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. Yet, given that the software paradigms which are currently employed by the software industry, such as service-oriented architectures or cloud computing, have much in common with agent-oriented software engineering, industrial software projects could greatly benefit from agent technology. In this paper, we make an approach to analyse the requirements of current industry-driven software projects and show how we were able to cope with these requirements in the JIAC V agent framework. 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.

1 Introduction

The concept of Agent Oriented Software Engineering, or AOSE, dates back as far as 1997, when Michael Wooldridge published his widely cited and influential article [34] and established an entirely new branch of research. Today—fifteen years later—there are countless theories, methodologies, tools and frameworks, each supporting the development of agent-based software applications in the one or the other aspect. Yet, despite the multitude of scientific works on AOSE, it is far from being audacious to say that research in agent technology has as yet failed to convince the industry to adopt their ideas [1]. The reasons for this are not entirely clear, but it is obvious that certain incentives are lacking [33].

In this paper we present the fifth iteration of the Java Intelligent Agent Componentware framework (JIAC) [14]. 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 premise to cover a wide spectrum of requirements and to further the industrial adoption process. Originated in the year 2002 and being in its fifth incarnation today, JIAC has reached technical maturity. The development was focused on the objective to provide a robust communication infrastructure. We implemented this infrastructure based on ActiveMQ and thus allow for a reliable inter-agent communication, even beyond the borders of homogeneous computer networks. Additional agent capabilities are encapsulated within reusable modules, namely AgentBeans which can be added as needed. This mechanism allows for multi-agent system (MAS) solutions which are tailored to the application context. Following this mechanism, JIAC currently offers modules for migration, rule interpretation, persistence, scripting languages, load measurement, OSGi-integration and human-agent communication, to name but a few.

Despite many practical appliances with convincing results, our experience is that JIAC is only marginally known within the agent community. As such, it is the aim of this paper to present the JIAC framework with particular emphasis on features that we regard to be beneficial for industrial projects. We do not consider JIAC to be an ultimate solution, but as 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 extensions 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 we derived from our application projects over the last couple of years. In order to achieve this, we first give a short description of the projects in which the JIAC V agent framework was used and developed. We further emphasise the different domains in which the framework was applied. We conclude this section with an explanation of 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. Within the project 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.

The technologies developed in the Multi Access – Modular Services (MAMS+) project 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 [30]. 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) [32] the goal was to develop a decentralised intelligent energy management system that utilises 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), while taking into account forecasts of available wind energy.

In order 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 and reaction as well as notification and remote management [29]. 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 [19] 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 [18]. 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 [9] 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 the several ‘smart devices’ in future home environments. Besides providing a layer of abstraction for controlling devices by diverse vendors, another goal is to supply an infrastructure for developing, publishing and deploying ‘assistants’ to the users’ home environments. In this context, an ‘assistant’ is a single agent or a coalition of agents that help a future home user to achieve his 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 will include 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 in agent based artificial intelligence that started in 2005 [2]. The contest is an attempt to stimulate research in the field of programming multi-agent system. Our team participated 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 un-deploying 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 had 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.
3 State of the Art

When we compared the requirements of our projects to existing agent frameworks, the results were twofold. On the one hand, platforms like Jason [4] or 3APL [12] 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. Even though there have been approaches to extend these frameworks—e.g. JASDL [17] for Jason—they were never intended to be industry-ready but geared towards research. As such, they do not fulfil our requirements when implementing large scale projects.

More pragmatic approaches such as JADE [3] or the JACK [7] framework are more focused on the engineering and development aspects of applications. However, development of the JACK framework seems to have stopped. The JADE framework on the other hand has a long list of extensions and additions, such as the Web Service Integration Gateway [10], AgentOWL [21], WADE [8] or the MASE framework [27]. On a point-by-point basis these extensions seem to fulfil many of our requirements. However, most of these extensions have been developed independently from each other and using them within the same software project will be tedious if not impossible. Thus, while many of the approaches are well thought-out and useful, the JADE framework with 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 [28]. The developers of Jadex have recently taken a number of approaches to improve Jadex in ways that make it more compatible with industry standards [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 typical 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.

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 [24] 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 provided e.g. 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 run time (management and monitoring)

In the following, we describe the JIAC framework in detail, and how those requirements are satisfied.

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) [13]. Using a powerful discovery and messaging infrastructure, JIAC agents can be distributed transparently over the network, or 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. 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). For an overview of how those individual features were used in projects, refer to Section 5 and Table 1 in Section 6.

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. AgentBeans support very flexible activation schemes: A bean may be executed at regular intervals or according to a life-cycle change, such as initialised, or started. Further, AgentBeans can attach observers to the agent’s memory, being called for instance each time the agent receives a message or updates its world model. AgentBeans also provide action methods, which are exposed to the directory and invoked either within the agent or by other agents.

Using these four mechanisms, it is possible to define all of the agents’ capabilities and behaviours [15]. 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 over the 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

\(^1\) Spring: http://www.springframework.org/
basic interior 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 by default 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 multi-agent system by simply adding the 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++ [14]. 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.

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 are integrated directly into the Eclipse IDE.

A special JIAC Project Wizard helps with creating new JIAC projects by generating a uniform project structure, including a readily 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

\(^2\) ActiveMQ: http://activemq.apache.org/

\(^3\) JBoss Drools: http://www.jboss.org/drools/
contain, as well as the possibility to start and to interact with newly created agents and services.

Complementary to those basic development tools and utilities, JIAC agents can be modelled using two high-level graphical editors: The Visual Service Design Tool (VSDT) and the Agent World Editor (AWE). Using the VSDT [20], both the workflows of individual agents as well as their interactions can be modelled as a series of BPMN [26] diagrams. Based on those diagrams, executable JIAC AgentBeans or JADL++ services can be generated. The AWE [23] can be used to create visual representations of entire multi-agent-systems 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 and JIAC AgentBean stubs for those systems.

Finally, the running multi-agent system can be monitored and manipulated using the ASGARD agent runtime monitor [31], providing an intuitive, three-dimensional view of all agent nodes and agents running in the local network and the communication between them.

5 Featured Projects

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

5.1 ILIas

The main motivation behind the ILIas project is that modern infrastructures are interdependent, such as power and telecommunication 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 telecommunication networks in the affected areas.

The approach chosen in ILIas is an agent-based decentralised smart grid management system, which observes and controls the grid. Prediction of grid behaviour is supplied by a simulation of power and telecommunication 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 simulation is also agent-based and implemented using the NeSSi² simulation framework [11]. The simulation is able to work both, offline, simulating a pre-defined 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 1) that visualises the smart grid topology.
When developing the p2p based approach in ILIas the most helpful characteristics of the JIAC V framework where the highly variable communication mechanisms, as well as the general agent based models used. The former where 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 with regard to the overall system stability.

Area of improvement 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. This functionality could be handled by the framework in the future.

5.2 Gesteuertes Laden V2.0

The goal of Gesteuertes Laden V2.0 [32] was to use electric vehicles as mobile and distributed energy storages in order to utilise the potential of fluctuating wind energy and to stabilise the 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 their charging and potential discharging events without limiting them in their flexibility. The solution to this is implemented as a live system including real EVs (three Mini-E vehicles provided by the project partner BMW) and charging stations.
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 [16]. The main contribution of the system is the creation of user-centric day schedules containing journeys, charging and discharging events [25]. In this context many different actor-dependent preferences and constraints had to be considered, such as the driver’s appointments (necessary for deduction of mobility needs), wind forecast, characteristics and current state of the EV, characteristics and availability of charging stations, and energy grid constraints.

The developed system features a high degree of complexity containing eight software agents in the back-end and three agents within each of the EVs (see Figure 2). Altogether more than 100 services are 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 are used, ensuring a reconnection after network stabilisation. The coordination of charging and discharging events is processed by the EV agents interacting with the charging stations via power line communication. In order to get all relevant information for the planning procedure, third party services such as wind forecasts and charging station status information were embedded into JIAC via the Web Service Gateway. Further a generic database agent has been developed providing access to MySQL databases. As the user interaction plays a very im-
important role in this scenario, a smartphone application has been developed. The integration into JIAC was done with the Human Agent Interface. The system was evaluated within a three-week field test [22]. During this time, the need for a monitoring component, which notifies the developers about the services’ availability, became apparent and was subsequently installed. Furthermore, the field test revealed, that the service advertisement messages between the CarPC Nodes and Backend Nodes aggregated to a significant amount of traffic, which became too high for a traffic limited connection, such as UMTS. As a consequence, the service advertisement interval has been increased.

5.3 SmartSenior

An ageing society causes high costs for health care and services which can be addressed by modern IT. Especially 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 was developed in the SmartSenior project, where the focus was to cover most aspects of daily needs while keeping the system usable. The system uses sensors, processors and effectors in order to detect situations at home for performing appropriate (re-)actions [29]. The system as described below has been installed and tested during a field study in 32 apartments. During the eight weeks of the study the system was running stable with no significant errors.

In each apartment two JIAC nodes containing several agents were running. Both nodes were wrapped as OSGi-Bundles and installed in an OSGi-execution environment (Knopflerfish 3.1). The system at the home side contained the following agents: OSGi-Proxy Agent, WebServer Agent, Sensor Agent, SensorDetection Agent, UPnPSensor Agent, Detection Agent, Reaction Agent, Database Agent, ReactionDeployment Agent, Notification Agent. Each agent was designed to perform a specific task. The interaction among these agents resolved into a global system behaviour. The interaction heavily used JIAC’s communication mechanisms, service invocations as well as interpreter services provided by the ReactionDeployment Agent. 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 specific situations and react in a predefined manner. A situation is defined as a set of specific sensor values in a certain time frame. Sensors send their information to a gateway and eventually the SensorDetection Agent receives these values and checks whether they are from the right home. Sensor values are send via internal communication to the Sensor Agent, which enriches the original information with additional semantic data, such as sensor type, time, room type and apartment id. This information is send to both the Database and Detection Agent, again using internal communication. The Database Agent includes a hibernate wrapper and stores sensor information in a local MySQL Database. It also provides actions to read from the database. These are later used by the WebServer Agent to display a
histogram of sensor values to the user. The Detection Agent stores the same sensory information in its knowledge base. It contains a Drools rule engine which is used to detect different situations. Each detected situation is passed to the ReactionDeployment Agent, which eventually triggers certain reactions, which in turn are provided by the Reaction Agent. First, the mapping between situations and reactions is modelled in BPMN [26] using the Visual Service Design Tool (VSDT, see Section 4.3). The model includes a trigger for reactions and a set of services to be executed. The mapping model is transformed into a JADL++ script [13] and deployed at runtime into the ReactionDeployment Agent using the distributed service directory and JIAC’s JMX interface (see below). Second, the ReactionDeployment Agent executes the right JADL++ script according to the detected situation by using the JADL++ interpreter agent bean. During script execution the previously specified reaction services are invoked. Finally, the Reaction Agent executes these services, e.g. sending notifications to the user. These are send via the Notification Agent, which is also able to receive notification request from other non-JIAC system components via OSGi-Events using the OSGi-Proxy Agent.

![SmartSenior global architecture and deployment chain.](image)

In order to be able to remotely deploy (and delete) reaction scripts, each sensor node (see Figure 3) connects to the back-end node. 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 un-deploy reaction-scripts to a specific sensor node. The VSDT connects to the back-end, allowing for use of the gateway services. Via several Eclipse views a user can choose which services are to be deployed or un-deployed into which apartment.
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. As such, we initiated the paper with a brief description of industrial projects in which JIAC was used, and emphasised the features that were required for the technical realisation of each project. 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. We further mentioned basic and extending capabilities of JIAC agents and nodes. Based on this description we elaborated on selected projects (namely ILIas, Gesteuertes Laden V2.0 and SmartSenior) and emphasised the respective technical integration of required framework features.

A comprehensive overview of implemented features as well as their appliances within industrial projects is given in Table 1.

<table>
<thead>
<tr>
<th>Table 1. Usage of JIAC V features in projects.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SerCHo</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Core</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Plugins</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

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 which 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 which are not all too familiar with the agent paradigm, to consider agent technology.

References


32. 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 Universitätshälibibliothek Hannover (TIB) (2011)
