Using Multi-Agent Systems to Simulate Dynamic Infrastructures Populated with Large Numbers of Web Service Enabled Devices

Stamatis Karnouskos
SAP Research
Vincenz-Priessnitz-Str 1, D-76131, Karlsruhe, Germany
Email: stamatis.karnouskos@sap.com

Mian Mohammad Junaid Tariq
SAP Research
Vincenz-Priessnitz-Str 1, D-76131, Karlsruhe, Germany
Email: mian.mohammad.junaid.tariq@sap.com

***Abstract*—A common practice in enterprise environments is the usage of Service Oriented Architectures (SOA) which implies heavy usage of web services. Lately the trend is to extend these approaches down to the embedded device layer i.e. have devices running web services natively and offer their functionality as a service. The Internet of Things (IoT) envisions millions of such heterogeneous, usually mobile, devices communicating with each other creating a very dynamic infrastructure. In this paper we concentrate on our efforts to use a multi-agent system to simulate this emerging dynamic infrastructure populated with large number of web-service enabled devices.**

I. Introduction

The last years we have witnessed two major trends with respect to the device world. On the one hand hardware is getting cheaper, smaller and more capable. According to the Internet of Things vision (IoT), the majority of the devices will have communication and computation capabilities, which they will use to connect, interact and cooperate with their surrounding environment. On the other hand, the software industry is moving towards service-oriented approaches and especially in the business world new complex applications are based on the composition and collaboration of other services. The Internet of Sevices vision (IoS) assumes this on a large scale where services rely on different layers e.g. enterprise, network, or even at item level. Web service enabled devices glue IoT and IoS, by providing their functionality as a web service, in an interoperable way that can be used by other entities such as enterprise applications or even other devices.

The trend shows that in the future, a much more diversified infrastructure will emerge, and the way we interact with it will change significantly. As depicted in Fig. 1, a mash-up of services will be created, that can be combined and used in a cross-layer way. Enterprise applications will be able to connect directly if needed to devices, without the use of proprietary drivers, while non-web-service enabled devices can still be attached and their functionality wrapped by gateways or at middleware layer. Peer to peer communication among the devices will push SOA concepts down to device layer and create new opportunities for functionality discovery and collaboration.

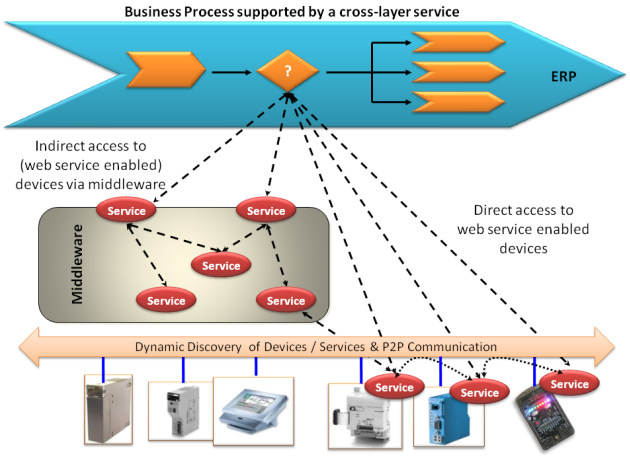

Fig. 1. A cross-layer web-service mashup

Web services are suitable and capable of running natively on embedded devices, providing an interoperability layer and easy coupling with other components in highly heterogeneous shop-floors. Device Profile for Web Services (DPWS [7]) and OPC UA [8] are emerging technologies for realizing web service enabled controllers and devices. Several projects such as SIRENA (www.sirena-itea.org), SODA (www.soda-itea.org) and SOCRADES (www.socrades.eu) provide a platform to develop a DPWS stack targeting the industrial automation devices on the shop floor [1].

Integration of the devices on the functional level allows us to focus on orchestrating services based on their role in a process, and not the device per se. Devices can host a variety of services needed to discover, understand their functionality and integrate them. Collaboration is enhanced among the entities which leads to the minimization of islands of heterogeneous devices and boosts interoperability. However the whole approach poses some significant challenges as well. Understanding the semantics as well as evaluating them against a specific context is needed while communicating in a vertical way across different layers as well as in a horizontal

one on the same level.

II. Simulation requirements

Networked embedded devices offering their functionality in a service oriented way e.g. via web services, are expected to be mainstream in the mid-term. Several companies in automation domain such as Beckhoff, Schneider Electric, Siemens, ABB etc already have or plan to release in the near term devices that feature web service access to their functionality (in the simplest versions) and/or dynamically allow their existence and services to be discovered and used. As envisioned in the Internet of Things, the population of such devices could be in a very bigger magnitude as the one most applications are designed to interact with and handle today. Especially with the introduction of IPv6 for devices such as the 6lowpan, an application might be directly IP-connected to thousands or millions of ubiquitous devices. This is also the vision of the newly formed IPSO Alliance (www.ipso-alliance.org) that promotes an IP network of smart objects. It is clear that scalability as well as performance will be key issues, apart from security, privacy and dependability. This new infrastructure will need to be modelled and many applications will need to be trialed out to see what implementation or design changes are required. Traditionally Enterprise applications are heavyweight (lightweight approaches usually running on embedded devices are acting as proxies to the back-end systems) and not designed to be scalable at the order of magnitude of services this new infrastructure introduces. Considering also that Enterprise applications evolve slower and need to go through extensive checks related to their performance and reliability, we urgently need a way to test them, identify the shortcuts and enhance them in order to be able to take full advantage of the Internet of Things.

To that extend some basic requirements arise:

A. Dynamic Discovery of Devices and Services

Web service devices will provide their functionality as a set of one or more services. Although for static devices a static reference would be enough, we focus on mobile devices that are expected to be the majority of future web-service enabled devices. As such it is important that these devices are discoverable in a dynamic way by their neighbours. Furthermore their services should also by dynamically discovered and used, as we expect that they will change/adapt over time.

B. Roaming

A mobile device roams various networks, governed by different entities and depicting different capabilities/functionality. First of all it should be possible to simulate this roaming behaviour. We distinguish two cases here

- The device changes location but stays in the same domain/network segment
- The device changes location and domain/network segment

Of course these can be further broken down based on additional criteria e.g. security aspects governing the different network segments that the device roams, but this is out of the scope of this paper. Additionally, it should be possible to track the device and use its services in a transparent way while it is on the move. This implies the need for a contact point that can reliably offer information about the current location of a device so that it can be used, or alternatively act as a mediator, hiding the true location of the device but still make it possible to interact with it.

C. Volatile devices

In a highly dynamic infrastructure devices may change between online and offline status multiple times. This can be a result of a device malfunctioning, a network break, a denial of service attack, or simply part of the device's lifecycle management (e.g. to save energy). As an example a wireless sensor node collecting temperature measurements has a very weak and unstable link, eventually depicting a volatile behaviour by being (randomly) online and offline. It should be possible to simulate this behaviour.

D. Cooperation capabilities

The first generation efforts in the Internet of Things focuses simply on giving devices the ability to expose their functionality in a service oriented way. However, subsequently efforts will advance towards creating more complex behaviors by composing cross-layered services and adopting cooperative techniques. As such we will see swarms of devices cooperating for common goals. The simulator should provide the capability of creating such cooperation scenarios.

E. Heterogeneity

The Internet of Things envisions a highly heterogeneous infrastructure. As such the distinct aspects of each device should be depicted also in the simulation, meaning that it should be possible to create a wide variety of devices with respect to the heterogeneity of them. Therefore any characteristics of a web service enabled device should be simulatable.

F. Large scale

The Internet of Things foresees very large scale infrastructures, therefore any simulation effort should in theory be scalable to accommodate large numbers of such devices. The architectural design of the simulator should not put limits on this, while of course the real boundaries with respect to resources available on the hosting machine should be trialed.

G. Complex Devices Simulation

Although some devices may be pretty simple (e.g. a wireless sensor node), some others are much more complex possibly composed of hundreds of other smaller devices. It should be possible to simulate highly complex physical devices as well as be flexible and adaptable to changes that might occur in the hardware or software of these devices. As such, the simulator should allow easy extensibility of simulated devices to better depict the reality or wished behaviour. These complex devices can either be real or virtual e.g. composed of many others to investigate their behaviour in a test setup.

H. Real and virtual data sources

The source of data reported by the simulated devices should be configurable and possibly coming from

- Real data sources: We consider here the real physical devices whose values are propagated in a timely manner to the simulated device. The data can be used by the simulated device on an one-to-one basis or after computing of them e.g. temperature values from -5°C to +40°C can be normalized in the -60°C up to +80°C range.
- Virtual data sources: We consider here any virtual source of data e.g. a stream of values stored in a database, an algorithm generating the required etc.

I. Transparent device simulation

The simulated device should acquire the majority of the characteristics of the device it simulates, effectively making it almost impossible to distinguish from the original device, at least for the simulated functionality. As such the simulated infrastructure should be transparent to the simulator external applications, meaning that the infrastructure hosting n devices composed of r real and s simulated ones would act as a composition of them. Of course looking closer at the device specific info one could realise the interconnection between real and simulated ones, however their behaviour should be as a total indistinguishable of a similar infrastructure composed of only real devices.

J. Self-* behaviour

In complex environments self-* features such as self-configuration (automatic configuration of components), self-healing (automatic discovery and correction of faults), self-optimization (automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements) and self-protection (proactive identification and protection from arbitrary attacks) are expected to exist. Being able to integrate such capability in the simulated behaviour will provide a more realistic macroscopic behaviour of the whole infrastructure.

III. Addressing the simulation requirements with multi-agent systems and web services

To address the majority of requirements mentioned, we have selected to use multi-agent systems i.e. the JADE platform [5], and web services targeting the device world i.e. the Devices Profile for Web Services (DPWS) [7].

Multi-agent systems (MAS) [4] are considered one of the most important paradigms for conceptualizing, designing, implementing and simulating software systems. They support group behaviour of agents in dynamic situations, and are capable of simulating systems with large number of heterogeneous entities behaving differently. As such MAS are suitable for evaluating distributed systems that involve complex interaction between entities, e.g. humans, industrial robots, smart devices. As agents are autonomous and operate without human intervention MAS can model really complex non-deterministic systems governed by common and possibly even conflicting goals.

DPWS defines a minimal set of implementation constraints to enable secure Web Service messaging, discovery, description, and eventing on resource-constrained devices. DPWS builds on several core Web Services standards such as WSDL 1.1, XML Schema, SOAP 1.2, WS-Addressing, WS-Metadata Exchange, WS-Transfer, WS-Policy, WS-Security, WS-Discovery and WS-Eventing. In Aug 2008, a new committee [10] was formed in OASIS to enable secure web service discovery and control of networked devices, which in detail is about standardisation of WS-Discovery, SOAP-over-UDP, and DPWS. The main issues associated with DPWS and relevant to our requirements are i) it runs on resource-constrained devices, ii) allows dynamic discovery of devices and services running on devices, and iii) developer implementations of it are available as open source in Java and C (e.g. www.soa4d.org, www.soa4d.org). A DPWS implementation (WSDAPI) is also included by default in Microsoft Windows Vista and Windows Embedded CE (Windows Communication Framework).

- Dynamic Discovery of Devices and Services: As DPWS-enabled devices support WS-Discovery, any device using DPWS can be discovered. Furthermore subscription to the events generated is possible which gives us more capabilities in simulating event-based infrastructures (which are the most likely paradigm to dominate real world environments e.g. industrial shop-floors).
- Roaming: As devices are connected to and controlled by agents, we can use the mobility capabilities of the agent system to simulate the movement of devices across domains with the movement of agents among agent systems.
- Volatile Devices: the agents can change the status of the simulated services (e.g. start, stop etc) based on their internal strategy, therefore effectively create the effect of a volatile infrastructure.
- Cooperation Capabilities: Agent cooperation is one of their characteristics that has been developed thoroughly the last decades. Several social phenomena are already modelled and simulated by agents. Furthermore due to the variety of available tools such as the agent communication language (ACL), interoperability framework etc., it is easy to built on top and create more sophisticated cooperation capabilities. Already defined semantics and ontologies ease the communication evaluation and promote the cooperation.
- Heterogeneity: Traditionally agents have been used to wrap heterogeneous functionality to enhance interoperability, but also due to their internal logic, external behaviour can be easily changed which gives us the capability of simulating heterogeneity.
- Large Scale: Multi-agent systems are designed to be scalable and accommodate thousands of agents that carry out their tasks. As such they are suitable for accommodating simulation of large numbers of devices.

- Complex Devices Simulation: Complex devices are built from simpler components that operate based on a bigger plan. Agents can simulate simple devices, however due to their communication as well as the goal-based behaviour, they can be combined/grouped and controlled by other agents, in the essence forming an ecosystem that depicts complex behaviour. Therefore in theory we can simulate extremely complex devices, but if the implementation overhead is justified, needs to be evaluated.
- Real and Virtual Data Sources: Real devices will be connected and emit data e.g. temperature. It should be possible to realize this in an interoperable way and this will be done by implementing it as a web service via DPWS, where others can subscribe to. Furthermore we need to support virtual data sources. These can either be pre-gathered values stored in a database, on-the fly generated values by an algorithm, or even values created by a real device that are processed and presented also as a web service for consumption.
- Transparent Device Simulation: The agents can discover and use the data emitted by a real device and then create duplicate simulated devices that depict exactly the same behaviour to the outside world. To realize this the metadata of each device should be collected and processed, as well as the real-time readings or results of a service the device offers. This can be partially covered by DPWS as it makes device specific info available.
- Self-* behaviour: Autonomy is a key characteristic of agents, while their capability of realising behaviours has already been used by the research community to create agents depicting self-* behaviour. As such part of it can be also observed on the simulated devices that are created and controlled by the respective agents.

IV. The Multi-Agent Simulation Environment

As a proof of concept a simulation environment has been developed. In Fig. 2 several layers can be seen. The devices at the lowest layer make available their functionality via web services, while a subscription can be made to its services. The device layer consists of devices that directly implement web services e.g. via the DPWS protocol, and/or via DPWS gateway (due to resource constraints etc). Typical examples of such devices that implement web services (SOA-ready) are programmable logic controllers (PLCs), robots, advanced sensors e.g. SunSPOTs etc., and example of devices connected via a DPWS gateway could be RFID tags that connect via an RFID reader that acts as a DPWS gateway etc. At execution layer, the mobile agent system hosts several agents that not only cooperate but also control the created virtual devices. One layer higher relies the logic, which describes the scenarios the users run within the simulator. The scenarios range from simple ones running standalone up to complex which may start other simpler scenarios first. Finally at enterprise layer, various services and applications can communicate via web services with the devices, both real and simulated ones.

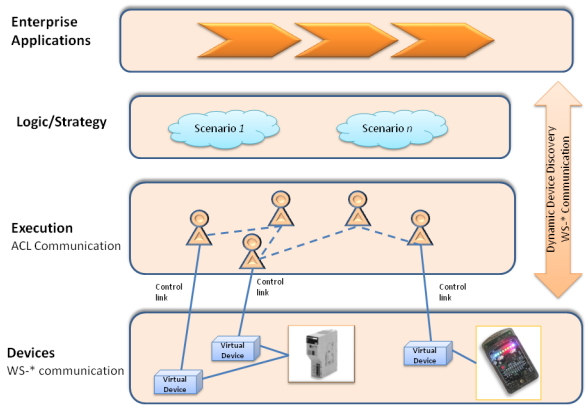

Fig. 2. Simulator Overview

Each Agent represents one SOA-ready device. Some concepts and implementation details can be found in [1]. The Agent can get initial or continuous data either from the devices they simulate in real-time by connecting to them via WS, or by getting predefined values stored in a database or even by generating their own based on internal algorithms. The simulator is part of an ecosystem and completely transparent to the other actors of the infrastructure including the enterprise applications and other services and devices. As an example, SAP's Manufacturing, Integration and Intelligence (MII) product that typically links a shop floor system with an ERP and provides enterprise services with information from plant floor applications and systems can discover all devices via WS-Discovery [9]. The functionality of MII is extended via a DPWS client that can discover the devices created by the simulator and cannot distinguish them from the real devices. This allows us to create large-scale infrastructures in agnostic-ways for the other layers. We have to note, that agents representing devices can communicate with the outside world via the DPWS, while internally the facilities offered by the Agent platform can be used in parallel i.e. the Agent Communication Language (ACL). This gives us extended capabilities as we have two different communication and control planes (that of Agent system and of DPWS) that can be used independently.

The simulation environment consists of a basic set of agents, each of which has its goals and internal logic.

- Management Agent: Tasks of this Agent include evaluation of user arguments, creation of other agents and other management functions e.g. logging etc.
- Device Explorer Agent: This Agent is based on the concept of DC-Agent (DPWS Client Agent [1]) and its job is to discover all the DPWS enabled devices in the network based with a specific scope
- Device Generator Agent: The core function of this Agent is to receive and execute requests towards creating and initializing service agents that simulate a specific service.
- Scenario Agent: This agent is specific for each scenario as it executes its strategy/logic.

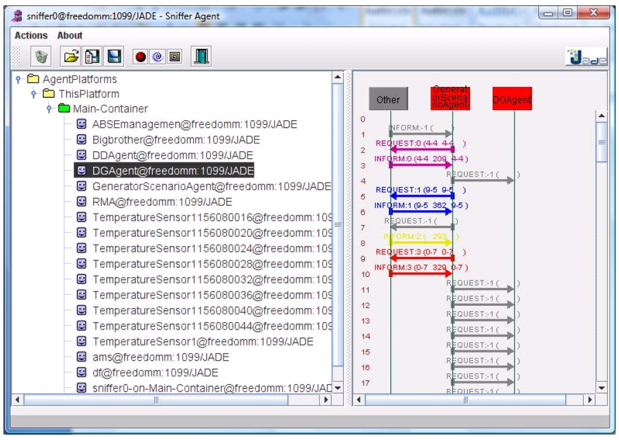

Fig. 3. Agents simulating temperature sensors

- Service Agent(s): Design of a service Agent is based on the DS-Agent model [1]. Such types of agents simulate a DPWS service and are visible to the external world via DPWS communication.

V. Selective demonstration scenarios

In order to investigate the capabilities of the simulator we have investigated some demonstration scenarios, that also depict how the different requirements are addressed. These have been implemented in an evolutionary approach, starting with simple functionality and building on top of it more advanced ones.

A. Generator scenario

This scenario creates Service Agents that expose specific virtual services. To service consumers these virtual services appear just like any other real DPWS service available through some DPWS enabled physical devices. An amplified number of the discovered DPWS devices is created, depending on the parameters. For instance, if two devices are discovered on the network and the amplification size is of value ten, then it will create ten virtual services for each one of the two devices, thus the total number of the DPWS devices exposed on the network will be twenty-two (twenty created and two real devices). Of course if no devices are available or if it is wished, devices can be created based on user defined device profiles (described in XML). Any of the created devices also acts as source device and other virtual services can be created from these (tree structure). The goal here is to test the dynamic discovery and creation of virtual devices based on discovered ones. As the numbers increase and network communication comes to play the result is non-deterministic as some devices may not be discovered within the specific discovery timeframe.

B. Amplification scenario

This scenario is an advanced version of Generator simulation scenario. It supports real-time data transfer between a source device service and the virtual services simulating this source. In other words it supports Service Agents with integrated DPWS clients, and a DPWS connection and subscription feature to transfer data in real-time. Let us consider this example: Suppose the amplification size is ten and Device Explorer Agent discovers three DPWS devices active on network. Then this scenario will create ten virtual devices for each discovered device (as shown in Fig. 3). Then the data channels based on DPWS connection and subscription features will be established between the source device and simulated services. In other words one source device will provide data to ten virtual devices/ services in real-time. The goal here is to subscribe to live events created by the source devices and see the effects on the resources available, the network utilization etc. For our experiments the SunSPOTs wireless sensors of SUN Microsystems have been used where their functionality is offered as DPWS services [9]. The simulated devices subscribe to the temperature readings that are in real-time measured by the SunSPOT's sensor and offered as a web service (as depicted in Fig. 3).

C. BigBrother scenario

The BigBrother simulation scenario heavily depends on the above mentioned scenarios. It compares and evaluates the number of Service Agents present in the system versus the number of simulated DPWS services active in the network. As the amplification size of the simulation increases, and many parameters come into play e.g. network reliability, high volume of messages on the network channel, resource consumption etc. the difference between the number of Service Agents and virtual DPWS services detected might differ. A Service Agent and its DPWS service are two independent entities and they are connected to each other via Agent's integrated DPWS client [1]. So as the load on the resources increases (due to large amplification size of the simulation), it is possible that the Service Agent will continue to function but its service might crash, or not be reachable or the Agent might not be able to start the service etc. This will lead to a difference between the actual number of agents and DPWS devices started. The aim of this Agent is to monitor and identify this gap.

D. Assassin and saver

The concept of this scenario is similar to the Big Brother simulation scenario but allows us to include strategies and bring in dynamicity. The main purpose is to simulate a highly dynamic system where devices appear and disappear non-deterministically. The same holds true for a non-reliable service infrastructure. We realise this by trying to terminate and create Service Agents simultaneously. This is a scenario where the one or more agents from each kind (saver or assassin) fight to achieve their goals. The Saver agent(s) try to keep the population of devices/services with specific characteristics stable, while in parallel the Assassin agent(s) try exactly the opposite i.e. to minimize agent populations based on specific criteria. The logic of behind this scenario can range from simple (like the one we implemented here as proof of concept) up to very complex ones e.g. from the game theory domain.

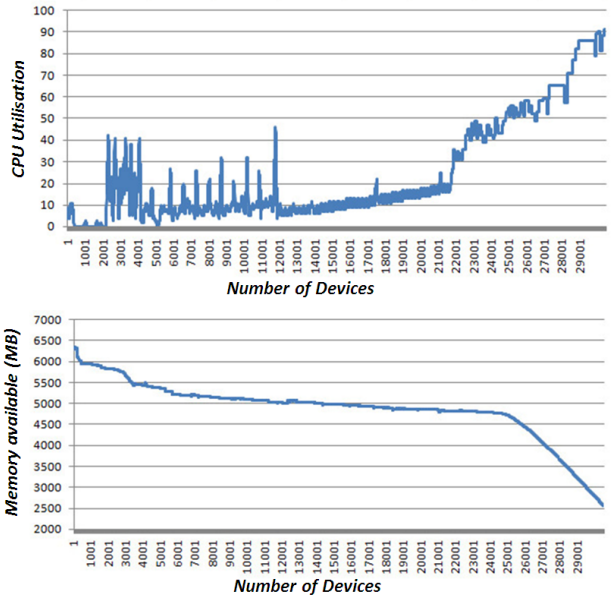

Fig. 4. Example: Memory and CPU measurements with respect to 30000 simulated agents and DPWS devices in the "amplification scenario"

E. Evaluation

We have made some initial measurements by monitoring system resources, the time to create a service agent, the number of agents created etc. All of the tests were done in a PC with the following configuration: Intel®Core™2 Duo CPU 6600 , 2.4 GHz (x64), 64-bit Microsoft Windows VISTA, 8 GB RAM, 64-bit SUN Microsystems Java SE v1.6 (server VM with increased heap size).

As it is depicted in Fig. 4 for up to 25000 devices created, the simulator is responsive and can create an infrastructure simulating this amount of devices. However from then on, the system resources utilization i.e. memory and CPU increases dramatically, which results to the system being non responsive. Although devices and their respective agents can be created, the subscription to the services they offer after a specific point either fail to start or take too long to provide the necessary events. As such we propose to use a single physical system as the one described here only for simulations of up to approx. 20000 devices. If subscriptions are to be fully functional in a timely manner, it is also advisable to limit the number of simulated DPWS devices below the 10000 limit. Physically extending the resources either by adding more RAM or even better use other computers running the simulator (in a cluster), may lead to extending the numbers mentioned, which would lead to larger scale infrastructures. Please also have in mind that the whole system presented here was implemented as a proof of concept, and no optimizations whatsoever have been done on the code or on the concepts to take optimal advantage of the resources. Therefore, higher number of devices might be possible with a single computer if the code or parts of the architecture are optimized. It has however to be pointed out that this number of devices within one network segment fully covers existing infrastructures.

VI. Future Work and Conclusion

The work presented shows how the real world devices can be coupled with virtual devices simulated by a multi-agent system and how both of them can create an ecosystem that can be used transparently by applications and other services relying at enterprise, network or device layer. This work was implemented as proof of concept, and a number of issues have been identified that could build on top.

- Large Scale Distributed Infrastructures: As it was demonstrated, we have run this simulation for thousands devices. However in the future efforts should concentrate on running it in a distributed infrastructure with multiple physical computers and devices across different network segments. Larger number of devices in the millions domain, might provide more interesting research questions and results
- Strategy: Most of the scenarios are pretty simple, and were done as proof of concept. However, the basic blocks are there to realize much more complex scenarios e.g. coming from the artificial intelligence and game theory domains.
- Mobility: The focus of this work was on the functionality and on stationary agents. While it is not considered to be a significant difficulty to introduce mobility in all the aforementioned work and scenarios (JADE supports it seamlessly), this will fundamentally change the results in the simulation scenarios introducing more complexity. However this will provide very interesting research challenges, that come closer to real-world infrastructures of roaming devices.
- Cooperation: The cooperation aspects so far have been kept to minimum. However the agents provide very advanced capabilities for cooperation, and this should be further investigated also with the use of semantic technologies.
- Toolkit: A toolkit that would allow easy configuration and real-time interaction with the framework itself needs to be developed in order to minimize the learning curve and make the simulator more user-friendly.
- Enterprise Integration: Stronger enterprise integration should be achieved, i.e. services that would make it possible to configure and manage the framework in a transparent way. Extension of some enterprise modelling tools could allow enterprise service developers to create on the fly large scale infrastructures and test their applications with different configurations.
- Performance: Enhancements should be made to enhance the existing performance of the system, including further investigation of the exact limitations that the system can reach.

Future enterprise services will heavily depend on the data acquired from millions of devices; therefore simulating such infrastructures in order to test aspects of it such as communication overheads, performance, model services capable of

dealing with dynamic changes etc will become critical. From our viewpoint the developed simulation framework depicted here can offer great insights on how this could be realized. As the basic blocks are conceptually and technologically there, others can built more complex systems on top of this in order to more realistically simulate a real-world infrastructure.

ACKNOWLEDGEMENT

The authors would like to thank the European Commission and the partners of the European IST FP6 project "Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices" (SOCRADES - www.socrades.eu), for their support.

REFERENCES

[1] Stamatis Karnouskos, Mian Mohammad Junaid Tariq, "An agent-based simulation of SOA-ready devices", IEEE supported 10th International Conference on Computer Modelling and Simulation, 1-3 April 2008, Cambridge, England.

[2] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation", IEEE Transactions on Industrial Informatics, p. 62-70, 2005.

[3] Stamatis Karnouskos, Oliver Baecker, Luciana Moreira Sa de Souza, Patrik Spiess, "Integration of SOA-ready Networked Embedded Devices in Enterprise Systems via a Cross-Layered Web Service Infrastructure", 12th IEEE Conference on Emerging Technologies and Factory Automation (ETFA), September 25-28, 2007, Patras, Greece

[4] Michael Wooldridge, "An Introduction to Multiagent Systems", February 2002, John Wiley & Sons (England). ISBN 0 47149691X.

[5] Fabio Luigi Bellifemine, Giovanni Caire, Dominic Greenwood, "Developing Multi-Agent Systems with JADE", Wiley Series in Agent Technology, 2007, ISBN-10: 0470057475

[6] E. Fleisch and F. Mattern, editors. "Das Internet der Dinge: Ubiquitous Computing und RFID in der Praxis: Visionen, Technologien, Anwendungen, Handlungsanleitungen" (in German), Springer, 2005.

[7] Shannon Chan et al., "Devices profile for web services", February 2006. http://schemas.xmlsoap.org/ws/2006/02/devprof

[8] OPC Unified Architecture (OPC-UA), OPC Foundation http://www.opcfoundation.org/UA

[9] Domnic Savio, Stamatis Karnouskos, "Web-service enabled Wireless Sensors in SOA environments", 13th IEEE Conference on Emerging Technologies and Factory Automation (ETFA), September 15-18,2008, Hamburg, Germany.

[10] OASIS WS-DD Technical Committee, http://www.oasis-open.org/committees/ws-dd/