Multi-Agent System for Adaptation of Distributed Control System

Dariusz Choiński and Michał Senik
Silesian University of Technology, ul. Akademicka 16, 44-100, Gliwice, Poland

Keywords: DCS, Multi-Agent Systems, Ontology, Java, .Net, JADE, FIPA, OPC, XML, NHibernate, Hybrid Systems, Concurrent Programming, Knowledge Sharing, Learning.

Abstract: A solution based on Multi-Agent Systems properties has been proposed. The presented structure is designed to Distributed Control System behaviour abstraction and encapsulation of the technical characteristics of its individual elements such as OPC (OLE for process control) servers. An ontology facilitating the creation of user interface for Multi-Agent System environment has been proposed. This ontology is based on a set of concepts and symbols understandable for the operator and the knowledge defining the hierarchical structure of object. Presented solution is not only a conception but it is a real, cross platform implementation based on the both Java and .Net programming platform. It practically shows how new programming solutions, tools and methodologies can be integrated and reused to solve real life, practical automation system problems.

1 INTRODUCTION

Distributed Control Systems (DCS) are one of the cornerstones of modern industry, which aim to achieve high flexibility and customization of production. Such objectives require reconfiguration of the control system. However, the idea of software architecture and modern measurement and control systems are not prepared conceptually for such purposes (Strasser et al., 2011). Adaptation of DCS to changing requirements and configuration involves the integration of various components of their architecture and is a major programming challenge (Vyatkin, 2011). In case of such adaptation the programmer is forced to define and reallocate resources of DCS. However, this operation is usually performed only in the design or upgrade of the system. This is because such activities are required for system designers and not their users. Therefore, the integration is carried out mostly in the connection of software components through the communication protocols. Providing a dynamic adaptation, DCS during its operation must address the problem of operating mechanism for data sharing and allocation of resources.

The solution presented in this paper concerns the use of the OPC standard and Multi-Agent System (MAS) (Metzger and Polakow, 2011). OPC standard (Iwanitz and Lange, 2006) was established as a method for efficient communication between automation devices and systems. One of the basic specifications is the OPC DA (Data Access) specification. It defines the communication between the client and the server hosting the real time process data. Data Access Clients have access to data from the automation system via Data Access Servers. Communication interface between the client and the server is completely independent of the physical data source. OPC DA specification also defines two main structures for describing data shared by the server. These are the namespace (Namespace) and OPC objects (Figure 1). Namespace is used to produce data structured in a tree structure, provided by the OPC server. The structure of the OPC objects is created by the client user. OPC object, within the established structure, is attributed to identifiable characteristics, such as: the value of the corresponding variable, time of measurement, quality measurement and other (Choinski and Senik, 2010, 2011).

The idea of communication between agents, using serialization of speech acts requires the creation of an ontology that allows the partition of the message by the agent, so that the intentions of the objectives are clear and unambiguous. At the same time it should be a feature of the ontology that it is as easily processed by both man and machine. The interface of the user wanting to work with multi-agent environment should be located in a way, so that it operates on the same ontology as agents operate on. The effect of this reasoning is the

Choiński D. and Senik M..
Multi-Agent System for Adaptation of Distributed Control System.
DOI: 10.5220/0004034902060211
In *Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2012), pages 206-211
ISBN: 978-989-8565-21-1

concept of using, for the development of ontology, a set of terms and symbols understood by the operator and used by him for categorizing and prioritizing the knowledge of the object. The rationale for the development of the basis of ontology is the choice of language that is independent from the specific area of ontology application. Therefore, it has been decided to incorporate the definition of structures utilizing the OPC standard into the ontology dynamic adaptation of the control system (Choinski and Senik, 2010).

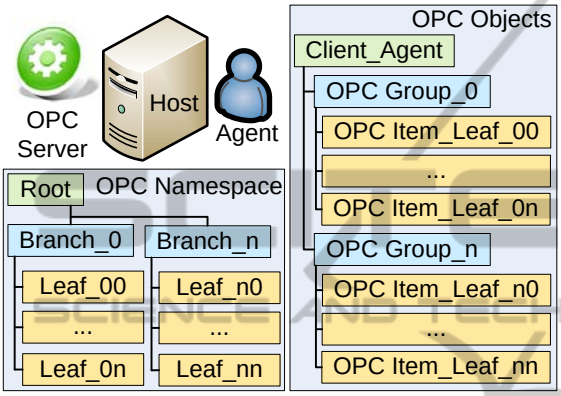

Figure 1: OPC Server hierarchical structure and MAS integration scenario.

Structure of the transmitted information is particularly important in the use of Multi-agent technology in control and design of control systems. The set of concepts, which in this case is a description list of the data points, used and controlled variables in the whole process of designing the control system and its software is practically constant. What changes mainly is the structure of mutual connections and the structure of the information used for decision making and activities related to the control.

An obvious advantage of the system of agents that communicate using messages based on a universal definition of a standard FIPA (Bellifemineet al., 2007) is the ability to remote boot services, regardless of the particular software implementation. Within the commonly used protocols in the DCS, it is necessary to know the structure of instances of the individual objects performing services or storing information (attributes). However, for the MAS, the fact that there is no need to have this knowledge is laden with the necessity of contribution to the development of an ontology resulting in complexity comparable to the development of the transmission protocol. A class of distributed system is determined by the technical sophistication used in the transmission protocol. The same way, the "intelligence" of agents may not be better than the ontology designed for those agents. Thus, the principle of creating ontology for communication in MAS is identical with the description of the transmission protocol. Similarly, the terminology glossary should be defined describing terms used by agents in the area of communication, hence the definition of agent resources from the network side. In addition, a set of defining the structure and relationships between concepts must be defined, that is their semantics and hierarchical structure.

2 MAS BASED APPROACH TO THE DCS INTEGRATION

Resources as a part of the greater DCS usually are treated as a source of a various different pieces of information that needs to be integrated and analyzed to be meaningful. This data is usually stored in a resource's hierarchical memory structures.

Process of the fast data collection and analysis is an essential activity for each integration system that as a result allows for the proper and efficient control and maintenance. Both data and resource allocation can change over time. It is a dynamic process and because of that each such integration system must follow strict rules to meet those challenging conditions. Inability to respond to those conditions drastically lowers down the integration system quality and performance. In traditional integration systems resources modifications are controlled manually. This is because those systems are not able to evolve. Each such system is designed to meet only the current situation while the synchronization process must be performed manually. Synchronization process must take place always whenever the resource changes its state otherwise the problem of data integrity arises. Normally the state of the resource changes due to data structure reorganization or resource status modification. Each such modification ought to trigger integration system response which should notify the operator. This is where the integration system's reasoning processes stops leaving all the synchronization process up to the user. In order to enhance this activity integration system ought to be designed to be more autonomous. Autonomy of the integration system is a complex task because it requires additional analysis of the DCS as a whole from the perspective of the each single integrated resource. Performed analysis ought to treat resource as a finite

state machine and result in explicitly defined state set that would cover its whole functionality. It is important noticing that those states must not lead to the integration system deadlock. Each state should have its predecessor and successor defined. This as a result forms integration system states reachability graph (Peterson, 1981). Moreover, each defined state should describe how integration system will react and respond to the resource behaviour. Integration system designed in this way should also present an open and scalable architecture that would allow the system to respond to the dynamic DCS resources changes to evolve and grow in time.

It is impossible to address all the mentioned features using traditional approach during DCS integration system design and implementation. All features however can be easily introduced in the MAS. Another crucial aspect of the MAS is the presence of the formalized system knowledge (Figure 2) which allows for independent reacting and reasoning activities. MAS's knowledge fundamentals are usually modelled during system design stage prior to any real implementation. This knowledge is usually referred to as ontology.

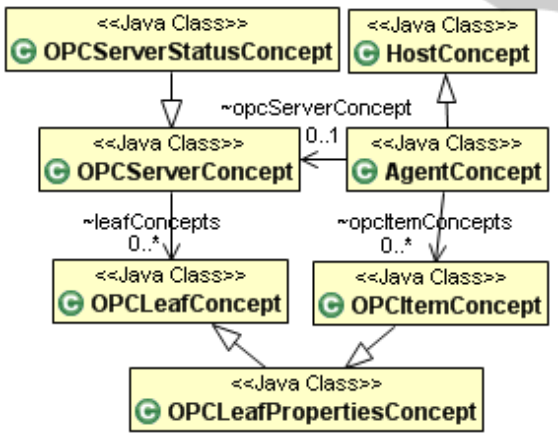

Figure 2: Ontology based knowledge representation – OPC Server hierarchical data structure.

Ontology describes both static properties of the system and its runtime states in which the system can reside as well as the conditions of transitions between those states. Ontology can also model the integrated automation system resources (Figure 1, 2). Based on the ontology, MAS agents share their knowledge and work together cooperating over the integrated system's problems (Choinski and Senik, 2010, 2011). Technically ontology is a set of very simple and much more complex rules such as concepts, predicates and actions which can be created in many available programming environments such as Java or .Net. (Choinski and Senik, 2010, 2011). Each ontological expression can be organized in a hierarchical structure which means that simpler entities can be nested in more complex entities. Agent's reasoning capabilities reuse those ontological structures during cooperation processes over many different integrated system resources.

3 ONTOLOGY BASED KNOWLEDGE MANAGEMENT

Having established the DCS ontology integration and implementation phase can start. In order to properly implement the system's knowledge management mechanisms additional architectural aspects has to be introduced beforehand. It is worth mentioning that ontology is not only the system description, its expressions are also reused during messaging and knowledge management processes. MAS's ontology concepts are the most basic carriers of the various different pieces of information. Each concept is explicitly described via the ontology schema definition that specifies the rules under which the concept ought to be structured. In most situations concepts are used to form complex hierarchical data structures that basically are passed between sender and receiver asynchronously updating their states (Figure 2). In order to efficiently reuse those concept definitions during runtime each concept must fulfil additional conditions firstly drawn up by the object oriented (OO) methodology. Each concept is nothing more than a raw memory object that implements two methods that should explicitly formalize its uniqueness (HashCode method) and equality (Equals method) (Sierra and Bates, 2008). This aspect is consistent amongst the two currently leading OO programming platforms such as Java and .Net. Basically knowledge management is a comparison of two different ontological concepts. Knowledge sharing and management process (Figure 3, 4) is always a two side task that occurs always whenever two different agent entities enter into the cooperative mode in which various different messages are passed between them (Wooldridge, 2002). Each message carries an ontological expression that can be reused by the receiver agent to perform certain tasks and to update its current knowledge. Agent knowledge ought to be updated whenever an agent realizes that passed ontology expression modifies its current state. As a response receiver agent ought to generate acknowledge message containing either original message content or received message status flag.

This message traverses back to the sender agent modifying its knowledge about the receiver agent. Each time new message is send, message's ontological content is compared against the knowledge; sender agent gathered over time on the receiver agent.

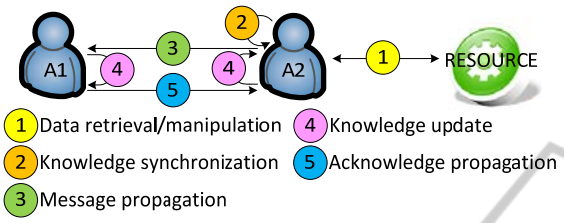

Figure 3: Ontology based knowledge sharing and management general concept.

That way only needed data is send which drastically minimizes both network traffic workload and agent message processing time (Figure 3, 4).

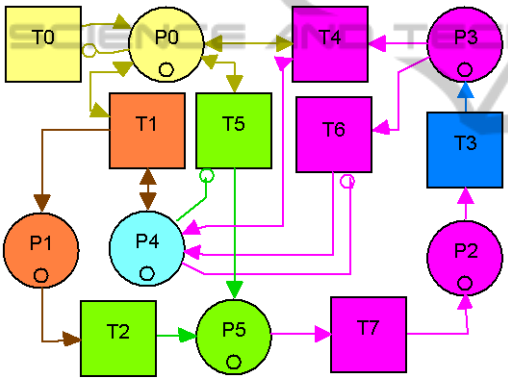

Figure 4: Ontology based knowledge sharing and management – Petri Net.

4 MAS PLATFORM DESIGN

Each MAS agent by performing its own tasks acts on a behalf of its parent MAS platform. In order to efficiently realize their tasks agents has to be designed to cooperate with the assigned resource as close as possible. Therefore agents must have the ability to reach the resource physically i.e. agent must be mobile (Wooldridge, 2002). That way MAS resource agent can perform all the necessary operations faster in data origin place sending only the results of those operations. The extension of this approach would be to design a mobile supervisory agent that after reaching the source of the data initializes whole package of various different agent types that would cooperate with each other over the given resource in one place forming a small, parent MAS agent colony. Such agent colony can be treated as a partially autonomous MAS that can act on a behalf of its parent MAS. This approach allows for a dynamic agent to resource allocation which results in an anticipated on a design stage, MAS evolution capability. This however could not be achieved without the presence of agent environment discovery mechanisms which are vital for the MAS management (Bellifemine et al., 2007). In order for the each newly created agent utilize existing MAS platform knowledge it is required to manifest itself in the platform via its name, address and services names. It is also required to design and reuse the MAS platform subscription mechanism because it is much faster to obtain all the necessary pieces of information from the various different MAS platform points without asking.

Agent discovery and subscription mechanisms are the key concepts of the agent methodology that states that agents should not only react in response of the external stimulus but take over the initiative over the integrated system as well. This explicitly means that agents are not simple and reactive components but because of their features agents are far more sophisticated and intelligent entities entitled to reason about the DCS integration system and enabled for the various different proactive behaviours inside the MAS platform.

Design of the MAS integration system should also involve layered aspect of the system architecture. This is because MAS integration system is not flat as it is purely hierarchical. Deep analysis reveals that MAS integration system should be divided into three main layers (Choinski and Senik, 2011) i.e.: direct cooperation layer, agent internal cooperation layer and user cooperation layer. Each layer of the MAS integration system is responsible for a specific set of activities. Direct cooperation layer is responsible for the direct agent to DCS resource cooperation. It is a backbone of the MAS integration system because it provides whole system with various pieces of the process

information. Moreover it is a place in which initial stage of the process analysis can be performed. Agent internal cooperation layer forms all required MAS integration system analysis and reasoning fundamentals. User cooperation layer stands between the user and DCS forming GUI that enables human operator for the indirect system cooperation including complex output data analysis. Direct cooperation layer and agent internal cooperation layer are hidden in the background of the MAS integration system whereas user cooperation layer is exposed entirely to the human operator making the system maintainable, accessible and transparent. Each of the presented design aspects makes the MAS based systems far more scalable and robust solutions in comparison to the traditional solutions that presents neither autonomy nor openness and scalability. Each MAS based solution is enabled for the self management activities that enforce their evolution over time. Each such activity drastically increases DCS performance and quality.

5 MAXS PLATFORM

MAXS (Multi Agent Cross Platform System) platform is a real time processing, hierarchical, multilayered, MAS integration system capable of dynamical adjustment to the existing information structure (Leduc, Lawford, Dai 2006), (Choinski and Senik 2011). It wraps over the existing JADE framework (Bellifemine et al., 2007) using a set of newly implemented agents thus creating a complex MAS. MAXS can function on one or more remotely controlled hosts, which may also vary in time. In the current development stage, MAXS platform can interact with various different OPC DA and database servers. In addition to the JADE FIPA (Bellifemine, Caire, Greenwood, 2007) specific communication, MAXS platform agents simultaneously reuse integrated databases as a parallel redundant communication channel. Normally, MAXS reuses database communication channel to store or update both raw process data obtained from the system and platform configuration settings that can be reused during system restore. Integration with various database servers is achieved by means of the NHibernate (Dentler 2010) entity framework. NHibernate framework is designed to be used over the .Net platform only and in order to utilize it MAXS platform was additionally integrated with the .Net platform by means of the JADE Sharp. JADE Sharp is a JADE add-on which comes as an additional .dll library module which enables creation .Net agents compliant with the JADE framework. Original version of the JADE Sharp was proposed by the TILab (Bellifemine, Caire, Greenwood 2007) For the MAXS purposes, JADE Sharp add-on module was strongly modified, refractored and extended to fit the needs of the MAXS platform.

In order to establish efficient cross-platform communication each MAXS agent reuses common XML based messaging mechanism. The MAXS platform establishes efficient cooperation with various different OPC Servers through the Java Native Interface (JNI) (Liang 1999).

MAXS platform (Figure 5) in the current development stage consists of nine different types of agents: Supervisory Agent (SA), Node Agent (NA), Port Scanner Agent (PSA), Network Scanner Agent (NSA), Management Agent (MA), OPC Agent (OA), Discovery Agent (DA), Lifecycle

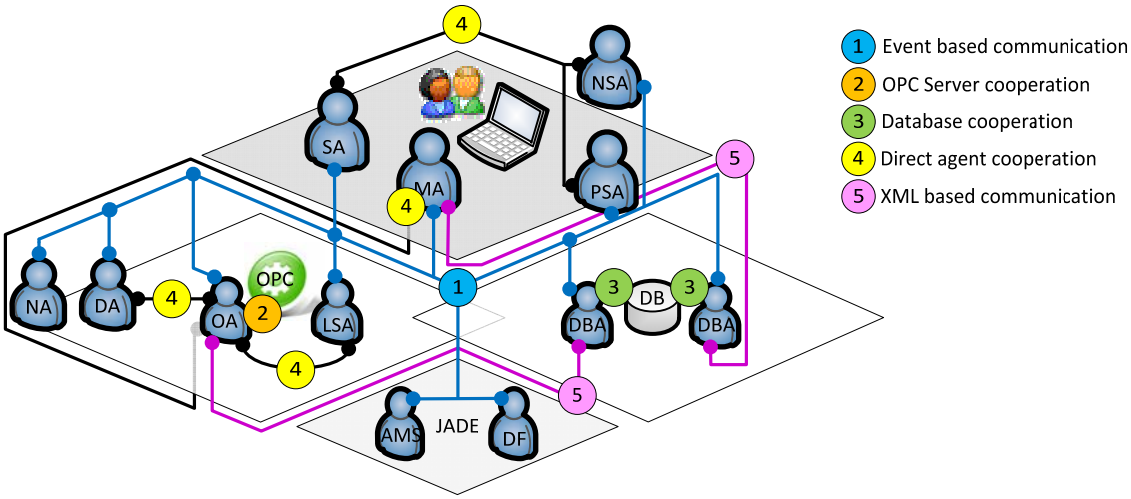

Figure 5: Implemented MAXS platform general design concept.

Supervisory Agent (LSA) and Database Agent (DBA). Complete description of all created agents can be found in (Choinski and Senik, et al., 2010, 2011). MAXS layered architecture (Choinski and Senik, 2011) presents openness and flexibility which implies self management nature. It is a key concept that satisfies dynamic conditions of the integrated environment.

Self management functionality enables MAXS platform agents to address both integrated environmental issues and MAXS platform specific events such as unexpected agent's internal errors or possible instabilities. Such approach allows fast situation diagnosis and proper MAXS counteraction. MAXS self management ability is a fundamental feature that guards and preserves system data integrity and quality. Each MAXS agent shares similar capabilities making them more compliant and consistent. Those capabilities are the key concepts of the internal platform architecture (Choinski and Senik, 2011).

To complete assigned tasks MAXS agents reuses common set of fully customized and concurrent behaviours (Choinski and Senik, 2010, 2011). Each one extends common mechanism of knowledge sharing and management that utilizes MAXS platform ontology.

MAXS reasoning model is based on the subscription and notification mechanisms as well as on the agent knowledge which is gradually gathered over time from other platform agents.

6 CONCLUSIONS

The dynamic of the DCS is a normal situation which has to be automatically handled in all integration systems. Traditional approach to the DCS integration seems to be insufficient to cover most of the real life integration problems. Each DCS grow and extend in time and the same should happen with the integration system. Such system must preserve an open architecture. It must be scalable enough to follow the DCS extensions. The solution to address those problems lays in adoption of the proper methodology and architecture.

MAS based solution is the right choice that meets the requirements of integration of the dynamic environment. MAS DCS integration is based on the proper resource knowledge management. Presented ontology based knowledge management is a generic mechanism. It helps acquire all needed pieces of information based on which analysis processes can be performed. Ontology always gives a meaning to each tiniest piece of information obtained from the integrated system thus making it more understandable and readable for both human user and agent entity.

ACKNOWLEDGEMENTS

This work was partially supported by the Polish Ministry of Scientific Research and Higher Education N N514 471539.

REFERENCES

Strasser, T., Zoitl, A., Christensen, J. H., Sunder, Ch., 2011. Design and Execution Issues in IEC 61499 Distributed Automation and Control Systems. *IEEE Trans. on Systems, Man, and Cybernetics—Part C: Applications and Reviews.* Vol. 41, pp. 41¬51

Vyatkin, V., 2011.IEC 61499 as Enabler of Distributed and IntelligentAutomation: State-of-the-Art Review. *IEEE Trans. on Ind. Inform.* Vol. 7, pp. 768¬780

Metzger, M., Polakow, G., 2011. A Survey on Applications of Agent Technology in Industrial Process Control. *IEEE Trans. Ind. Inform*. Vol. 7, pp. 570¬581

Iwanitz, F., Lange, J.: OPC – *Fundamentals, Implementation and Application.* Huthig Verlag Heidelberg (2006)

Choinski, D., Senik, M., 2010. Collaborative Control of Hierarchical System Based on JADE. In: Y. Luo (ed.), *CDVE 2010, LNCS.* Vol. 6240, Springer, Heidelberg, pp. 262-269.

Choinski, D., Senik, M., 2011. Multi-Agent oriented integration in Distributed Control System. *In: J. O'Shea et al. (eds.), KES-AMSTA 2011, LNAI.* Vol. 6682, Springer, Heidelberg, pp. 231-240.

Bellifemine, F., Caire, G., Greenwood, D.: *Developing multi-agent systems with JADE*. John Wiley & Sons, Chichester (2007).

Peterson J. L.: *Petri net theory and the modeling of systems.* Prentice Hall (1981)

Sierra, K., Bates, B.*: Sun Certified Programmer for Java 6 Study Guide*, McGraw-Hill (2008)

Wooldridge, M.: *An Introduction to Multiagent Systems*, John Wiley & Sons Ltd (2002)

Leduc, R. J., Lawford, M., Dai, P.: Hierarchical Interface-Based Supervisory Control of a Flexible Manufacturing System. *IEEE Transactions on Control Systems Technology*. 14, 654-668 (2006)

Dentler, J.: *NHibernate 3.0 Cookbook*. Packt Publishing Ltd, Birmingham. (2010).

Liang, S.: *The Java Native Interface Programmer's Guide and Specification*, Addison-Wesley (1999).