Multi-Agent Systems for Industrial Diagnostics

Martin Albert[1], Thomas Längle[1], Heinz Wörn[1], Michele Capobianco[2], Attilio Brighenti[2]

[1]Institute for Process Control and Robotics,
University of Karlsruhe, D-76128 Karlsruhe
{albert; laengle; woern}@ipr.ira.uka.de

[2]S.A.T.E Srl.
Systems & Advanced Technologies Engineering s.r.l
Santa Croce 664/a, 30135
{michele.capobianco; attilio.brigenti}@sate-italy.com

Abstract: Industrial diagnostic systems aim at anticipating the occurrence of failures or, should failures have occurred, at detecting them and identifying their cause. Similar to a process control system, a diagnostic system is closely connected with a specific industrial application. In case of large and complex industrial systems, such intimate link translates into the need to satisfy requirements for modularity, flexibility, and adaptability. This paper looks at the requirements connected with modern industrial applications of diagnostic systems. It introduces a "multi agent" software architecture as a possible solution to better tackle the variety of problems connected with building such systems.

Keywords: Agents, Complex systems, Co-operation, Co-ordination, De-centralised systems, Diagnosis, System architectures.

1. INTRODUCTION

The efficiency and reliability of modern diagnostic technologies increase steadily and they are getting more and more suitable to diagnose even complex and distributed industrial applications. To transfer these methodologies into the industrial world, they either have to be integrated within an existing process control or SCADA (Supervisory Control and Data Acquisition) system or, as an alternative, an entire diagnostic system has to be developed from scratch. In both cases, the software developers are faced with a complex, distributed and highly flexible industrial environment without having proper tools to support the development process.

A possible alternative approach builds upon the recent tendency to adopt "component-based architectures" in the industrial area (Balakrishnan, 1999). The adoption of a multi-agent architecture is here proposed as an approach to establish a link between the most advanced industrial control architectures, the latest development in software technology and the modern diagnostic approaches. In order to become widely accepted in industry, this software paradigm has to supplement the functionalities of standard agent platforms in order to increase the performance, the reliability and the predictability of the diagnostic process.

To utilise the diagnostic knowledge within a multi agent system, the available diagnostic capabilities have to be "agentified" by hiding all agent specific capabilities. In this manner, powerful diagnostic features will merge to popular industrial diagnostic systems without the detour to develop proprietary and application specific solutions. In this background, the EU funded MAGIC project (MAGIC, 2002) aims to develop a flexible multi-agent architecture to handle different diagnostic methods in parallel.

2. RELATED WORK

Recent research activities all over the world attempt pushing "agent" technology towards industrial applications. The industry on the other hand becomes gradually aware about the potential benefit when employing agent-based software approaches within their industrial environment. The potential tasks for agents range from monitoring and visualisation to very complex tasks like process control and diagnosis.

A general overview about distributed artificial intelligence in industry is given in (Parunak, 1998a). This paper reviews the industrial needs for distributed artificial intelligence, giving special attention to systems for manufacturing, scheduling and control.

The utilisation of a multi-agent approach for treating with industrial applications is reasonable due to the fact that agents are best suited for applications that are modular, decentralized, changeable, ill-structured and complex. Concerning these application characteristics, agent technology can yield a more robust and adaptable solution than one supported by other software paradigms (Parunak, 1998b).

A framework of different diagnostic processes running in parallel to handle large systems is described by Fabre (Fabre et al., 2002). This paper proposes to model the plant system as a graph of interacting subsystems. It provides the theoretical analysis to deal with these subsystems and to merge them for processing the large scale system.

A multi-agent approach with equivalent agents is presented by Letia to monitor and diagnose spatially distributed technical systems (Letia et al., 2000). The agents are interacting by the concepts of belief, desire and intention and an example is given for diagnosing a computer network while dealing with the propagation of symptoms through components.

A multi-agent system that is tailored to specific requirements of monitoring and diagnosis is the European ESPRIT project DIAMOND (Albert et al., 2001). The agent communication is based on CORBA together with an Agent Communication Language as defined by FIPA (Foundation for Intelligent Physical Agents). This system does not focus on explicit diagnostic algorithms but on establishing a standardised multi agent architecture.

3. AGENT TECHNOLOGY

3.1 Definition of Software Agent

A software agent is a “proactive object”. It is a software entity that encapsulates data and code and it is running within its own thread of control. The decision how and when to perform an action is controlled by the agent itself. Beyond this, it is able to execute an action autonomously without being invoked externally. It diverges from a passive software entity like a software component which is waiting for a remote interaction.

In order to be a reasonable piece of software, an agent has to interact with its environment (Parunak 1998a), (Jennings, 1998). For this purpose, the agent builds its own (eventually fragmental) model of the environment by accessing sensor data and input from other agents. Since other agents may be part of the environment, the agent has also to be aware of their existence and have to know when and how to interact with them.

In the MAGIC system, the agents differ in their environment and in the way each agent models its specific environment. Different diagnostic agents may be related with the same part of the industrial application, but their view about this differs. One diagnostic agent may cover the electrical, another may cover the mechanical behaviour of the same physical component. When employing different diagnostic mechanisms for the same industrial process, the diagnostic agents may be related with the same environment, but each is building its own model of it.

3.1 Structure of Agent

The structure of the MAGIC agents is tailored to the needs of an object oriented design that allows to reuse common software parts in and to forward the implementation process. Some parts of an agent are related with agent specific communication functionalities (Communication Layer), others are tailored to an agent type (Generic AgentType Layer) and a third part is specific to an explicit industrial application (Specific AgentType Layer).

Communication Layer. This layer is identical for all agents, independent to their type and to a specific industrial application. Common to all MAGIC agents is their capability to communicate with other agents. In addition to this, the MAGIC agents have to be administrated and they have to be able to access a central database that is part of the presented multi agent infrastructure. All these functionalities are provided by a set of software classes packaged together within the communication layer.

Generic AgentType Layer. This layer is identical for all agents, having the same type (data acquisition, diagnostic, ...). Application specific requirements are not covered by this layer. For example, different diagnostic agents have the same Generic Diagnostic Layer, independent to the physical application they are related with. For each agent type, an own generic layer is required.

Specific AgentType Layer. This layer depends on the agent type and on the specific needs of the industrial application. It does not keep on track with the way to

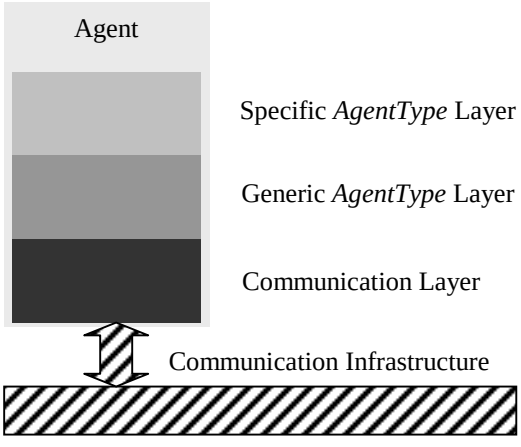

Fig. 1. Layered MAGIC Agent. An agent is composed of a communication layer, a generic and a specific agent-type layer.

to communicate knowledge with other agents, how to access data from the database or how to utilise CORBA. All functionalities of an agent that are specific to an industrial application and that can not be described generally are realised within this layer. For example the application specific diagnostic algorithms will be realised within this specific layer. Different diagnostic algorithms will be developed that are part of this specific diagnostic layer. These algorithms are intended to be mostly independent to the application, but a customisation and configuration will be necessary.

4. MULTI AGENT ARCHITECTURE

4.1 MAGIC's Agents

Different specialised agents come into operation within the MAGIC architecture. Some agents effect each others behaviour others don't. Some agents interactions follow fixed patterns (industrial process → data acquisition agent → diagnostic agent → diagnostic decision agent → operator interface agent) others are much more flexible. This results in a heterogeneous multi agent architecture with different agent types, each having different capabilities. All agents are autonomous entities and there is no external process control unit. This allows the system to coordinate itself pending on the dynamics of environmental changes and agent interactions. It allows a high level of autonomy and decision making to all available entities in order to improve the reactivity to disturbances which is extremely important for handling unforeseen faults in the industrial environment.

This concept is justified for distributed industrial applications with a high degree of complexity, missing hierarchical structures and a need for robustness against disturbances. In addition to this, industry requires a diagnostic system with high performance and a widely predictable behaviour which can be solely partly achieved by a conventional multi agent system. For this purpose, the MAGIC infrastructure provides also well defined agent interactions, according to predefined data flow. This approach is adopted to process oriented examples with a central controlling unit and several responder units and a well known flow of information.

The MAGIC multi agent architecture is similar to a holonic paradigm which merges the properties of hierarchical and heterarchical systems. In this manner, the holonic system can have a high and predictable performance together with robustness against disturbances (Bongaerts, 1998). In MAGIC, different agents may perform some controlling tasks like triggering a new diagnostic process. The decision when to control other agents is taken by the agent itself and is not fixed by the architecture as it is described for holonic systems. In this way, the co-operation and co-ordination of the multi agent system can be optimised according to specific needs for industrial diagnosis.

4.2 Agent Interaction

The different MAGIC agents are performing very specific tasks that are solely a small part of the overall diagnostic process. In order to reach the superior goal, the agents have to exchange information. These communications make very different demands on the flexibility (which information are exchanged in which way), the multiplicity (how many receivers of a message), the performance (realtime conditions) and the reliability of the communication (verifying whether a communication was successfully completed or not). According to the variety of demands, the MAGIC agents will be able to communicate by employing two different communication concepts:

Based on Agent Communication Language: One sender agent communicates with one receiver agent, based on the FIPA Agent Communication Language (Labrou, 1999) which enables a high flexibility by identifying completely different contents within the defined exchanged message, different communication protocols (Request and Query as defined by FIPA) and different potential answers from the receiving agent. Beside the flexibility, the use of an agent communication language allows permanent control of the communication processing state which makes it very reliable. On the other side, this communication comes along with some overhead which reduces its performance which is often a major criticism of multi agent systems. This communication type will be used whenever an agent wants to communicate with one other agent and there is a minor requirement to the processing speed than to the reliability and the flexibility.

Event Based Communication. A second communication mechanism is provided to MAGIC that employs an event based broadcast – sending only one packet and all other agents in the network recognise it. All data, an agent wants to provide are stored within a buffer that acts as a server. Following, the agent broadcasts an event to all other MAGIC agents. This event contains the information about the type of data that are available and a link to the sever that is able to provide the data. A remote agent that is interested in these information is able to access the data by using a CORBA Query. This type of data exchange is less flexible than the agent communication language but more straight forward. It will be used, whenever a pre-defined information flow within the diagnostic process is indicated.

4.3 Ontology

If an agent is part of another's agent environment (multi agent system), the agent has to model this environment and in this meaning, it has to build a specific model of the remote agent. For this purpose, both agents are able to communicate with each other and to exchange information. Beside a well defined set of languages and protocols used for interaction, they have to have a common meaning about the

information they are exchanging. They have to use at least a partly similar vocabulary and have to have a similar understanding about it. They have to share a common knowledge about exchangeable information.

This knowledge about knowledge is called *ontology*. Thus, an *ontology* can be defined as a formal and explicit representation of unambiguous definitions of objects, concepts and the relationships that hold them.

In MAGIC, the ontology has a particular important position since many agents are co-operating closely together and have to share their knowledge about the measurements and potential faults of the industrial application. MAGIC is deemed to be a framework that enables the customisation of a diagnostic system for very different industrial applications. Therefore, it is essential to identify generic patterns to describe these environments and to structure a diagnostic process. Besides providing generic diagnostic algorithms, a detailed specification of different ontologies is required. Focus is on:

- Plant Ontology
- Behaviour Ontology
- Administrative Ontology
- Diagnostic Ontology

5. MEETING INDUSTRIAL NEEDS

The multi-agent paradigm like any other software concept has certain capabilities which makes it best suitable for a specific type of application where these capabilities are required.

Industrial systems are most often affected by evolution of different kinds:

- Evolution of system requirements, functional and non-functional.
- Evolution of technology related to different domains.
- Evolution of technology used in software products.
- Evolution of technology used for the product development.
- Evolution of HSEQ (Health, Safety, Environment, and Quality) requirements.
- Business Changes.
- Organizational Changes.

All these changes have a direct or indirect impact on the life cycle of manufacturing systems. The ability to adapt to these changes becomes the crucial factor in achieving business success.

In the manufacturing domain, modern systems are modular, decentralised, changeable and highly complex. Parallel to the increasing complexity of manufacturing systems, the complexity of controlling, supervising and diagnostic systems is increasing in the same manner.

The benefits of using a multi agent approach are matching these industrial demands quite well (Parunak 1998a).

5.1 Modularity

The term modular refers to the design of any system, both hardware and software, composed of separate modules that can be connected together. In this way, the increasing complexity is managed by refining the overall system into subsystems. Well defined interfaces between the modules allow to replace or add any module without effecting the rest of the system. The physical application itself is modular and asks for a software system that matches with this attribute. In the same way as the physical module encapsulates most of the internal behaviour, the software agent hides most of its internal data. Only a small subset of the behaviour of the physical module is provided to other modules and a subset of the available data within the software module is provided to others. The modularity of a physical system becomes transferred to a modular software system.

A multi agent system is modular by default since each agent can be seen as a single module. In addition to this, a single agent is composed by a set of different modules (e.g. the different layers) as it is realised by using an object oriented approach. This allows to reuse some of the objects (modules, MAGIC agent layers) of one agent also within another agent, having different capabilities.

5.2 Decentralised

An agent is a pro-active object which does not need to be invoked from outside, but autonomously interacts with its environment and takes action by its own. This behaviour can also be found in industrial applications where a stand-alone process is controlled by its own without being triggered by other processes. The flow of control does not go from a central point down to the various processes as it is organised in centralised applications. The increase of complexity and cross linking between different processes pushes the utilisation of decentralised industrial applications. Agent based systems are fitting perfectly to such decentralised organisational systems. The autonomous nature of agents within a multi agent system allows to built up a highly decentralised architecture which models best a decentralised application.

The decentralised concept also appears if different diagnostic agents model the same process or industrial component by using different views (e.g. by applying different diagnostic algorithms). In this manner, both views are equivalent, there is no higher level view. This is like a decentralised concept, not in words of process flow, but in the specific views of different diagnostic models.

5.3 Changeable

Industry demands the ability to change a system quickly and cost effectively with the possibility to keep major parts on the software unchanged. To

guarantee this short time modifications, the software has to provide a changeable structure from the first. The modularity and the decentralised approach of a multi agent architecture as described in the chapters before are most suitable to allow fast adoptions to changes of the industrial application which leads to a competitive advantage of the software product.

The advantage of a multi agent system to allow short time changes of the industrial application is covered in MAGIC by the capability of the system to support the customisation and configuration according to specific needs when integrating the diagnostic system with an industrial application by providing additional tools (e.g. integrated within the Process Specification Agent).

5.4 Ill-structured

For an ill-structured application, it is not possible to provide all necessary structural information while designing the software system. The information about the application entities and their interactions are evolving themselves during execution. A software architecture, designed to treat such an ill-structured application has to be able to react on these changes in a proper way. Multi agent systems are excellent qualified to adopt these changing conditions since the sequence of interactions between the agents is not predefined.

The specific view of the MAGIC agents does not focus on the structure and interaction of the application entities itself, but on potential faults and their interactions. Even if the structure and the interfaces of the application are sufficiently predetermined, the faults may not. In which situation a specific fault occurs, how this fault relates with other faults or how this fault may change with time are examples of questions that may not be answered completely during the design phase of a diagnostic system. In this manner, the specific view of a diagnostic system about an industrial application is ill-structured by default.

5.5 Complex

The complexity of an industrial application reflects the number of different states and behaviours required to describe it sufficiently. This complexity has to be covered by a responsible software either by specifying explicitly how to react on each case and behaviour or by providing mechanisms how to evaluate the most proper way to react. Especially for very large and complex distributed applications, a traditional software becomes often unclear and unmanageable because of its complexity. A multi agent approach on the other side allows to cover the complexity by identifying clear interfaces and rules how to react on specific states of the environment of each agent. The complexity of the application is handled by different interaction scenarios automatically. The code can be kept more clear and less extensive.

5.6 Connection with industrial application

A diagnostic system requires information about the states of the industrial application. These data may be provided by an existing process control system that uses the sensor data to optimise a process. The control system is usually connected with the industrial application via local controller and a bus network. This equipment might also be used for connecting a diagnostic system with the industrial application. In doing so, real time constraints for the control system have to be considered in a way that the control response time must not be reduced. A diagnostic system with lower real time constraints should be clearly separated from the control system in order to avoid any inferences. Additional workstations and an additional bus system for diagnostic purpose are indicated. A suitable option for integrating a diagnostic system with an existing process control network is pictured in figure 2. The connection between the Diagnostic bus and the Field bus is realised by different Data Acquisition Agents, accessing sensor data and pre-processing them to be used for the diagnostic process. These information are used by the Diagnostic Agents that are in charge with different diagnostic techniques. The Central Diagnostic Agents co-ordinate and administrate the overall multi-agent system.

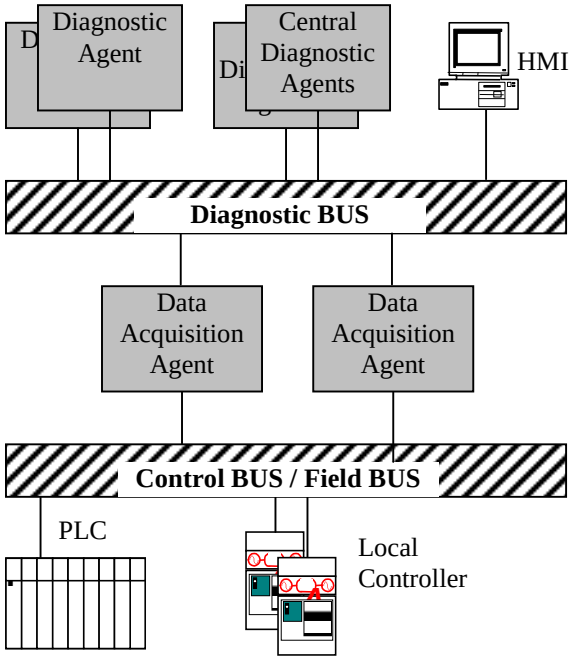

Fig. 2. Integration of MAGIC agents (data acquisition agent, several diagnostic agents and several central diagnostic agents) into an existing process control network.

5.7 Future trends in industry

Processing units will become smaller, more efficient and much cheaper in future. Process control and diagnostic capabilities could be decentralised in such a manner to become an integral part of a specific plant's unit. In other words, the physical unit might come with embedded diagnostic capabilities, realised by a diagnostic agent. The component manufacturer could equip a physical unit with a diagnostic agent, able to interact within a multi agent environment autonomously.

6. CONCLUSION

The overall objective of a monitoring and diagnostic system is to increase the reliability and the availability of complex applications. In order to build up such a system, a lot of generic tasks like data acquisition, knowledge management, communication between different units, performing diagnoses and displaying the diagnostic results have to be performed. The possibility to utilise different diagnostic engines in parallel is often a must for identifying faults. They may refer to different components of the industrial application or they may apply different diagnostic mechanisms.

The utilisation of a multi agent paradigm to provide these functionalities enables to develop a flexible and reliable monitoring and diagnostic system. The flexibility enables the diagnostic system to react efficiently to modifications in the state of the industrial application. Complex plants are generally speaking not being modified by changing the process or the workflow itself. The flexibility is required as soon as the plant diverges from the normal, fault-free behaviour into an unpredictable faulty state. In this case, a flexible diagnostic system is required to adopt automatically to the changes of the supervised application.

Together with a set of predefined ontologies and data processing mechanisms, different generic diagnostic algorithms and a set of reusable software agents, a standardised multi agent architecture for diagnostic purposes is formed. The presented multi agent paradigm which warrants the flexibility, the predictability behaviour, the extendibility and a cost effective development of a diagnostic system is most suitable to meet industrial requirements.

ACKNOWLEDGEMENT

This research work is being performed at the Institute for Process Control and Robotics, Prof. Dr.-Ing. H. Wörn, Department of Computer Science, University of Karlsruhe (Germany) in collaboration with the Gerhard-Mercator University of Duisburg (Germany), Institut National Polytechnique de Grenoble (France), S.A.T.E. s.r.l. (Italy) and SMS-DEMAG that is working in the steel processing industry. The research is partly funded by the European Commission in the MAGIC project under the contract No. IST-2000-30009.

REFERENCES

Albert, M., T. Längle, H. Wörn, A. Kazi, A. Brighenti, C. Senior, S. Revuelta Seijo, M. A. Sanz Bobi, J. Villar, 2001, Distributed Architecture for Monitoring and Diagnosis. EU Esprit Project No. 28735. http://wwwipr.ira.uka.de/~kamara/diamond/

Balakrishnan, Anantaram, R. T. S. Kumara, S. Sundaresan, 1999, Manufacturing in the Digital Age: Exploiting Information Technologies for Product Realization. In: *Published in Information Systems Frontiers, Kluwer Publishers, Inaugural Issue,1999*.

Bongaerts, L, L. Monostori, D. Mc Farlane, B. Kádár 1998, Hierarchy in distributed shop floor control. In: *Proc. of IMS-EUROPE 1998, the First International Workshop on Intelligent Manufacturing Systems*, pp.97-114.

Fabre, E., A. Benveniste, C. Jard, 2000, Distributed Diagnosis for large discrete events in dynamic systems, In: *Proceeding of the IFAC world congress*, (July 2002).

Milgrom, Elie, (2001), Methodology for Engineering Systems of Software Agents , CEDITI /, In: *project report of the EURESCOM project P907*.

Jennings, N.R., M.R. Wooldridge, (1998), Agent Technology Foundations, Applications and Markets, Springer-Verlag, ISBN: 3540635912

Yannis Labrou, T. Finin, Yun Peng, 1999, Agent Communication Languages: The Current Landscape. In: *IEEE March/April (1999)*

Letia, I.A., F. Cracium, Z. Kope, A. Netin, 2000, Distributed Diagnosis by BDI Agents, In: *IASTED International Conference: Applied informatics, (2000).*

Lind, J., (2001), Relating Agent Technology and Component Models, AgentLab, available at http://www.agentlab.de/.

MAGIC, B. Köppen-Seliger, T. Marcu, M. Capobianco, A. Brighenti, S. Gentil, M. Albert, T. Längle, S. Latzel, 2002, Multi-Agents-Based Diagnostic Data Acquisition and Management in Complex Systems, EU project IST-2000-30009, http://magic.uni-duisburg.de/

OMG, (2002) Object Management Group, Specifications data sheets of CORBA and its Services, In: http://www.omg.org.

Parunak, H. Van Dyke, (1998a), What can Agents do in Industry, and Why? An Overview of Industrially Oriented R&D, In *CEC, CIA98*.

Parunak, H. Van Dyke, (1998b), Practical and Industrial Applications of Agent-Based Systems. In: http://agents.umbc.edu/papers/apps98.pdf.

Poslad, S. J., R.A. Bourne, A.L.G. Hayzelden, P. Buckle, (1998), Agent Technology for Communications Infrastructure: An Introduction, In: *Agent Technology for Communications Infrastructure, A.L.G. Hayzelden, R. Bourne (Eds)*, John Wiley and Sons.