was used

and extended towards the requirements of the project and the project partners. We also emphasise the different domains in which the framework was used and conclude this section by explaining the requirements that arose from these very applications and guided the development of JIAC V.

2.1 Project Summaries

The goal of the *Service Centric Home* (**SerCHo**) project was the development of an open service platform that increases life quality at home. The platform was intended to support the quick and easy delivery of new context sensitive services into the home environment and the provisioning of a consistent user interface for these services. In this project we extended JIAC's capabilities to deal with the service metaphor. As such, we have focused on developing a service engineering methodology and tool suite as well as a service delivery platform to simplify service development, deployment and maintenance [15].

The focus of the *Multi Access – Modular Services*, or **MAMS+** project was to allow non-technical persons to fast and easily create, deploy and manage services, according to the users' needs. We have developed a service delivery platform based on our multi-agent framework [35]. The platform integrates modern technologies like IMS/SIP, allows for service composition and features service matching, load-balancing and self-healing mechanisms, to name but a few.

Within the project *Gesteuertes Laden V2.0* (**GL V2.0**, Managed Charging V2.0) [37] the goal was to develop a decentralised intelligent energy management system that uses electric vehicle's batteries as mobile energy storages. The purpose of the developed planning algorithms was to stabilise the energy grid and to maximise the amount of renewable energy within the electric vehicles (EVs) in dependence to forecasts of available wind energy.

To maintain a good standard of living for senior citizens, new technologies have been developed within the **SmartSenior** project. Our work included sensor-based situation -detection, -reaction, -notification and remote management [34]. During a field study the solutions have been successfully installed and tested in the home environments of more than thirty elderly participants.

The project *Energy Efficiency Controlling in the Automotive Industry*, or **EnEffCo**, aims at the implementation of a modular software system [22] to simulate operational modes of plant sections with relevant energy consumption. The software serves as a tool for decision makers in manufacturing, to whom it offers the identification and evaluation of strategies and tactics for establishing cost- and energy-efficient production schedules.

Intelligent Solutions for Protecting Interdependent Critical Infrastructures (**ILIas**) is a project aimed towards developing intelligent solutions for protecting critical infrastructures that provide electricity and telecommunication services to the general public [21]. These solutions need to be scalable and reconcile the need for fast automated reaction with manual supervision for highly critical decisions. Software solutions and protection mechanism efficiencies in large-scale networks are evaluated using simulated disaster scenarios. The simulation models

are supplemented by a hardware test laboratory where exemplary interdependent energy and telecommunication infrastructures are set up.

The on-going project **BeMobility 2.0** investigates the integration of electric vehicles (EVs) into urban transport and energy networks. In addition to the development of concepts that will combine different mobility services (e.g., vehicles, public transportation, etc.), an energy management system [11] for a Micro Smart Grid is being developed, in which a variety of system components, such as EVs, charging infrastructure, and energy sources, are taken into account.

The aim of the *Connected Living* project is to provide a system for integrating and managing 'smart devices' in future home environments: **CL-OS**. Besides providing a layer of abstraction for controlling devices by diverse vendors, another goal is to supply an infrastructure for developing, publishing and deploying agents or coalition of agents to the users' home environments to help future home user to achieve its goals.

The objective of the project *Extensible Architecture and Service Infrastructure for Cloud-aware Software*, or **EASI Clouds**, is to provide a comprehensive and easy-to-use cloud computing infrastructure with support for cloud interoperability and federation. The infrastructure includes advanced SLA (Service Level Agreement) management for all service layers, facilities for capacity planning, heterogeneous provisioning as well as accounting and billing.

The Multi-Agent Programming Contest is an annual competition that started in 2005 [2]. The contest is an attempt to stimulate research in the field of programming multi-agent system. Our team has been participating since 2007. We use it as a platform to teach students in the field of agent based design and implementation using the JIAC V agent framework.

2.2 Requirements Derived from the Projects

During the process of domain analysis and system design of those projects, several requirements have been identified and hence fulfilled in JIAC V. While many of them are typical for industrial or business software frameworks, it is our believe that a multi-agent framework does not have to stand behind.

In many projects the results had to be tested during field trials or user rollouts. Applications had to be running for months without problems. Therefore, *stability and robustness* are key issues for a good user experience. A certain level of robustness was important, especially in dynamic environments, e.g., *deploying and undeploying* new services or agents should not affect other parts of the application. The same holds true in a distributed context, e.g., when new nodes join or leave a system or agents *migrate* between them. The framework has to be able to handle a potentially *large number of agents and agent nodes* without a decrease in performance, a requirement especially affecting communication infrastructure and the distributed service directory. Several projects dealt with service delivery and management of services, resulting in various requirements like *support for service life-cycle*, *management interfaces*, *runtime deployment* and *third-party service integration*. Additional requirements related to management and adaptive behaviour are *monitoring and introspection*. It has to be possible to retrieve

status information from all framework components in a standardised way. Certain functionalities were required in multiple projects so that *component reuse* became necessary. Additionally, the framework needed to *be extensible* in order to be able to integrate future requirements.

Industrial adoption of agent technology was already discussed in a couple of surveys. These surveys conclude that technical maturity and coherence [39], connections to regular software engineering techniques [38], a comprehensive tool support [33] as well as robustness, feasibility and flexibility [29] are indispensably required for industrial adoption of multi-agent technology. We identified similar requirements and developed JIAC to counter well known problems of industrial adoption and to demonstrate that an agent framework is able to meet industrial requirements.

We proceed this work by emphasising conflicts between the capabilities of other agent frameworks and the above-mentioned requirements for industrial acceptance.

3 State of the Art

When we compare the requirements of our projects and those for a successful adoption of agent technology to existing agent frameworks, the results are twofold. On the one hand, platforms like Jason [4] or 3APL [14] have been created on a very strong theoretical background. They feature elaborate implementations of cognitive concepts that are important for the agent research agenda as a whole. On the other hand, even though there have been approaches to extend these frameworks—e.g., JASDL [20] for Jason—they were never intended to be used in an industrial context but geared towards research.

It is difficult to use these frameworks when implementing real world applications that are supposed to run for days, or even weeks. Consider the Gesteuertes Laden V2.0 project as an example for such application (see Section 2 for a summary and also Section 5.1 for more details). In this project, we were supposed to deploy autonomous decision-making software on the hardware of an electric vehicle. At worse, a system crash may have disabled the charging functionality of the vehicle—a non-tolerable situation, especially for the driver. Once a driver has picked a vehicle, it is difficult to access its software and to reboot or update malfunctioning agents. As such, we had to ensure a reliable operation. Academic agent frameworks, however, were not developed under the objective to ensure a reliable operation throughout days or weeks. To be clear about this, it is not our intention to overly criticise the above mentioned frameworks but to emphasise that those frameworks can not be used for the implementation of applications where system crashes cause non-tolerable situations for (test-)users.

More pragmatic approaches such as JADE [3] or the JACK [8] framework are more focused on the engineering and development aspects of applications. The JADE framework in particular has a long list of extensions and additions, such as the Web Service Integration Gateway [12], AgentOWL [24], WADE [9] or the MASE framework [31]. On a point by point basis these extensions seem

to fulfil many of our requirements. Nevertheless, we still think it is difficult to use these frameworks for our purpose as most extensions have been developed independently from each other. Using them within the same software project will be tedious if not impossible. In order to convince software developers of the agent paradigm a comprehensive programming framework is required. One can not expect them to collect required extensions before any implementation can be done. Especially the JADE framework with all of its extensions lacks the coherence and unity that we would expect from a modern software framework.

A current agent architecture that tries to bridge the gap between agent technology and the software industry is the Jadex framework [32]. The developers of Jadex have taken a number of approaches to improve Jadex in ways that make it more compatible with industry standards [6]. For instance, cloud concepts were integrated [7], an active component architecture was realised [6] and it is possible to define workflows by means of BPMN (business process modeling notation) [6].

However, while we appreciate the approach to adapt the framework to industry needs, we find that a number of design decisions do not comply with the requirements for our projects. The decision to base the framework on the active component model—with agents as the internal architecture of the components—reverses the control architecture from our point of view. We regard agents as the surrounding structure and expect them to have capabilities that enable communication and interaction. Furthermore, the integration of ontologies and workflows into Jadex is insufficient for us. In Jadex, ontologies are transformed into ADF belief JavaBeans. This procedure decouples them from the actual ontologies on the web—as envisioned by OWL—and thus we consider it a proprietary approach. Workflows on the other hand should be one way to describe capabilities of an agent, not an agent type of their own. As a result, we found the Jadex model to be too different from our vision of software agents, and thus could not model our systems in the way we envision modern agent oriented technologies.

For the above reasons, existing agent frameworks either do not fulfil our requirements for practical applications, or their models are too different from our modelling approach for agent oriented applications. In the following we describe the JIAC framework, which represents our approach to an agent architecture that fulfils our needs.

4 The JIAC V Framework

JIAC V is a Java-based multi-agent development framework and runtime environment [27] that has been both developed and deployed in a number of application projects. Based on the requirements of those projects (see Section 2) particular emphasis has been placed on the following aspects:

- robustness, scalability, modularity and extensibility
- adoption of a service-oriented view and integration of third-party services, e.g., provided as web services and/or OSGi bundles
- dynamically adding and removing services, agents, and nodes at runtime

– extensive tool support, both at design time (modelling and development) and at runtime (management and monitoring)

In the following, we describe how these requirements were satisfied in JIAC.

4.1 Core Mechanisms of JIAC Agents

One of the core aspects of JIAC is the integration of agents with the Service Oriented Architecture paradigm (SOA) [16]. Using a powerful discovery and messaging infrastructure, JIAC agents can be distributed transparently, even beyond network boundaries. An agent-platform comprises one or more 'agent nodes' which are physically distributed and provide the runtime environment for JIAC agents (see Figure 1). New agents, services, as well as further agent nodes can be deployed at runtime. Agents can interact with each other by means of service invocation, by sending messages to individual agents or multicast-channels, and by complex interaction protocols. Each individual agent's knowledge is stored in a tuple-space based memory. Finally, JIAC agents can be remotely monitored and controlled at runtime via the Java Management Extension Standard (JMX).

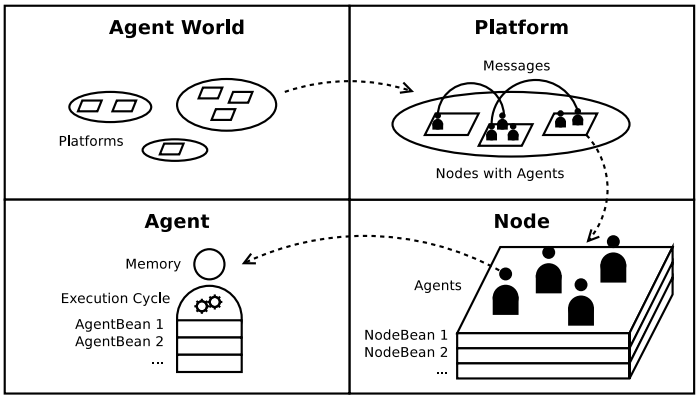

Fig. 1. Structural elements of a JIAC V multi-agent system.

Each agent contains a number of default components, such as an execution-cycle, a local memory and the communication adaptors. The agents' behaviours and capabilities are implemented in a number of so-called *AgentBeans*, which are controlled by the agent's life cycle. Each AgentBean may:

– implement a number of *life-cycle* methods, which are executed when the agent changes its life-cycle state, such as *initialized*, or *started*,
– implement an *execute*-method, which is called automatically at regular intervals once the agent is running (i.e., cyclic behaviour),
– attach *observers* to the agent's memory, being called for instance each time the agent receives a message or its world model is updated, and

– provide *action* methods, or services, which are exposed to the directory and can be invoked either from within the agent or by other agents.

Using these four mechanisms, it is possible to define all of the agents' capabilities and behaviours [18]. Furthermore, the structure of each agent contains a number of standard components, such as an execution-cycle, a local memory and the communication adaptors. The entire multi-agent system, i.e., which agent has which agent beans and how those agents are distributed to agent nodes, is then set up using one or more Spring[1] configuration files.

4.2 Default and Extension Components

JIAC agents contain a number of individual AgentBeans that are implemented as described above as well as a set of standard AgentBeans that constitute the basic capabilities of an agent. One such AgentBean each JIAC agent is equipped with by default is the *Communication Bean*. First, this component manages the inter-agent service communication; second, it allows the agents to exchange messages with other agents or groups of agents on the network, addressing individual agents or multi-casting to message channels. The messages are not restricted to FIPA messages but can have any data as payload.

Complementary to the AgentBeans, there are *NodeBeans*, adding functionality to the node as a whole. Each agent node is equipped with a *Directory NodeBean*, listing the actions of the different agents, and a *Message Broker NodeBean*, being the counterpart to the agent's communication bean and allowing them to transparently send messages from node to node using ActiveMQ.[2]

Other commonly used AgentBeans and NodeBeans can be added to a multiagent system by appending the respective bean to the agent's configuration. For the composition of services, JIAC includes an *Interpreter* AgentBean for the execution of the high-level service-oriented scripting language *JADL++* [17]. Reactive behaviour of agents can be enabled with a *Drools*[3] rule engine that can be synchronised with the agents' memory.

Extensions to the capabilities of nodes and agents include a *Migration* NodeBean, that enables strong agent migration between agent nodes, a *Persistence* NodeBean that saves the node configuration and allows for restarting the node later on, and NodeBeans for *Load Measurement* and *Load Balancing* that provide cross-node load information and distribute agents over nodes at start- and runtime. In order to support application development, JIAC also provides generic functionalities such as AgentBeans for *User Management*, *Human Agent Interfaces*, a *Webserver* NodeBean running an embedded Jetty-server, and a *Web Service Gateway* AgentBean that exposes JIAC actions as web services and vice versa. Last but not least, the *OSGi Gateway* allows JIAC nodes to be executed within an OSGi framework and to access other OSGi services.

[1] Spring: `http://www.springsource.org/`
[2] ActiveMQ: `http://activemq.apache.org/`
[3] JBoss Drools: `http://www.jboss.org/drools/`

4.3 Development Methods and Tools

Since JIAC is a Java-based agent framework, the bulk of the development work can be done using conventional Java development tools, such as Eclipse and Maven, as well as supportive tools like XML editors. Still, to improve the efficiency in application development, some additional tools are provided, all of which can be integrated directly into the Eclipse IDE.

A *JIAC Project Wizard* helps with creating new JIAC projects by generating a uniform project structure, including a configured Maven `pom.xml` file, listing the required dependencies, and a starter class for running the new JIAC application. Further, several Eclipse views provide information about nodes currently running on the network and the agents and services they contain, as well as the possibility to start and to interact with newly created agents and services.

JIAC agents can also be modelled using two high-level graphical editors: The *Visual Service Design Tool (VSDT)* and the *Agent World Editor (AWE)*. Using the VSDT, both the workflows of individual agents as well as their interactions can be modelled as a series of BPMN diagrams [30]. Based on those diagrams, executable JIAC AgentBeans or JADL++ services can be generated [23]. The AWE can be used to create visual representations of an entire multi-agent-system and its components, showing the different agents and agent nodes in a distributed system and the individual services and AgentBeans they provide. From these visual models, the tool can generate the corresponding Spring configuration files as well as JIAC AgentBean stubs [26].

Finally, the running multi-agent system can be monitored and manipulated using the *ASGARD* agent runtime monitor [36], providing a three-dimensional view of all agents running in the local network.

An overview of JIAC's core features and extensions, how they map to the requirements derived from the projects, and how they are used within those projects can be seen in Table 1.

5 Applications and Lessons Learned

Many features and components shown in the last section were developed as a consequence of project requirement analyses. The resulting modular structure of the JIAC system enables developers to tailor selected functionalities. In the following, we present industry projects from different domains—namely energy, electric mobility and health—and put emphasis on system engineering aspects.

5.1 Planning Electric Vehicle Charging Intervals

Description: The goal of Gesteuertes Laden V2.0 [37] was to use electric vehicles as mobile and distributed energy storages in order to utilise wind energy and to stabilise energy grids. As the driver's inherent needs for mobility are always the main objective to fulfil, a mechanism was needed that supports the users in planning charging- and feeding intervals without limiting their flexibility. Our solution was implemented as a live system including real EVs (three Mini-E vehicles provided by the project partner BMW) and charging stations.

Table 1. Project Requirements and use of JIAC V features in projects (bold: featured project, see Section 5). The letters next to the features indicate the several requirements the feature contributes to.

			SerCHo	MAMS+	**GL 2.0**	**Smart Senior**	**EnEffCo**	**ILIas**	BeMobility 2.0	CL-OS	EASI Clouds
Requirements	**R**obustness, Stability	R	x	x	x	x	x	x	x	x	x
	Hot **D**eployment	D	x	x	-	x	x	x	-	x	x
	Agent Migration	A	-	-	-	-	-	x	-	-	x
	Scalability	S	x	-	x	-	x	x	-	-	x
	Service **L**ife Cycle Mngmt.	L	x	x	-	x	-	-	-	x	-
	Third-party API integration	T	x	x	x	x	-	-	-	-	-
	Introspection, Monitoring	I	-	-	x	x	x	x	-	-	x
	Modularity, Reusability	M	x	x	x	x	x	x	x	x	x
JIAC Core	Distribution	R AS M	x	x	x	x	x	x	x	-	x
	Node Deployment	D S	-	x	x	-	x	-	-	-	x
	Services, SOA	LT M	x	x	x	x	-	x	x	x	x
	Message-based Comm.	R S	x	-	-	x	x	x	x	-	x
	Service Deployment	D L	-	x	-	x	-	x	-	x	-
	Agent Deployment	DAS	-	x	-	-	-	-	-	x	x
	Management & Monitoring	R I	-	x	x	x	x	x	x	x	x
JIAC Plugins	Webserver	T M	-	x	-	x	-	-	-	x	-
	Interpreter	D L M	-	x	-	x	-	x	-	-	-
	Rule Engine	D T M	x	-	-	x	-	x	-	x	-
	Migration	DAS	-	x	-	-	-	-	-	-	-
	Persistence	R L	-	x	-	x	-	-	-	x	x
	Load Balancing	RDASL	-	x	-	-	-	-	-	-	x
	Webservice Gateway	T	x	x	x	-	-	-	x	-	-
	OSGi Gateway	T M	x	-	-	x	-	-	-	-	x
	Human Agent Interface	I	-	x	x	-	x	-	x	-	-
	User Management	S	-	x	-	-	-	-	-	x	x

Implementation: In the project, a distributed mobility and energy management system was designed in which each of the involved actors—such as driver, vehicle manufacturer, energy provider, charging station, and grid operator—is represented by a software agent [19]. The system is able to create user-centric day schedules containing journeys, charging and discharging events [28] and takes into account actor-dependent preferences and constraints, such as the driver's appointments, wind forecasts, the EV, available charging stations, and energy grid constraints.

The developed system contains eight software agents in the back-end and three agents within each of the EVs (see Figure 2). More than 100 services are

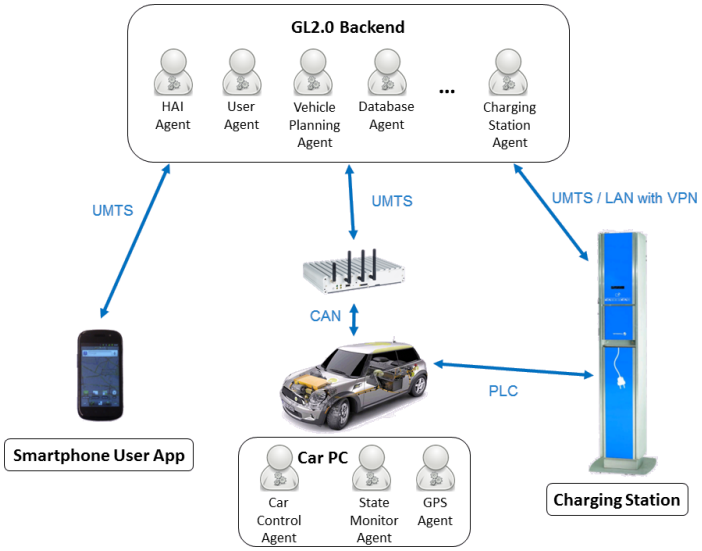

Fig. 2. GL V2.0 System Architecture.

running simultaneously, offering different tasks, ranging from simple information services to complex planning algorithms. For each user and each electric vehicle an additional agent representation is running in the back-end, taking the main responsibility for developing user and vehicle schedules.

The data exchange between EV and back-end agents is based on unreliable telecommunication networks (e.g., UMTS), therefore failover mechanisms ensure a reconnection after network stabilisation. The coordination of charging and feeding events is processed by the EV agents interacting with the charging stations via power line communication. Third party services such as wind forecasts and charging station status information were embedded into JIAC via the Web Service Gateway. Furthermore, a generic MySQL database agent has been developed. As the user interaction plays an important role, a smartphone application has been developed. The integration into JIAC was done with the Human Agent Interface.

Lessons Learned: The system was evaluated within a three-week field test [25]. During this time, the need for a maintainability component, which notifies the developers about the services' availability, became apparent and was subsequently developed and installed. Furthermore, the field test revealed, that the service advertisement messages that propose or refresh the existence of services in each of the local service directories aggregated to a significant amount of traffic. The communication exceeded the bandwidth of the UMTS connection that was used between the CarPC Nodes and Backend Nodes, thus, we decided to increase the service advertisement interval. Given that we already encountered performance issues when dealing with three electric vehicles, we also tried to assess

the performance of large-scale distributed systems comprising thousands of electric vehicles. In this case the adaption of the advertisement message rate might not be sufficient, since an extremely high interval would be necessary which in turn would have the drawback that the disappearance of a service would be recognised rather late. Two other possibilities were therefore discussed. On the one hand the concept of the service advertisement messages could be changed. In this case messages are only sent once a new node appears or disappears. However, this approach bears risks in unreliable communication infrastructures, thus we selected the second option, namely a more sophisticated group communication concept. Instead of using one group per node we relaxed this strict assumption and allow nodes to join multiple groups, but not to forward advertisement messages between those. According to our project scenario, the CarPC Nodes now only share a group with those nodes they are really interacting with. The advertisement messages of all other nodes are not being sent via the UMTS connection, which results in a considerably more effective communication.

5.2 Multi-Agent Systems for an Ageing Society

Description: An ageing society causes high costs for health care and services that can be addressed by modern IT. Also, elderly people can regain a higher level of quality of life when using such IT, since dedicated health care will only be required on few occasions. Such a system has been developed in the SmartSenior project, where the focus was set to covering most aspects of daily needs while keeping the system usable. It uses sensors, processors and effectors in order to detect situations at home for performing appropriate (re-)actions [34]. The system has been installed and tested during a field study in 32 apartments.

Implementation: In each apartment, two JIAC nodes containing several agents were running. Both nodes have been wrapped as OSGi-Bundles and installed in an OSGi-execution environment (Knopflerfish 3.1). Each agent was designed to perform a specific task, and the interaction among these agents resolved into a global system behaviour. Besides JIAC's communication and service invocation infrastructure, the system also makes use of the rule engine abstraction and service interpreter agents. While it is beyond the scope of this work to explain each agent in detail, an example will demonstrate the interaction and the features of the JIAC framework that have been used.

The main task of the system is to detect and react to specific situations, i.e., sets of specific sensor values in a certain time frame. The sensor values are aggregated and enriched with additional semantic data by a Sensor Agent. This information is then sent to both, a Database Agent, storing the values in a local MySQL database, and the Detection Agent, using the Drools rule engine to detect different situations in the sensor data. Those specific situations are then sent to the Reaction Agent, triggering the appropriate reaction in the form of a service, using the JADL++ interpreter. Finally, a number of different agents provide means for, e.g., displaying a histogram of sensor values to the user, sending out notifications, or integrating with other OSGi services.

The mapping between situations and reactions is modelled in BPMN [30] using the Visual Service Design Tool (see Section 4.3). The processes, which usually include a trigger for situations and a set of services to be executed, are transformed to JADL++ scripts [16] and deployed to the runtime using the distributed service directory and JIAC's JMX interface.

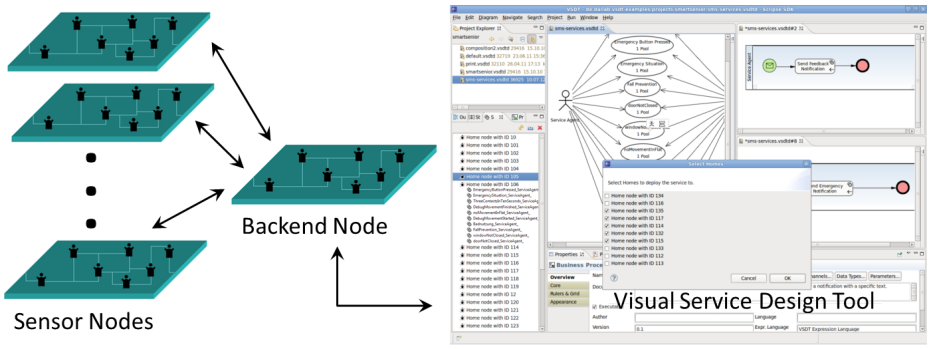

Fig. 3. SmartSenior global architecture and deployment chain.

In order to be able to remotely deploy (and delete) reaction scripts, each sensor node connects to the back-end node (see Figure 3). The connection needs to be static but fail-safe, therefore it is configured to reconnect every time a disconnect occurs. The back-end node functions as a gateway and provides services to retrieve all registered sensor nodes or to deploy and undeploy reaction-scripts to a specific sensor node. The VSDT connects to the back-end, and via several Eclipse views a user can choose which services are to be deployed or undeployed into which apartment.

Lessons Learned: During the eight weeks of the field study, the system was running stable in 32 apartments with no significant errors. Just once, a problem with already processed messages building up in the message queues required a restart of the system after the bug was being fixed. Apart from that, there were no further incidents and the entire system ran smoothly for the entire time.

5.3 Distributed Optimisation of Production Schedules

Description: The current rise in renewable energy production makes it necessary to either store large amounts of energy in between periods of high energy production, or to adapt energy consumption to energy production. This can be helped by day-ahead energy markets such as the European Energy Exchange AG (EEX), where prices for short-term energy procurement vary according to demand and supply. This provides an incentive to the industry to adapt their production schedules to the energy production that is reflected in the price predictions, e.g., by shifting energy-intense production activities to times with low

energy prices, or by filling storage areas with intermediate products when prices are low, to feed on them when energy is more expensive. The aim of the EnEffCo project was to provide a system for optimising production schedules in this regard.

Implementation: We developed an optimisation framework that takes as an input a production process model and an energy price curve and produces the optimal production schedule for that setup [22]. For modelling the various production activities and their dependencies, a very simple model is used: Similar to Petri nets, it consists primarily of activities (the individual production steps), and resources (the parts and products consumed and created by those activities). Using this simple model, a wide range of processes can be specified and optimised, from manufacturing processes to charging schedules for electric vehicles [11].

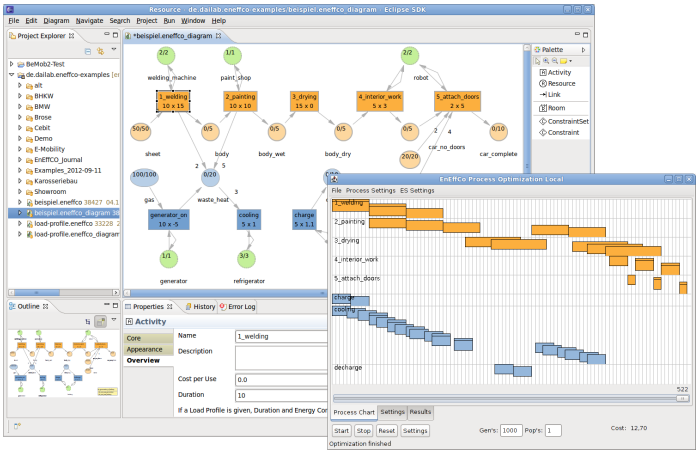

Fig. 4. EnEffCo Process Model Editor and Optimisation GUI.

Once the process has been modelled, it is sent to the actual optimisation framework (Figure 4). After creating an initial, naive production schedule, evolutionary algorithms are used for mutating and recombining those schedules, by randomly inserting, moving and removing activities, until a satisfactory result (w.r.t. overall energy consumption, energy price, etc.) is reached.

While the actual optimisation algorithm does not make use of the JIAC framework, it is used for transparently distributing and connecting the several components of the system. First, using a simple interaction protocol [22], client agents can distribute their optimisation jobs to any number of optimisation server agents. This way, several 'populations' can be optimised in parallel, improving the chances to find an optimal schedule without increasing the running time of the optimisation. Second, each of the other components of the EnEffCo

system, such as the process model editor, a Web frontend, or a database holding energy consumption data for different activities, are connected to each other using JIAC agents.

Lessons Learned: While the original plan was to have one agent for each production step, and/or for each (intermediate) product, and to have those agents negotiate when to produce what product, we soon came to the conclusion that it would be much more pragmatic and practical to implement the performance-critical optimisation algorithm in a traditional way and to use agent only at a much higher level, for distributing and integrating the several components. This way, the communication overhead could be reduced drastically.

5.4 Managing Cascading Failures in the Power Grid

Description: The main motivation behind the ILIas project is that modern infrastructures are interdependent, such as power and telecommunication (TC) grids. In case of failures, this can create cascading effects in several or all of the involved infrastructures. The objective of the project is to research and create intelligent and scalable management systems that provide prediction and reaction to cascading failure effects, so that actions to stabilise the managed infrastructure can be taken. An example for this is the reaction to power outages and the consequent failure of TC networks in the affected areas.

Implementation: The approach chosen in ILIas is an agent-based decentralised smart grid management system [10] that observes and controls the grid. Each smart grid entity is managed by a separate management agent that tries to maintain a best-as-possible state for its controlled entity and interacts with neighbouring entities' management agents to provide a balance between its own entities' requirements and overall goals such as (sub-)network stability. As real-world power and TC networks contain very large numbers of entities, scalability was highly important requirement. Using a decentralised management system fulfils the requirements of a scalable and flexible system, as processing load can be distributed over a large number of systems. In addition to that, decentralisation also provides a degree of overall stability, as single points of failure are avoided.

Prediction of grid behaviour is supplied by a simulation of power and TC networks. The management agents are able to interact with both physical as well as simulated smart grid entities, which allows for easy testing of even large scale systems. This agent-based simulation is implemented using the NeSSi^2 simulation framework [13] and is able to work both, offline, simulating a predefined scenario, as well as online, by using the current grid state as a starting point to calculate predictions. The human-machine interface is provided by a visual monitoring application (Figure 5) that visualises the smart grid topology.

Lessons Learned: As JIAC was already a very mature framework when the ILIas project started, the requirements of the project's objectives did not influence the development of JIAC significantly. Instead the system specification phase

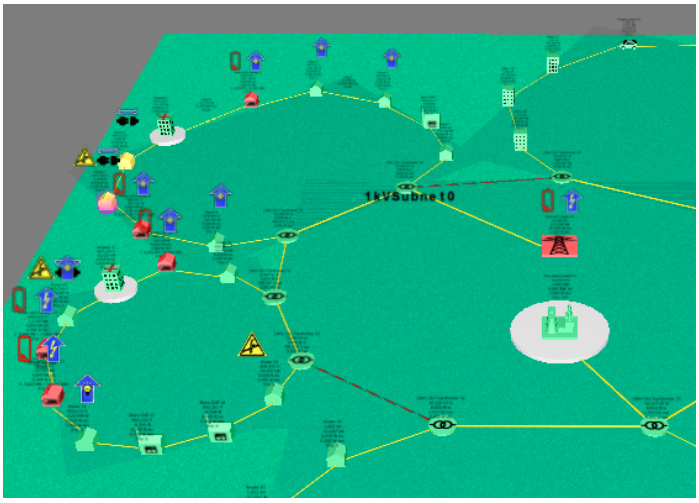

Fig. 5. Smart grid live monitoring in ILIas (image detail).

revealed that the features offered by JIAC were matching the requirements in ILIas quite good. For the p2p based approach in ILIas, the most helpful characteristics of the JIAC V Framework were the highly variable communication mechanisms, as well as the general agent based models used. The former were able to automatically adopt to changes in the infrastructure, e.g., when handling failure scenarios, the latter allowed very easy and quickly reconfigurable mappings of a given grid topology into a running management system. The resulting system proved to be very reliable in regard to overall system stability. The agent based models were also important for the required scalability of the system, as they provided the possibility to balance system load by distributing the agents over physically separated systems without any changes in the agent implementations.

During the development of the ILIas solution, two weak points in JIAC that require improvement were found. One of them was found in the lack of supported database services. Occasionally larger amounts of data had to be handled, which required database integration in the application development. As there was no generic solution for this provided by JIAC, an application-specific solution had to be implemented for the project. This functionality could be handled by the framework in the future.

The other field of improvement was the underlying communication infrastructure. While it provided the required flexibility and stability for most of the project's objectives, one weakness was identified in the behaviour with unstable connections, especially in scenarios where a connection would break away for a short time. An improvement for this could consists of better connection failure handling and improved reconnection times, to ensure basic functionality of the agents even without communication as well as minimising times of disconnection.

6 Conclusion

In this paper we presented the JIAC V agent framework. The basic idea behind JIAC was to provide an agent framework which meets industrial requirements and is able to facilitate the industrial adoption of the agent paradigm.

We started with a brief description of industrial projects in which JIAC is used and respectively emphasised features that were required for the technical realisation of each project. In doing so, we compiled a comprehensive list of requirements that includes: robustness, stability, hot deployment, agent migration, scalability, service life cycle management, third-party API integration, introspection and monitoring, modularity as well as reusability.

We compared the collected requirements to the capabilities of state-of-the-art framework solutions and thus motivated the necessity of JIAC. To be clear about this, we do not deem JIAC to be superior to other frameworks, though, some (well established) frameworks (e.g., Jason [4] or 3APL [14]) have been created on a very strong academic background and it has been argued that a focus on research issues is not necessarily required by industry and may even hamper industrial adoption [39]. Other frameworks that were geared towards industrial projects (e.g., the JADE framework [3]) were extended without an overarching concept and thus lack the coherence and unity that we would expect from a modern software framework, though, technical maturity and coherence were also identified as important factors on the way to industrial technology adoption [39].

After comparing the collected requirements to the capabilities of state-of-the-art framework solutions, we presented JIAC V in more detail, describing the architecture of JIAC multi-agent systems as well as the modular assembly of JIAC agents. Furthermore, we mentioned basic and extending capabilities of JIAC agents and nodes. Based on this description, we elaborated on selected projects, namely *GL 2.0*, *SmartSenior*, *EnEffCo*, and *ILIas* and emphasised the respective technical integration of required framework features.

As mentioned at the beginning, we certainly recognise that there are many agent frameworks available—each one with a focus on particular multi-agent system characteristics. Yet, as of today, the agent community was not able to convince industrial players to adopt their ideas. As opposed to comparable frameworks, JIAC was never intended to include the cutting edge of agent research but to constitute a robust, reliable, homogeneous and well-documented foundation for the development of agent-based software applications.

It was also our intention to equip JIAC with features that are generally required for extensive industrial appliances. Today, the JIAC framework provides a set of well-evaluated and useful capabilities. Common requirements, such as distribution or access to SOA-compliant services, were integrated as core functionalities. Other, less broadly used features, were developed as optional modules.

We do not consider JIAC to be an ultimate solution for the discrepancy between agent research and the applying industry. Yet, given the fact that JIAC was originally streamlined towards industrial projects and also towards ease of use, it is our opinion that JIAC has the potential to provide new incentives for

industrial stakeholders and users who are not all too familiar with the agent paradigm to consider agent technology.

References

1. Balke, T., Hirsch, B., Lützenberger, M.: Assessing agent applications — r&D vs. R&d. In: Ganzha, M., Jain, L.C. (eds.) Multiagent Systems and Applications — Volume 1: Practice and Experience, Intelligent Systems Reference Library, vol. 45, pp. 1–20. Springer Berlin / Heidelberg (2013)
2. Behrens, T., Köster, M., Schlesinger, F., Dix, J., Hübner, J.: The multi-agent programming contest 2011: A résumé. In: Dennis, L., Boissier, O., Bordini, R. (eds.) Programming Multi-Agent Systems, Lecture Notes in Computer Science, vol. 7217, pp. 155–172. Springer Berlin / Heidelberg (2012)
3. Bellifemine, F., Poggi, A., Rimassa, G.: JADE — A FIPA-compliant agent framework. Internal technical report, CSELT (1999), part of this report has been also published in Proceedings of PAAM 1999, London, April 1999, pp.97–108.
4. Bordini, R.H., Hübner, J.F., et al.: Jason: A Java Based AgentSpeak Interpreter Used with SACI for Multi-Agent Distribution over the Net. Website: `http://jason.sourceforge.net/Jason.pdf` (February 2007), last visited on 15.03.2013.
5. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-agent Systems in AgentSpeak Using Jason. Wiley Series in Agent Technology, Wiley-Blackwell (October 2007)
6. Braubach, L., Pokahr, A.: Addressing challenges of distributed systems using active components. In: Brazier, F., Nieuwenhuis, K., Pavlin, G., Warnier, M., Badica, C. (eds.) Intelligent Distributed Computing V, Studies in Computational Intelligence, vol. 382, pp. 141–151. Springer Berlin Heidelberg (2012)
7. Braubach, L., Pokahr, A.: Conceptual integration of agents with WSDL and RESTful web services. In: Dastani, M., Hübner, J.F., Logan, B. (eds.) Programming Multi-Agent Systems, Lecture Notes in Computer Science, vol. 7837, pp. 17–34. Springer Berlin Heidelberg (2013)
8. Busetta, P., Rönnquist, R., Hodgson, A., Lucas, A.: JACK — Components for intelligent agents in java. Tech. rep., Agent Oriented Software Pty, Ltd. (1999)
9. Caire, G., Gotta, D., Banzi, M.: WADE: A software platform to develop mission critical applications exploiting agents and workflows. In: Padgham, L., Parkes, D.C., Müller, J., Parsons, S. (eds.) Proceedings of the 7th international Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal. pp. 29–36. IFAAMAS (2008)
10. Chinnow, J., Tonn, J., Bsufka, K., Konnerth, T., Albayrak, S.: A tool set for the evaluation of security and reliability in smart grids. In: Cuellar, J. (ed.) Smart Grid Security, Lecture Notes in Computer Science, vol. 7823, pp. 45–57. Springer Berlin Heidelberg (2013)
11. Freund, D., Raab, A.F., Küster, T., Albayrak, S., Strunz, K.: Agent-based integration of an electric car sharing fleet into a smart distribution feeder. In: 3rd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies (ISGT Europe), Berlin, Germany. pp. 1–8. IEEE (October 2012)
12. Greenwood, D., Buhler, P., Reitbauer, A.: Web service discovery and composition using the web service integration gateway. In: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE 2005), Hong Kong, China. pp. 789–790. IEEE (2005)

13. Grunewald, D., Lützenberger, M., Chinnow, J., Bye, R., Bsufka, K., Albayrak, S.: Agent-based network security simulation (demonstration). In: Tumer, K., Yolum, P., Sonenberg, L., Stone, P. (eds.) Proceedings of the 10th International Joint Conference on Autonomous Agents and Multiagent Systems, Taipei, Taiwan. pp. 1325–1326 (Mai 2011)
14. Hindriks, K.V., Boer, F.S.D., der Hoek, W.V., Meyer, J.J.: Agent programming in 3APL. Autonomous Agents and Multi-Agent Systems 2(4), 357–401 (1999)
15. Hirsch, B., Konnerth, T., Hessler, A., Albayrak, S.: A serviceware framework for designing ambient services. In: Mana, A., Lotz, V. (eds.) Developing Ambient Intelligence (AmID'06). pp. 124–136. Springer France (2006)
16. Hirsch, B., Konnerth, T., Burkhardt, M., Albayrak, S.: Programming service oriented agents. In: Calisti, M., Dignum, F.P., Kowalczyk, R., Leymann, F., Unland, R. (eds.) Service-Oriented Architecture and (Multi-)Agent Systems Technology. No. 10021 in Dagstuhl Seminar Proceedings, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, Dagstuhl, Germany (2010)
17. Hirsch, B., Konnerth, T., Heßler, A.: Merging agents and services — The JIAC agent platform. In: Bordini, R.H., Dastani, M., Dix, J., Seghrouchni, A.E.F. (eds.) Multi-Agent Programming: Languages, Tools and Applications, pp. 159–185. Multiagent Systems, Artificial Societies, and Simulated Organizations, Springer (2009)
18. JIAC Development Team: JIAC — Java Intelligent Agent Componentware, Version 5.1.3. DAI-Labor, TU Berlin (October 2012), `http://www.jiac.de`
19. Keiser, J., Lützenberger, M., Masuch, N.: Agents cut emissions – On how a multiagent system contributes to a more sustainable energy consumption. Procedia Computer Science 10(0), 866–873 (August 2012)
20. Klapiscak, T., Bordini, R.H.: JASDL: A practical programming approach combining agent and semantic web technologies. In: Baldoni, M., Son, T., Riemsdijk, M., Winikoff, M. (eds.) Declarative Agent Languages and Technologies VI, Lecture Notes in Computer Science, vol. 5397, pp. 91–110. Springer Berlin Heidelberg (2009)
21. Konnerth, T., Chinnow, J., Kaiser, S., Grunewald, D., Bsufka, K., Albayrak, S.: Integration of simulations and MAS for smart grid management systems. In: Proceedings of the 3rd International Workshop on Agent Technologies for Energy Systems (ATES 2012), Valencia, Spain. pp. 51–58 (2012)
22. Küster, T., Lützenberger, M., Freund, D., Albayrak, S.: Distributed evolutionary optimisation for electricity price responsive manufacturing using multi-agent system technology. Int. Journal On Advances in Intelligent Systems 7(1&2) (2013)
23. Küster, T., Lützenberger, M., Heßler, A., Hirsch, B.: Integrating process modelling into multi-agent system engineering. Multiagent and Grid Systems 8(1), 105–124 (January 2012)
24. Laclavik, M., Babik, M., Balogh, Z., Hluchy, L.: AgentOWL: Semantic knowledge model and agent architecture. Computing and Informatics 25, 419–437 (2006)
25. Lützenberger, M., Keiser, J., Masuch, N., Albayrak, S.: Agent based assistance for electric vehicles — An evaluation. In: Huang, R., Ghorbani, A.A., Pasi, G., Yamaguchi, T., Yen, N.Y., Jin, B. (eds.) Active Media Technology, 8th International Conference, AMT 2012, Macau, China, December 2012, Proceedings, Lecture Notes in Computer Science, vol. 7669, pp. 145–154. Springer Berlin / Heidelberg (2012)
26. Lützenberger, M., Küster, T., Heßler, A., Hirsch, B.: Unifying JIAC agent development with AWE. In: Braubach, L., van der Hoek, W., Petta, P., Pokahr, A. (eds.) Multiagent System Technologies, 7th German Conference, MATES 2009, Hamburg, Germany, September 2009, Proceedings, Lecture Notes in Artificial Intelligence, vol. 5774, pp. 220–225. Springer Berlin / Heidelberg (2009)

27. Lützenberger, M., Küster, T., Konnerth, T., Thiele, A., Masuch, N., Heßler, A., Keiser, J., Burkhardt, M., Kaiser, S., Albayrak, S.: JIAC V — A MAS framework for industrial applications (extended abstract). In: Ito, T., Jonker, C., Gini, M., Shehory, O. (eds.) Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, Saint Paul, MN, USA (2013), to appear
28. Masuch, N., Keiser, J., Lützenberger, M., Albayrak, S.: Wind power-aware vehicle-to-grid algorithms for sustainable ev energy management systems. In: Proceedings of the IEEE International Electric Vehicle Conference, Greenville, SC, USA. pp. 1–7. IEEE (March 2012)
29. Mařík, V., McFarlane, D.: Industrial adoption of agent-based technologies. Intelligent Systems, IEEE 20(1), 27–35 (2005)
30. Object Management Group: Business process modeling notation (BPMN) version 1.2. Specification formal/2009-01-03, Object Management Group (January 2009)
31. Poggi, A., Tomaiuolo, M., Turci, P.: An agent-based service oriented architecture. In: Baldoni, M., Boccalatte, A., Paoli, F.D., Martelli, M., Mascardi, V. (eds.) WOA 2007: Dagli Oggetti agli Agenti. 8th AI*IA/TABOO Joint Workshop 'From Objects to Agents': Agents and Industry: Technological Applications of Software Agents, 24-25 September 2007, Genova, Italy. pp. 157–165. Seneca Edizioni Torino (2007)
32. Pokahr, A., Braubach, L., Jander, K.: Unifying agent and component concepts. In: Dix, J., Witteveen, C. (eds.) Multiagent System Technologies, 8th German Conference, MATES 2010, Leipzig, Germany, September 27-29, 2010, Lecture Notes in Computer Science, vol. 6251, pp. 100–112. Springer Berlin Heidelberg (2010)
33. Pěchoučyek, M., Mařík, V.: Industrial deployment of multi-agent technologies: review and selected case studies. Autonomous Agents and Multi-Agent Systems 17(3), 397–431 (2008)
34. Raddatz, K., Schmidt, A.D., Thiele, A., Chinnow, J., Grunewald, D., Albayrak, S.: Sensor-basierte Erkennung und Reaktion im häuslichen Umfeld. In: Proceedings of 5th German AAL congress 2012, Berlin, Germany. VDE Verlag (2012)
35. Thiele, A., Kaiser, S., Konnerth, T., Hirsch, B.: MAMS service framework. In: Kowalczyk, R., Vo, Q., Maamar, Z., Huhns, M. (eds.) Service-Oriented Computing: Agents, Semantics, and Engineering, Lecture Notes in Computer Science, vol. 5907, pp. 126–142. Springer Berlin Heidelberg (2009)
36. Tonn, J., Kaiser, S.: ASGARD — A graphical monitoring tool for distributed agent infrastructures. In: Demazeau, Y., Dignum, F., Corchado, J., Pérez, J. (eds.) Advances in Practical Applications of Agents and Multiagent Systems, Advances in Intelligent and Soft Computing, vol. 70, pp. 163–173. Springer Berlin Heidelberg (2010)
37. Vattenfall, BMW, TU Berlin, TU Chemnitz, TU Ilmenau: Increasing the effectiveness and efficiency of the applications wind-to-vehicle (W2V) and vehicle-to-grid (V2G) including charging infrastructure (Managed Charging V2.0). Technische Universitätsbibliothek Hannover (TIB) (2011)
38. Weyns, D., Helleboogh, A., Holvoet, T.: How to get multi-agent systems accepted in industry? International Journal of Agent-Oriented Software Engineering (IJAOSE) 3(4), 383–390 (May 2009)
39. Weyns, D., Van, H., Parunak, D., Shehory, O.: The future of software engineering and multi-agent systems. Special Issue on Future of Software Engineering and Multi-Agent Systems, International Journal of Agent-Oriented Software Engineering (IJAOSE) (2008)
40. Wooldridge, M.: Agent-based software engineering. IEE Proceedings — Software 144(1), 26–37 (1997)