Network Technology for Distributed Plant Automation

Thilo Sauter
Vienna University of Technology
Institute of Computer Technology
Gußhausstraße 27/E384, A-1040 Vienna, Austria
sauter@ict.tuwien.ac.at

Peter Palensky
Envidatec GmbH
Blohmstraße 31, D-21079 Hamburg, Germany
peter.palensky@envidatec.de

Abstract — This paper briefly describes the goals and strategies of the PABADIS project that uses software agent technology to create distributed structures in plant automation. Subsequently, we discuss the network requirements for flexible automation systems, in particular with respect to plug-and-participate mechanisms. Finally, we investigate whether mobile agents are reasonable in such a context. It will be shown that atomicity is a strong argument in favor of mobility, and that mobile agents can improve the flexibility and robustness of the system.

I. INTRODUCTION

One of the greatest achievements in recent plant and factory automation was the introduction of networks in the various levels of the communication hierarchy. Local area networks in the office area and field area networks at the control level allow for a comprehensive data acquisition and processing. Finally, the interconnection of these networks laid the grounds for distributed automation and control systems both in a horizontal and vertical manner – at least in theory.

From an application point of view, however, the current situation is still dominated by centralized solutions. What we find in practice is a hierarchical structure consisting of an ERP (enterprise resource planning) system at highest level, the MES (manufacturing execution system) and SCE (supply chain execution) systems in between and the actual control devices (such as PLCs, NCs, etc.) at the lowest level. Fig. 1 shows this classical three-level hierarchy and a rough correspondence with the well-known levels of the communication hierarchy in plant automation. However, the boundaries between the different levels are not clearly defined. In fact, functions of the MES and SCADA level can as well be implemented either within the control at field level or within the ERP system.

One of the major drawbacks of this hierarchy is its static structure, in particular of the ERP system. The real-world situation of the plant has to be appropriately described inside the system during setup to make the available resources known to the planning tools. Any changes in the real world such as the adding or replacement of production machines always imply reprogramming or at least reconfiguration of the software, which severely impairs the flexibility of the system.

Contrary to the conventional centralized approach, the PABADIS project (Plant Automation based on Distributed Systems, funded by the European Commission within the IST program [1]) focuses on automation in one piece production plants using distributed systems. Making use of software agent concepts and the Java-based Jini networking technology, the project's goal is to create a plug-and-participate environment in plant automation that allows a manufacturer

- to simply plug in a new machine and use it without major changes within the legacy systems and
- to make job control more flexible by augmenting "conventional" (mainstream) ERP functionality with intelligence inherent in software agents.

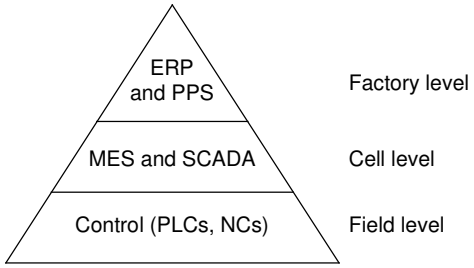

Fig. 1: Traditional three-level hierarchy of plant automation and the corresponding levels of the communication hierarchy.

The baseline vision of the project is that every workpiece has an agent "attached to it" carrying the necessary product information and moving through the plant the same way the workpiece does.

The remainder of the paper is organized as follows: In section II, we will briefly describe the goals and strategies of the PABADIS project. In section III, we will discuss the requirements for such a new type of flexible automation network. Section IV is devoted to a discussion whether or not mobile agents are reasonable in distributed automation systems. Finally, we draw some conclusions from the preceding argumentation.

II. THE PABADIS SYSTEM CONCEPT

Every production system needs two main ingredients: the actual physical workpiece and information. If we consider a single piece production system, most of this information is tightly connected to the individual product, such as

- production sequence and schedule,
- machine-related production data,
- status of the processing,
- general administrative information about the order.

In addition, there is information associated with the entire production system, such as

- overall resource use,
- overall production schedule,
- machine status information,
- quality control information.

The system-wide scheduling and resource planning data should of course be consistent with the product-specific data sets, hence they can be compiled or deduced from each other. Traditionally, these data are generated by the

ERP system in a strictly centralized fashion. Detailed planning and adjustments to the overall scheduling are subsequently done by the MES. With a view to the information distribution sketched above, it seems reasonable to largely remove the planning functionality from the ERP and distribute it on the level below among the "products" that can independently keep track of their processing needs and status. This requires the introduction of an information-oriented "alter ego" for each product, and software agents seem to be a suitable approach for it.

A. *Why agents?*

PABADIS uses object-oriented models and object-oriented software technology to describe and perform automation tasks. The workpiece is seen as an object that has all its necessary information regarding its production somehow embedded or attached. It seems natural to use an Intelligent Software Agent for such a purpose [2]. Software agents are the real-world manifestation of object-oriented and distributed functionality [3]. The combination of software agents and physical instances (like machines or the workpiece in our case) is sometimes also referred to as "holon" [4, 5], however, we prefer to stay with the term "agent".

Using software agent technology helps to design the system in a natural way that is easy to comprehend. Agents can be assigned to the physical instance they are responsible for. Other agents can represent and manage machinery or resources [6]. These agents inhabit a multi-agent system (MAS) and can cooperate to perform their tasks as shown in Fig. 2.

Cooperation and other methods from distributed artificial intelligence are further advantages that MASs can incorporate. Finally, the option to have mobile code adds another degree of freedom and improves the flexibility of the system. Other approaches like the classical client/server architecture can in principle also be used to implement distributed systems, but do not have the flexibility of agents.

B. *System topology and agent types*

Based on such agent-oriented design a PABADIS system basically consists of a set of so-called BIIOs (Basic Independent Intelligent Objects) that provide meaningful services to the manufacturing process. This definition is deliberately abstract and makes no assumptions about the physical realization. In fact, we can distinguish three different types of BIIOs:

- Manufacturing BIIOs are used for the physical processing of the products. Depending the granularity of the process steps and the envisaged level of abstraction, these entities can be individual machines (like dedicated drilling or milling machines), multipurpose manufacturing cells or full production lines. Even manual workplaces can be included in the system, provided they have a suitable HMI to the communication system.
- Transportation BIIOs link the manufacturing BIIOs and move the workpieces. Depending on the complexity of the plant, there may be one or several independent transportation BIIOs.

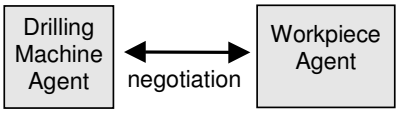

Fig. 2.: A workpiece agent negotiates with a drilling machine agent.

- Logical BIIOs provide computational services and have nothing to do with the physical processing of the products. Instead, they are consulted by the agents for special tasks like complex scheduling algorithms, database search, or the like.

The BIIO community has an interface to the ERP system, the so-called Agency. The task of this component is the creation of the software agents and finally also their extinction upon completion of their task. All these components are connected via a backbone network, as shown in Fig. 3.

The agents scattered in this system are only partially mobile, depending on the necessities of their tasks. From a functional point of view, three agent types exist:

- Residential Agents are the interface between the BIIOs and the agent community. They are stationary and tied to their specific BIIO. Their task is to provide information about the capabilities of the BIIO and to allow other agents to access the respective resources.
- Product Agents are associated with the actual workpieces being produced. They control the manufacturing process from the viewpoint of the individual product and take care of scheduling, resource allocation, or reporting. To this end, they have to be mobile.
- Plant Management Agents finally organize the manufacturing process from a system-wide perspective. Their tasks include quality management, reporting, and the like. These agents are not necessarily mobile. If they are stationary, they reside in the agency and perform their tasks by message exchange with the agent community.

All system components that participate in the agent community, i.e., BIIOs and the agency, need an agent container or agent host providing a runtime environment for the agents. This environment has to provide suitable communication facilities and abstraction from the underlying operating system and hardware, as well as

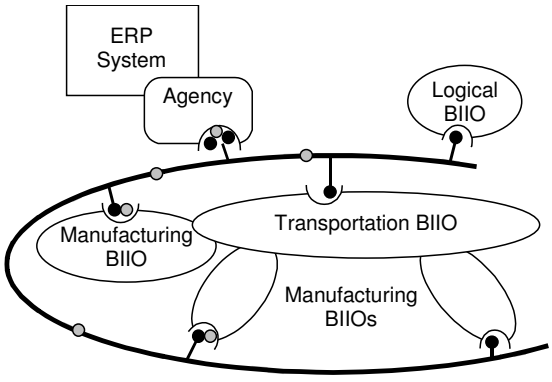

Fig. 3: Topology of a PABADIS plant. The gray balls signify mobile software agents, the black balls are residential agents.

agent mobility. At present, the two Java-based systems Grasshopper [7] and LANA [8] are under consideration.

C. *Workflow in a PABADIS plant*

The manufacturing process according to the PABADIS idea starts with the generation of a conventional production order by the ERP system. This order comprises the sequence of required processing steps together with the appropriate parameters. The production order is passed to the agency, where it is translated into a product agent and joins the multi-agent system.

Step by step, the product agent executes its production plan. The basic procedure it follows is always the same: it consults a lookup service, which present in the network to find the BIIOs that can provide the needed manufacturing service. The lookup service acts as a central service broker and will be dealt with later on. Subsequently, the product agent contacts the residential agents of the BIIOs and asks for information necessary to decide which service provider it should choose. Such information includes the availability (i.e., the earliest possible free time slot for processing), the expected duration of the action, but perhaps also the location of the BIIO for calculation of the transportation costs. Based on these data, the agent selects the "optimal" BIIO.

The selection process sketched here is fairly simple. More complex scheduling procedures would involve the communication of the agent with competitors to dynamically change the resource allocation and create different sequencing and dispatching plans. Although the development of distributed scheduling algorithms is not in the focus of the PABADIS project, such advanced features can be added to the agents at any time by replacing their selection and negotiation modules at creation time. Alternatively, specialized logical BIIOs can provide these services. Note that in order to facilitate the upgrading of the system at a later date, the residential agents (which necessarily have to be supplied by the manufacturers of the BIIO hardware) are expected to be rather unintelligent. They do not play an active role in the scheduling negotiations, they just maintain their local processing sequence once it has been fixed.

Throughout the manufacturing process, the product agent guides the workpiece. Upon completion, it returns to the agency and is destroyed there. The agency then generates a report to the ERP system using the data the agent has collected on its way through the production (if so desired by the ERP system in the production order). In parallel, plant management agents are created by the agency to fulfil specific control or supervision tasks that are not related to individual products.

III. NETWORK REQUIREMENTS

Multi-agent systems like the one that is going to be used for PABADIS are usually based on some sort of network operating system (NOS) [9] that provides services like communication, directories and migration. This NOS requires a communication network that offers

- peer-to-peer communication,
- message-oriented services, and
- uni- and multicast transport services.

Using such services, the NOS can implement directories where the agents can find each other and other agent-relevant services. PABADIS will utilize IP-based LANs, in particular on the basis of Industrial Ethernet, for its communication. The industrial environment where the application of this system is situated usually demands hard real time characteristics and reliable network connections. Although conventional LANs are not well suited to real time communication, the use of agent technology can support these requirements on application level, as will be shown later in this paper.

Directory services are a key to achieve the desired flexibility in PABADIS, and they are the basis for the plug-and-participate environment mentioned in the introduction as one main characteristic of PABADIS. By the term "plug-and-participate", we primarily mean the self-organization of services in the network from an application point of view, i.e., communicating partners can use each other's services without manual configuration of the respective interfaces. Independent of this aspect is the low-level network setup for the individual devices, which is frequently subsumed under the term "plug-and-play", but not necessarily the primary goal of PABADIS.

At any rate, plug-and-participate as we understand it requires some sort of "middleware" layer enabling the abstract formulation of distributed objects and services. There are several middleware technologies available today, which shall be briefly screened for their suitability.

Java/RMI (remote method invocation), DCOM (distributed component object model), and CORBA (common object request broker architecture) allow the specification of distributed objects and their use. However, they have no generic built-in plug-and-participate features. Although they support mechanisms such as naming or lookup services to retrieve the objects, they rely on the user (i.e., the application running on top of the middleware) to properly register the distributed objects. Consequently, it would in principle be possible to implement plug-and-participate mechanisms with these paradigms, but at the expense of additional effort required for every application in the distributed system.

To date, there are three popular technologies that provide so-called service discovery protocols, which is actually what we need if we want plug-and-participate functionality. UPnP (universal plug and play) is an open system pushed primarily by Microsoft and mainly targeted at connecting appliances and PCs. It is based on additions to the TCP/IP suite rather than on application level protocols to create platform-independence. Jini is based on Java/RMI to support ad-hoc networking. The "run anywhere" feature of Java makes it independent of the platform as long as a Java Virtual Machine is available. The third technology is Salutation, which is designed to be fully independent of platforms, operating systems and even transport protocols. Other approaches are the SLP (service location protocol) developed by IETF and the SDP (service discovery protocol) used in Bluetooth. Comparisons of all these protocols and additional information can be found in [10, 11, 12] and the references cited therein.

For PABADIS, Jini [13] was selected as the middleware of choice. The reason for this decision was a rather pragmatic one. The agent systems considered are based on Java to permit mobility. Hence Jini snugly fits in and

complements the system. What is actually used within the framework of PABADIS is Jini's lookup service. A new BIIO (to be precise, its residential agent) that is attached to the system registers with the lookup service and announces the processing services it can provide. The product agents in turn use the lookup service to find appropriate BIIOs as described in the previous section (see also Fig. 4). It is noteworthy that many pure MAS also have some sort of lookup service. This procedure is usually performed by some facilitator-agent or broker-agent [14]. However, there is no self-registration in these systems, and this is why we use Jini. The only problem with Jini is the a priori unknown resource consumption, which depends on the size of the interfaces (the Jini proxies) specified by individual services and may impair the use of Jini on devices with limited memory resources. Since exactly such devices are currently of particular interest in plant automation systems, a Jini derivative will be used for the actual implementation [15].

If we return to the beginning of this section, we could state that Jini is the NOS of choice for PABADIS. However, Jini does not provide all aspects that one would expect from a NOS, so we shall see it as only one layer of the NOS. In particular, Jini does not provide a migration service for mobile agents, or network- and CPU-load balancing for distributed applications. Jini just offers a sophisticated lookup service for services and data, the remaining services needed by the agents must be provided by the MAS framework.

Last not least, another requirement for the PABADIS network is the possibility to form topological hierarchies. Subnets of the system should have independent characteristics like bus load or physical properties. Some parts of the plant might require low-power nodes for hazardous environments, others high-speed backbones. Traditional IP routing gives a well proven, simple and robust manner for hierarchies, which is also a convincing argument in favor of the envisaged IP-based network infrastructure.

IV. TO MOVE OR NOT TO MOVE

The PABADIS project aims at robust and flexible distribution of functionality, and we have been talking much about mobile software agents in the preceding sections. However, it should be stressed in this context that mobility of software code is not a dogma. Conversely, it should be applied with care, and only, when and where it is reasonable in order to improve the performance and quality of the system. The question is, if it is necessary or unnecessary for PABADIS to use mobile software agents [16] that can roam the automation network.

Let us hinge the discussion of this focal question upon an example. We suppose a PABADIS agent is in charge of having a workpiece painted. It knows the sequence and color of paints that are to be sprayed and gives the corresponding orders to the agent that is in charge of controlling the painting machine. For the sake of simplicity, we also presume that the machine is controlled by some PLC that can host software agents.

A product agent as defined in section II would be a mobile agent that migrates to the PLC of the machine to perform local communication. By contrast, a stationary agent would stay on its agent host (most likely the agency

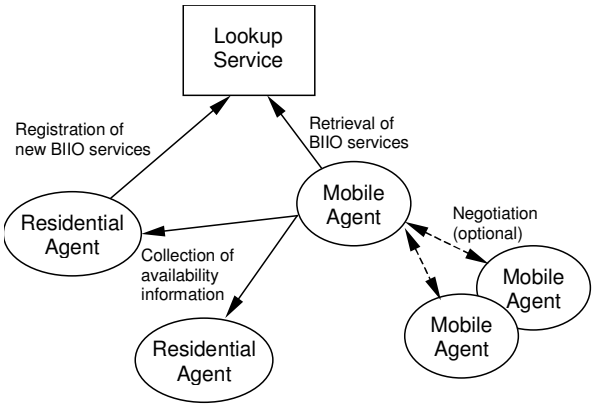

Fig. 4.: Communication relations between the lookup service and the various software agents.

shown in Fig. 3) and communicate with residential agent of the PLC via the automation network. Generally, the decision against or in favor of mobile agents is usually based on the relation between communication load and agent size. If an agent has the ability to search a large database in some specialized manner, it can make sense to send this agent to the database instead of having it search via the network if the size of this agent is smaller than the data that has to be explored.

In our case, the commands the product agent sends to the machine agent in order to control the execution of the painting process are rather short messages that can easily be transmitted via some message. So we do not expect any bandwidth reasons for using mobile agents. Considering the bandwidth can even lead to the insight that mobile code is more harmful for the performance of the entire system than beneficial. If the agent is mobile and equipped with every processing information needed for the product, it also has to carry the PLC program that is to be executed on the painting machine. Every time the agent moves to another machine, it has to take the program with it. Consequently, although the program is used only once, it adds to the network load each time the agent changes is working place. On the other hand, if the product agent is stationary, the program needs to be uploaded to the PLC only when it is to be executed – a tremendous reduction in network load and a good argument *against* mobility.

Apart from the network performance aspect, having the product agents hosted on some powerful agent host computer gives a number of additional advantages:

- The system administration of centralized architectures is sometimes easier than dealing with distributed systems.
- The system is less complicated.
- Debugging and supervision is easier.
- One powerful agent host might be cheaper than a number of PLCs that can host mobile agents.

These are typical advantages of centralized architectures. In our case they also apply to a non-mobile architecture. Highly distributed systems, by contrast, show their benefits when the system tends to become very large or has to be very flexible.

The requirements for an automation network within the PABADIS concept cannot be fully satisfied with a

centralized and non-mobile architecture. Suppose a network infrastructure that is subject to frequent maintenance and that is situated in a harsh environment. The reliability and availability of such a network might not be sufficient for an automation task. Idle machines and machines waiting for commands on broken communication channels cause delays and therefore increased costs, not to speak of potential safety problems when online communications fail. One way to overcome such problems is to design the network in a fault-tolerant and robust manner with redundant paths, robust physical layers and flexible routing.

PABADIS takes another way. By using mobile software agents, equipped with all necessary schedules, plans and information to perform its task, it is possible to temporarily lose connection to the rest of the network and to finish the task locally. This independence of the source is an important argument *in favor* of mobility.

Mobile agents are preferably used for networks with low availability (like dial-up connections). The networks used in PABADIS are actually not of this type, they are considered to be reliable and always online. But the required flexibility and robustness results in an automation concept that can cope with unreliable network connections.

One key point in the context of reliability is atomicity, meaning that an action is either fully executed or not executed at all. Sending an agent to a PLC is such an atomic operation. Once the agent is there, it is fully capable to perform its task, because the subsequent operations are local. This is not the case if the product agent is stationary on a different host, unless the actions the agent performs is restricted to a simply "start" command of a previously uploaded PLC program. However, if the interaction of the product agent with the PLC are more complex (which we have to presume in a general automation framework), we can guarantee the completion of a task only if we have local control. The consideration as to which processing steps ought to be atomic is by the way one design criterion for the definition of the BIIOs in an actual PABADIS plant.

It should be noted that the migration of the agent in itself is also an atomic operation. If the connection is lost during migration, the received code is not a complete agent and therefore rejected. Having the agent local at the PLC it controls or interacts with also brings about a number of advantages:

- There is no need for a hard real-time protocol on the network since control commands relevant for the processing task are no longer sent via the network.
- Agents can migrate "in advance" when network load is low.
- Agents can be electronically signed and increase system security as they encapsulate their information.
- The PLC can perform a basic plausibility check of the agent script before it is executed.
- The computational load is distributed to multiple PLCs and does not require a powerful agent host node.
- The system has no obvious "single-point of failure" like an agent host.

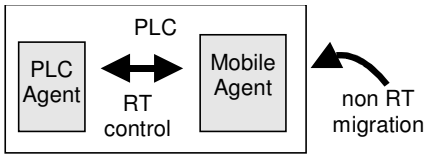

Fig. 5.: A mobile agent migrated to a PLC

The main advantage beside the robustness against network errors is the relaxed real-time requirements for the network protocol due to its local operation as depicted in Fig. 5. PABADIS' IP-based LANs have no or only insufficient real-time characteristics, unless special quality-of-service extensions are used. Therefore the typical low-level control commands are eliminated from the communication over the network, and also other tasks can be done based on non-hard real-time communication. If agents need to coordinate each other they will simply exchange schedules with some common time base. This mitigates the constraints for the network and makes coexistence easier for other applications. Yet it is sound not to run, e.g., office applications over the same network that need to transfer large amounts of data. Although possible, this will increase the network load and adversely affect the performance of the automation system.

In total, the arguments that speak in favor of mobility are slightly stronger. A mobile agent would come along with our workpiece to the painting machines, would operate these machines, log statistics, and do other things on behalf of the workpiece like paying virtual money to the machines for time and paint that the product price will be based on afterwards. To avoid unnecessary network load, PLC programs and other common knowledge needed by more than one agent can be administered and provided by specialized database BIIOs. They provide a system-wide library of knowledge and information that can be used by all members of the system. The product agent only carries links to the item it needs for a certain processing step. Before it executes this subtask, it moves to the database BIIO, fetches, e.g., the respective PLC program and moves on to the actual machine. This ensures that large programs are transferred only when they are really needed.

Once again, the basic functionality of distributed systems can as well be achieved with non-mobile software agents. The autonomous behavior and the encapsulation of data and commands is independent of the mobility of the agents. However, making them mobile simply increases the flexibility and robustness of the system.

V. CONCLUSIONS

PABADIS tries to enhance plant automation by using state-of-the-art communication technology and distributed functionality. Instead of endangering the robustness of the system, this increased complexity can support the autonomy and fault tolerance of the automation application.

The consistent implementation of distributed applications and agent-oriented software helps to achieve a flexible, modular and scalable system. The positive characteristics of multi-agent systems like self-

organization and adaptability are important contributions to modern automation networks.

The decision whether or not mobile agents should be used is an ambivalent and controversial one. Actually almost every task can be done with or without mobile agents. At first sight, mobile agents just increase the complexity of the system, which should be avoided. The main final reason for the use of mobile agents is their atomic and reliable function. In addition, their mobility reflects the real-world situation in a plant better than static agents do, which results in a clearer and more comprehensible model for the algorithms and the corresponding software.

REFERENCES

[1] PABADIS (Plant Automation Based on Distributed Systems), IST-1999-60016, http://www.pabadis.org.

[2] J.P. Müller: "The design of intelligent agents: a layered aproach", Lecture Notes in Computer Science, Vol. 1177, Springer Verlag, Heidelberg, 1996.

[3] J. Bryson, B. McGonigle: "Agent Architecture as Object Oriented Design", in Intelligent Agents IV, Springer Verlag, 1998.

[4] L. Bongaerts, J. Wyns, J. Detand, H. Van Brussel, P. Valckenaers: "Schedule execution for a holonic shop floor control", in Proceedings of European Workshop on Agent-Oriented Systems in Manufacturing, 1996, Berlin, Germany, 1996.

[5] M. Fletcher, S.M. Deen: "Fault-tolerant holonic manufacturing systems", Concurrency and Computation Practice & Experience, vol.13, no.1, pp.43-70, 2001.

[6] J. Bredin, D. Kotz, D. Rus: "Market-based resource control for mobile agents", Second International Conference on Autonomous Agents, Minneapolis, MN, May 1998, ACM Press, pp. 197-204, 1998.

[7] http://www.grasshopper.de

[8] C. Bryce, J. Vitek: "The JavaSeal Mobile Agent Kernel", in D. Milojevic (ed.), Proceedings of the 1st International Symposium on Agent Systems and Applications, Third International Symposium on Mobile Agents (ASAMA'99), Palm Springs, May 9-13, 1999, ACM Press, pp. 176-189, 1999.

[9] P.H. Enslow Jr., T.G. Saponas: "Parallel control in distributed systems- a discussion of modes", in David J. Evans (ed.), Parallel Processing Systems, Cambridge University Press, Cambridge, 1982.

[10] C. Bettstetter, C. Renner: "A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol", Sixth EUNICE Open European Summer School: Innovative Internet Applications, Twente, Netherlands, September 13-15, 2000, http://www.lkn.ei.tum.de/~chris/publications/eunice2000-slp.pdf.

[11] C. Lee, S. Helal: "Protocols for Service Discovery in Dynamic and Mobile Networks", www.harris.cise.ufl.edu/projects/publications/servicediscovery.pdf

[12] G.G. Richard: "Service advertisement and discovery: enabling universal device cooperation", IEEE-Internet-Computing. Vol.4, No.5, p.18-26, 2000.

[13] B. Venners: "Jini: New technology for a networked world", in Javaworld 06/99, 1999.

[14] Nishida, Takeda: "Towards the Knowledgeable Community", Proceedings International Conference on Building and Sharing of Very-Large Scale Knowledge Bases '93 (KBKS '93), 1993.

[15] S. Deter, K. Sohr: "Pini - A Jini-like Plug&Play Technology for the KVM/CLDC", Proceedings of Innovative Internet Computing Systems, Springer, pp. 54-66, 2001.

[16] J. Kiniry, D. Zimmerman: "A hands-on look at JAVA mobile agents", IEEE Internet Computing, Vol.1, No.4, 1997.