www.mdpi.com/journal/sensors

Conference Proceedings Paper - S2 Smart Systems and Structures

Industrial Agents and Distributed Agent-based Learning

Stefan Bosse*

University of Bremen, Department of Mathematics & Computer Science, Robert Hooke Str. 5, 28359 Bremen, Germany

* Author to whom correspondence should be addressed; E-Mail: sbosse@uni-bremen.de
Published: 5 November 2015

Abstract: Today sensor data processing and information mining become more and more complex concerning the amount of sensor data to be processed, the data dimension, the data quality, and the relationship between derived information and input data. This is the case especially in large-scale sensing and measuring processes embedded in Cloud environments. Measuring uncertainties, calibration errors, and unreliability of sensors have a significant impact on the derivation quality of suitable information. In the technical and industrial context the raising complexity and distribution of data processing is a special issue. Commonly, information is derived from raw input data by using some kind of mathematical model and functions, but often being incomplete or unknown. If reasoning of statements is primarily desired, Machine Learning can be an alternative. Traditionally, sensor data is acquired and delivered to and processed by a central processing unit. In this paper, the deployment of distributed Machine Learning using mobile Agents forming self-organizing and self-adaptive systems (self-X) is discussed and posing the benefit for the enhancement of the sensor and data processing in technical and industrial systems. This also addresses the quality of the computed statements, e.g., an accurate prediction of run-time parameters like mechanical loads or health conditions, the efficiency, and the reliability in the presence of partial system failures

Keywords: Industrial Agents, Multi-agent Systems, Self-X Systems, Distributed Learning

1. Introduction

In the last decades there was a shift from passive single sensors towards networks of smarts sensors equipped with Information-Communication Technologies (ICT). Furthermore, there was a significant increase of the sensor density in sensor networks **[1]**, shown in Fig. **1**. Ongoing miniaturization of sensors and new micro-system technologies enable the integration of sensor networks in materials and technical structures **[2]**. In contrast to generic computer networks, Material-integrated sensor networks pose specific requirements and constraints of the information processing regarding resources, computational latency, energy consumption, and robustness. Usually there is limited or no possibility of maintenance in the case of technical failures, demanding self-organizational and self-adaptive ICT systems. The size of a smart sensor node reaches currently the mm^3 scale, like the Smart Dust Mote **[3]**

One one side there is a miniaturization trend making things smaller, on the other side there is a growing demand for Cloud computing and solutions, e.g., used for coupling of production, design, and products for life-cycle management **[4]**. A Cloud is characterized by its localized virtualization of storage and computational power, commonly using service-based architectures. A Cloud provides a coarse-grained distribution primarily offered by data centers that are scalable by adding servers and more data centers. Cloud environments make a significant contribution for solving the Big-data issue and the information extraction of a large set of uncorrelated data.

Future industrial environments consisting of production, design and customers require multi-scale data processing, from micro- to macro-scale computing. These computing and networking environments are strongly heterogeneous, regarding host platforms and network technologies.

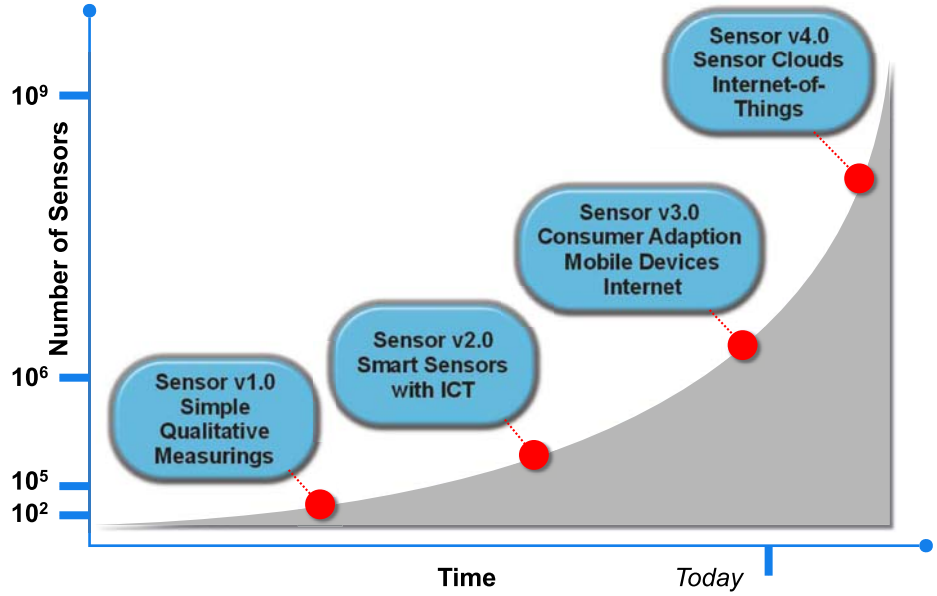

Figure 1. Generations of Sensors: From passive Sensors towards Sensor Clouds

This article points out new methodologies using mobile and learning agents, enabling the design of efficient and scalable data processing of the future in sensor and industrial networks. Self-organizing and self-adaptive structures will be key methodologies decreasing the administrative work and increasing the reliability, and finally integrating Cloud-based solutions.

2. Multi-Agent Systems and Industrial Agents

Agents are semi- or full autonomous computational units. They consist basically of a perception, control, and planning module accessing private data and encapsulated by a computational process. This process is characterized by its dynamic control- and data state. The private data of an agent bases firstly on a perception process of the environment (e.g., sensor data), and secondly on computed data. The dynamic behaviour and behaviour adaption (of the control module) depending on current and past data are essential features of an agent. The adaptivity is strongly related to the concept of learning.

The behaviour of agents is related to living things, e.g., the widespread Belief-Desire-Intention (BDI) architecture **[5]**, mostly using declarative descriptions. An Activity-Transition Graph (ATG) (see Fig. **2**) is a more simplified behaviour model, which models the behaviour of the agent using a set of activities. Activities (the nodes of the graph) perform actions, e.g., computation data and interaction with the environment including migration. There are transitions between activities (the edges of the graph),

commonly depending on conditions and private data of the agent. The ATG model is suitable to meet the high-level features of agents: Autonomy, social abilities, reactivity, pro-activity, and self-organization combined with self-adaptivity (self-X).

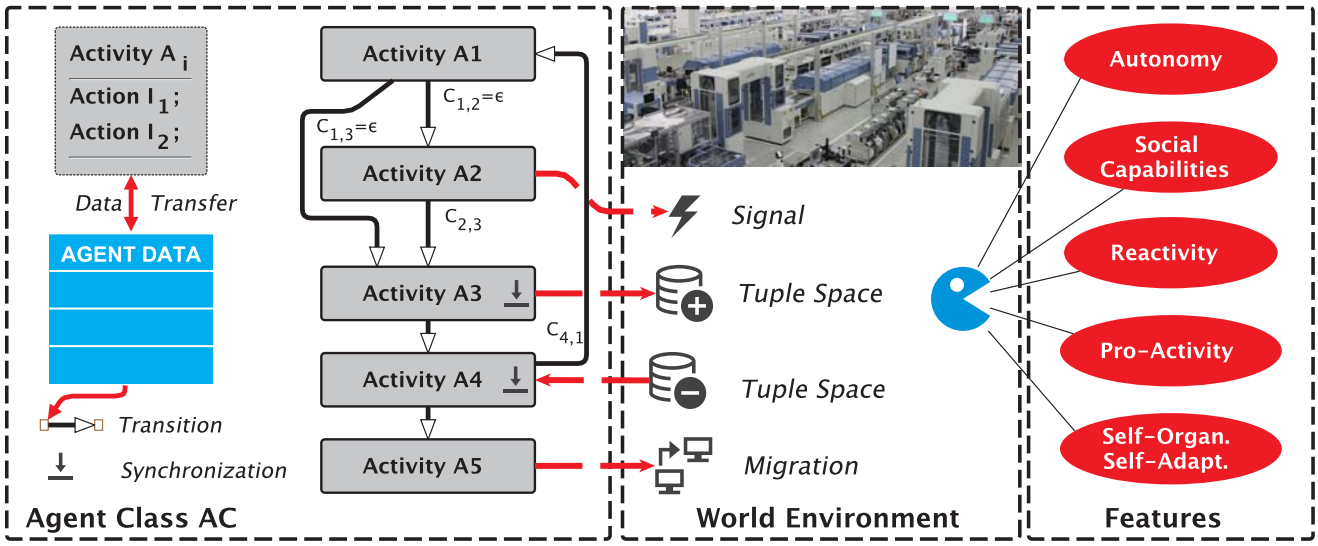

Figure 2. Activity-Transition Graphs for simple modeling of the agent behaviour.

The activities (i.e., statement blocks) allow a partitioning of the overall agent behaviour basically defining the goals of the agent, and they can be considered as coarse-grained execution steps. Activities are executed by an agent platform as a host. An ATG can be modified at run-time by the agent itself, enabling self-adaption, by other agents, or by the platform. The modification of the ATG is done by a change of the transitions and/or activities (removal, exchange, addition).

Agents are already successfully deployed in industrial environments, mainly for planning tasks **[6]** and manufacturing control **[7]**, but increasingly for global networking of production and design processes **[4]**.

Beside the control of production processes agents are deployed in fields of maintenance, modular production systems (assembly control), quality control, and energy management. Self-organizational and adaptive capabilities of agents play an increasing role in industrial environments. The new paradigm of industrial agents can provide a significant contribution to future modular and flexible industrial environments embedded in Clouds.

Mobile agents are capable to transfer a snapshot, consisting of their data and control state, from one to another agent platform, enabling a seamless execution of agents. Mobile agents can reflect a mobile service architecture.

Multi-agent systems consist of a large number of different agents communicating with each other. Communication can take place directly by using messages (e.g., FIPA-ACL), or implicitly and much more decoupled by using tuple spaces with pattern matching search **[8]**. In distributed systems a big task is mapped on many simple tasks using agents, following the divide-and-conquer approach. One example is distributed Data Mining with a Map&Reduce approach.

Scaling of industrial data processing applications towards complex cloud-based and widespread distributed networks including sensor networks results in the deployment of thousands and millions of agents. The agent processing platform is therefore a central key technology, discussed later.

3. Distributed Machine and Agent Learning

Machine Learning (ML) can be classified in supervised, reward feedback, and non-supervised learning, shown in Fig. **3**. Trained supervised learning is commonly used for a deviation of a classification function *K:* f(***x***) → *l* using a learner (model builder) *M*: f(***D***) → *K* from a labeled data set ***D***, consisting of ***x*** ∈ ***X*** n-dimensional vectors, e.g., sensor data ***S***, and a set of associated labels (symbols) *l* ∈ ***L***. The classification function is derived in the learning phase, applying known labels to known data sets performed either by a human or a machine. An application phase follows the learning phase, applying the classificator to an unknown data set ***x*** and delivering a prediction of a label *l*. An example is shown later using ML to recognize (classify) different load situations of a mechanical structure from sensor data. Feedback learning (e.g., reinforcement learning with reward feedback) assesses the actual perception regarding the effect of selected actions and tries to adapt the behaviour planning to choose appropriate actions finally to reach defined goals.

Unsupervised learning is commonly used for data clustering, and can be considered as a self-organizing approach.

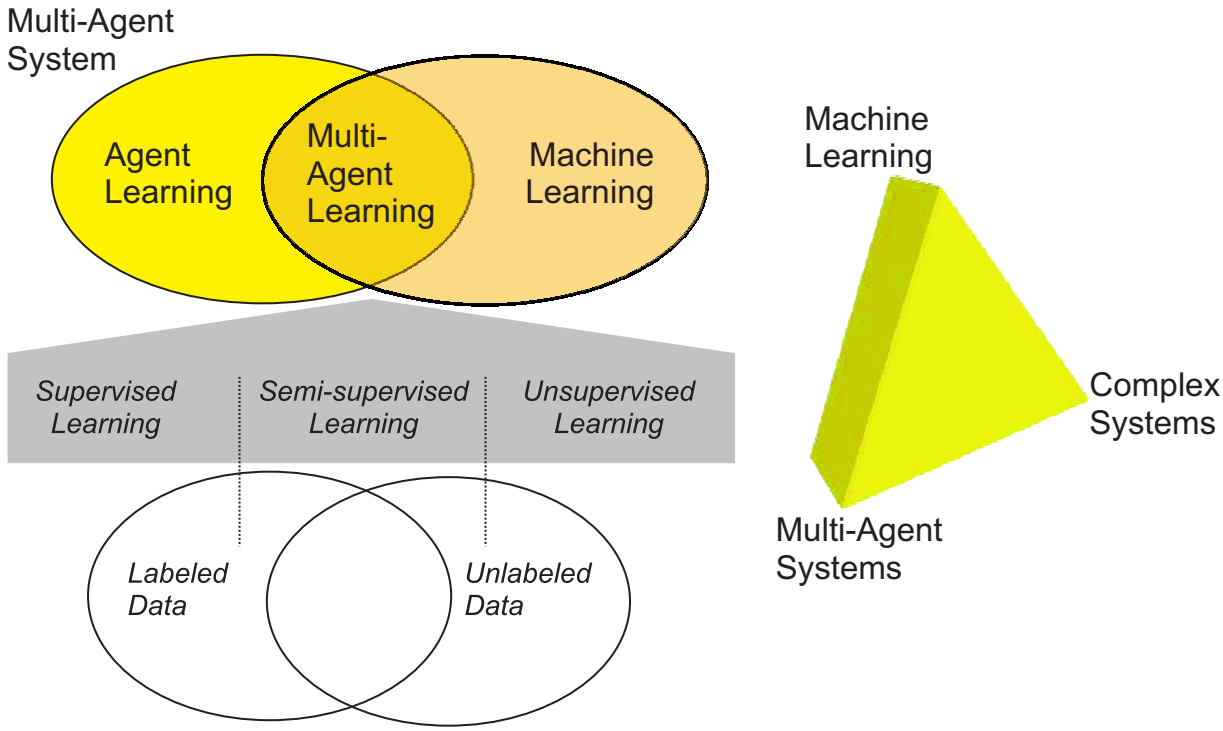

Figure 3. The combination of Machine Learning and Agents creates synergies for the composition of complex systems.

The concept of machine learning and agents can be combined offering synergy. Mobile agents are suitable for exploration tasks in spatially distributed network regions, i.e., collecting and processing of sensor data. The distributed sensor data can be used to learn classification models related to specific situations, e.g. different load situations of a mechanical structure. The learned model can be carried by a learner agent and applied at another location. This capability is important in mobile network environments, especially using mobile devices like smart phones as sensor nodes. The mobility of the learner can be also used to migrate between different mobile devices and collecting sensor data from different devices (e.g., smart phones) within a spatially bounded region (Ubiquitous Computing, **[9]**). Reinforcement learning is closely related to the agent model due to their adaptive behaviour and planning of actions based on perception. Possible actions of such learner agents can be the optimization of actuator and machine control.

Beside classical learn algorithms with separated learning and application phases the usage of incremental (on-line) learners gains importance **[10]**. They can collect new data sets at run-time to improve an already learned model (e.g., decision tree learner or neuronal networks) without the requirement to save the entire training data set base.

Usually learner are centralized, i.e., all input data is collected and processed by one program. This architecture introduces a single point of failure and high data stream densities in the network. But there are approaches to distribute learners using agents. One possible approach bases on the partitioning of the learning process in multiple local learners operating on a data sub-set. The locally learned models are finally fusioned to a global model. This is realized by partitioning the (sensor) network in spatial regions (Regions of Interest ROI) and deploying multiple learners with each learner operating in a specific ROI. Learning of classification models and application uses hence only a local data set providing a local view of the world. From a global view, the results from multiple local classifications can differ. A suitable method to derive a reliable global classification (building the global model) can be provided by a majority election and voting process. Each local learner votes for a classification prediction. The assumption made is that the majority decision delivers the most probable result. The distributed learner systems have a very good scaling capability compared to central learner, and there is no single point of failure. Defective nodes or missing votes only lower the global prediction accuracy.

Formally, the spatial distribution of the learning processes deriving a classification model from sensor data is given by:

$$\begin{array}{l} M : D \to K(S) \\ l \in \boldsymbol{L} \\ K : S \to l \\ D : \{(S^1, l^1), (S^2, l^2), \cdots\} \\ S : \begin{pmatrix} x_{1,1} & \cdots & x_{n,1} \\ \vdots & \ddots & \vdots \\ x_{1,m} & \cdots & x_{n,m} \end{pmatrix} \end{array} \xrightarrow{\text{Distribution}} \begin{array}{l} m_{i,j} : d_{i,j} \to k_{i,j}(s) \\ k_{i,j} : s_{i,j} \to l_{i,j} \\ K : (l_{1,1}, l_{1,2}, \cdots) \to l \\ d_{i,j} : \{(s_{i,j}^1, l^1), (s_{i,j}^2, l^2), \cdots\} \\ s_{i,j} : \begin{pmatrix} x_{i-u,j-v} & \cdots & x_{i+u,j-v} \\ \vdots & \ddots & \vdots \\ x_{i+u,j-v} & \cdots & x_{i+u,j+v} \end{pmatrix} \end{array} \quad (1)$$

if $\boldsymbol{S}$=(s_1,s_2,..) is a matrix (or vector) of single spatially distributed sensors with data $x_{n,m}$ at position (n,m), and $\boldsymbol{S}_{i,j}$ a sub-matrix around a spatial center point (i,j), K the global, and $k_{i,j}$ the local learned classification function (model). The training data sets consist of tuples (s,l) with an associated label.

4. Agent Platforms

The agent processing platform is an enabling technology in strong heterogeneous environments that supports the seamless migration of mobile agents between different host platforms and network environments without any further transformation. Currently existing agent platforms cannot handle a large number of agents (below 1000 agents), and the deployment is often limited to the laboratory scale **[7]**. If the multi-agent approach should be applied to large scale problems, a significant larger number of agents must be handled.

Furthermore, the deployment of agent platforms in strong heterogeneous environments, e.g., by connecting sensor, industrial, and inter-networks, demand for different platform implementations (Hardware, Software, Simulation, Browser & Server) on different host platforms (microchip, embedded and mobile system, generic computer, server, WEB browser), shown in Fig. **4**. There are only a few

prototypical agent platforms delivering this broad range of implementations. One example is the portable JavaScript Agent Machine (JAM) platform **[11]**, being capable to execute agents with mobile JavaScript code efficiently. This code also embeds the control and data state of an agent. A stack-processor based solution is used in the Pipelined Agent Virtual Machine (PAVM) **[12]**, which also includes the hardware implementation level. The PAVM executes FORTH code with embedded data. Both PAVM and JAM base on the ATG agent model and are being basically compatible on an operational level.

The JAM platform is one of the few platforms offering Machine Learning as a service, i.e., a learner agent saves only a learned model, but not the learner code, increasing mobile efficiency significantly, which depends on the agent's code size.

The deployment of mobile agents and agent platforms in the Internet or industrial networks require additional organization and security structures. E.g., in a material-integrated sensor network all nodes have a geometric neighbourhood connectivity, not existing in the Internet. Additionally, the Internet is a network-of-networks. Mobility of agents therefore require virtual connectivity and communication structures delivered by the platform as a service. One possible approach are distributed directory services placing host and agent platforms in domains based on, e.g., geographic coordinates **[11][12]**. This virtual word is required for mobile agents making decisions about the direction they will move. Finally, directory graphs can reflect hierarchical network structures and network-of-networks.

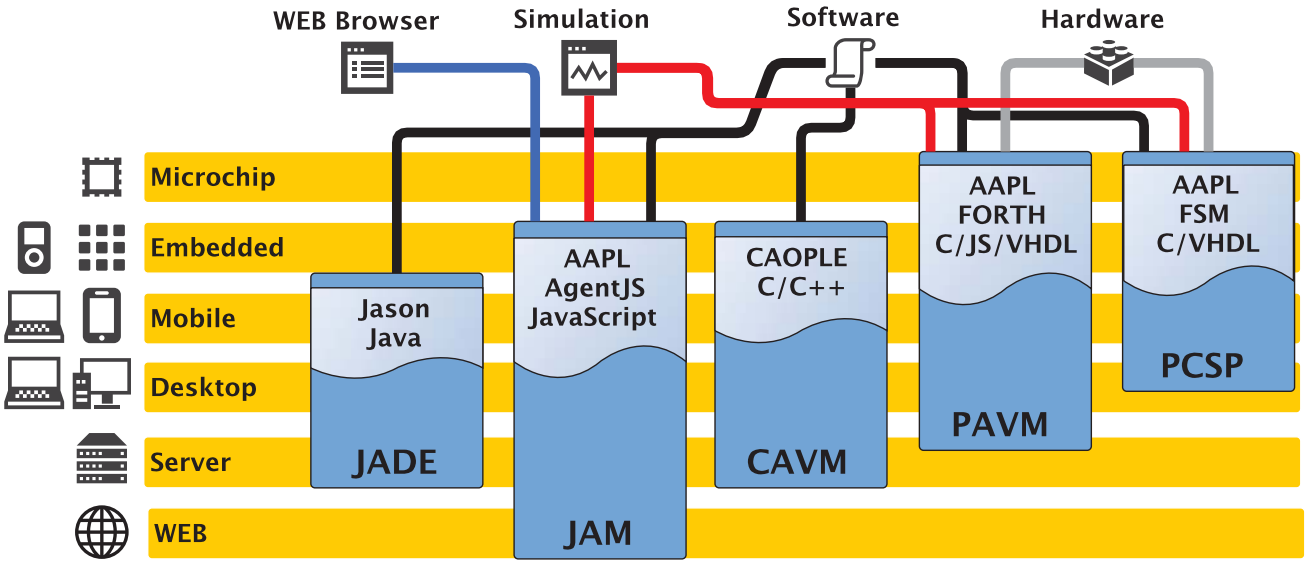

Figure 4. Different host platform levels, implementations, and Agent platforms (JADE: **[17]**, JAM: **[11]**, CAVM: **[13]**, PAVM: **[12]** , PCSP: **[14]**)

Mobility of agents, i.e., processes, require two important features: (I) Efficient creation of a process snapshot in program format, containing the code and the actual control and data state of the process, (II) Low dependency of the program on the platform interface, the host platform (computer architecture), and on the network architecture and topologies. This concludes the avoidance of binary machine code. The programming language JavaScript provides an isomorphic transformation between the actually executed code and program text at run-time on common virtual machines (node.js, Google V8,etc.). Furthermore, the capabilities to reconfigure and adapt the agent behaviour require the capability to modify the program code of an agent at run-time (often by the agent itself).

5. Example: Load Monitoring of Mechanical Structures

A major application example for distributed learning with respect to the approach from Eq. **1** is Load and Structural Health Monitoring (LM/SHM) of mechanical components. Learning can be used to recognize different load situations (l_1,l_2,,..) derived from sensor data without any specific mechanical model of the component (e.g., a Finite-Element model). It is assumed that a sensor network delivers the required spatially resolved sensor data from strain gauge sensors, either applied to the surface of the component or integrated in the material, shown in Fig. **5** on the left side. Each node of the network are provides beside sensors the agent platform. A specific load situation has impact on the component, e.g., it is useful to recognize situations causing damages or delivering input for a device control to reduce loads.

In this scenario a Multi-agent system consisting of a couple of different mobile and non mobile agents populates the network (see Fig. **5**, right side). The agents have different goals, tasks, and behaviours:

- **Node agent**: Each sensor node is populated with a non-mobile node agent performing sensor acquisition, sensor preprocessing, and event detection (i.e., recognizing a significant stimulus). A node agent can instantiate new agents for specific tasks or notify already working agents.
- **Learner agent**: Each sensor node has at least one learner agent, which is instantiated and activated by the node agent if there was a sensor stimulus detected. This learner has two different modes: (I) Learning (II) Classification with a learned model. The mode selection is performed by **notification agents**, distributed in the network and (I) Notifying learners about a characteristic load situation (providing a label *l*) and induce the learners to create a training data set with a specific label, (II) Switching learners to application mode. The learner agents have access to sensor data from the near neighbourhood, collected by explorer agents and passed by the tuple space data base (a platform service). The learners create a local sensor-load situation model.
- **Exploration agent**: This agent delivers input for the prediction of a significant sensor stimulus and for the learning with sensor data in a spatially constrained Region of Interest (ROI). The spatial sensor exploration is performed with a divide-and-conquer approach by a set of exploration agents collecting the sensor data and delivering the data to the node or learner agents. Each explorer agent operating on a specific node in the ROI creates explorer child agents exploring data on neighbourhood nodes.
- **Voting agent**: If a learner agent classifies a load situation from his local view (and the local model using local data) it will send out voting agents with a prediction of the load situation. The voting agents deliver the votes to **election agents**, which perform a majority election for a global and most probable prediction of a load situation.

Most agents are created dynamically by other agents, e.g., the exploration agents are created by node, learner, and other explorer agents. Agent interaction takes place by using tuple spaces (synchronized data exchange based on patterns). Furthermore, mobile signals are used for notification of other agents.

In **[15]** an example network consisting of 64 sensor nodes deploying such a MAS were simulated, shown in Fig. **5**. It is assumed that the mechanical structure is equipped with spatially distributed strain gauge sensors. Depending on the load situation and the change of sensor values the network has peak populations with several hundred up to thousand agents, most of them migrating between nodes. An efficient processing and migration of agents can be realized with the JAM platform. The used simulator was built on top of a JAM platform.

A major advantage is the event-based sensor processing activating only stimulated areas, in contrast to common continuous stream-based processing activating the entire network and creating high communication load. The event-based approach reduces computational and communication workload significantly (up to 90%). JAM agents can migrate between different network environments, i.e., between sensor networks and the Internet, enabling a partition in on-line and external off-line processing, e.g., combining on-line learning with off-line computational intensive numeric methods **[16]**. This feature enables the seamless integration of sensor network in Clouds.

The simulation results of the distributed learning and the election approach showed a good classification quality of different load situations with a reasonable high prediction probability **[15]**. The mean accuracy of the global classification (correct positive votes) was about 80% (i.e., 20% of the votes were incorrect).

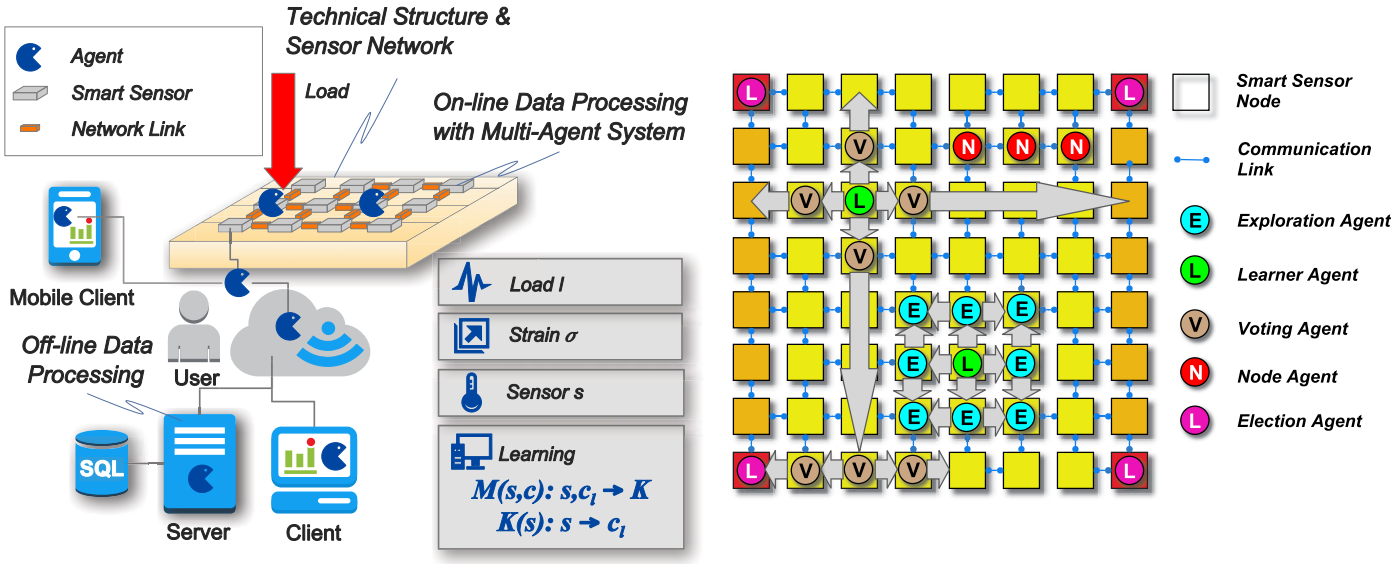

Figure 5. (Left) Technical structure equipped with a sensor network and connected to Intra- and Internet environments (Right) Logical network view and deployment with multiple agents instantiated from different behaviour classes.

6. Conclusion

Growing complexity and heterogeneity of industrial networks and their integration in the Internet and Cloud environments requires self-organizing and self-adaptive approaches composed of autonomous basic cells. Multi-agent systems are suitable to provide a scalable and efficient ICT approach for robust system design, enhanced by learning agents. Structural monitoring is one major field of application for the MAS deployment, shown by an example use case and evaluated by simulation. MAS maps the entire complex problem to be solved on multiple simple agents differing in their behaviour, goals, and operations. Local learning with global fusion based on a majority decision making process is an appropriate method to identify specific situations in a distributed sensor network, e.g., a reliable recognition of mechanical load situations.

The agent platform is a key technology. The introduced JAM platform that is entirely implemented in JavaScript provides a portable multi-platform processing platform for mobile agents, suitable for strong heterogeneous environments. The agents are programmed in JavaScript, too, and snapshots of agent processes are transferred with JavaScript text-code embedding the control and data state. The text-code can be executed by any host platform (no machine dependencies). This enable a seamless migration of agents between different host platforms and networks.

References

1. V. Di Lecce, M. Calabrese, and C. Martines, *From Sensors to Applications: A Proposal to Fill the Gap,* Sensors & Transducers, vol. 18, no. Special Isse, pp. 5–13, 2013.
2. D. Lehmhus et al., *When nothing is constant but change: Adaptive and sensorial materials and their impact on product design*, Journal of Intelligent Material Systems and Structures, Sep. 2013
3. B. Warneke, M. Last, and B. Liebowitz, *Smart dust: Communicating with a cubic-millimeter computer*, Computer, 2001.
4. D. Lehmhus, T. Wuest, S. Wellsandt, S. Bosse, T. Kaihara, K.-D. Thoben, and M. Busse, *Cloud-Based Automated Design and Additive Manufacturing: A Usage Data-Enabled Paradigm Shift*, Sensors MDPI, vol. 15, no. 12, pp. 32079–32122, 2015
5. R. H. Bordini and J. F. Hübner, *BDI agent programming in AgentSpeak using Jason*, Computational Logic in Multi-Agent Systems, Volume 3900 of the series Lecture Notes in Computer Science, Springer, 2006, pp. 143-164.
6. M. Caridi and A. Sianesi, *Multi-agent systems in production planning and control: An application to the scheduling of mixed-model assembly lines*, Int. J. Production Economics, vol. 68, pp. 29–42, 2000.
7. M. Pechoucek, V. Marík, 2008. *Industrial deployment of multi-agent technologies: review and selected case studies*. Auton. Agent. Multi-Agent Syst. 17 (3), 397–431
8. L. Chunlina, L. Zhengdinga, L. Layuanb, and Z. Shuzhia, *A mobile agent platform based on tuple space coordination*, Advances in Engineering Software, vol. 33, no. 4, pp. 215–225, 2002
9. E. Pournaras, I. Moise, D. Helbing, *Privacy-preserving ubiquitous social mining via modular and compositional virtual sensors*. In 2015 IEEE 29th International Conference on Advanced Information Networking and Applications (pp. 332-338).
10. F. Jiang, Y. Sui, and C. Cao, *An incremental decision tree algorithm based on rough sets and its application in intrusion detection*, Artif Intell Rev, vol. 40, pp. 517–530, 2013.
11. S. Bosse, *Mobile Multi-Agent Systems for the Internet-of-Things and Clouds using the JavaScript Agent Machine Platform and Machine Learning as a Service*, in The IEEE 4th International Conference on Future Internet of Things and Cloud , 22-24 August 2016, Vienna, Austria, 2016.
12. S. Bosse, *Unified Distributed Computing and Co-ordination in Pervasive/Ubiquitous Networks with Mobile Multi-Agent Systems using a Modular and Portable Agent Code Processing Platform*, in The Proc. of the 6th EUSPN 2015, Procedia Computer Science.
13. B. Zhou and H. Zhu, *A Virtual Machine for Distributed Agent-oriented Programming*, in Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering (SEKE'2008), San Francisco, CA, USA, July 1-3, 2008, 2008.
14. S. Bosse, *Distributed Agent-based Computing in Material-Embedded Sensor Network Systems with the Agent-on-Chip Architecture*, IEEE Sensors Journal, vol. 14, no. 7, pp. 2159-2170, 2014.
15. S. Bosse, *Structural Monitoring with Distributed-Regional and Event-based NN-Decision Tree Learning using Mobile Multi-Agent Systems and common JavaScript platforms*, in Procedia Technology, 2016, DOI 10.1016/j.protcy.2016.08.063
16. S. Bosse, A. Lechleiter, and D. Lehmhus, *Data evaluation in smart sensor networks using inverse methods and artificial intelligence (AI): Towards real-time capability and enhanced flexibility*, in Proc. of the CIMTEC, - 7th Forum on New Materials, Perugia, Italy, June 5 to 9, 2016
17. F. Bellifemine and G. Caire, *Developing Multi-Agent Systems with JADE*, John Wiley & Sons, Ltd, 2007.