Development Tool for Distributed Monitoring and Diagnosis Systems

M. Albert, T. Längle, H. Wörn
Institute for Process Control and Robotics
University of Karlsruhe
D-76128 Karlsruhe
Germany

Abstract

This paper introduces a concept for building up distributed monitoring and diagnostic systems for complex industrial applications. The diagnostic process, from accessing sensor data up to the visualization within a graphical user interface is described by universal applicable formalisms. Generic mechanisms were identified to improve the quality of a diagnosis by integrating legacy diagnostic engines and handling different diagnostic mechanisms in parallel. For this purpose, a modular multi-agent architecture and a set of development tools were implemented. This software architecture for monitoring and diagnosis was developed within the framework of the EU Esprit Program: "**DIAMOND**: **DI**stributed **A**rchitecture for **MON**itoring and **D**iagnosis".

1 Introduction

To compete within industry, manufacturers are demanded to optimize their productive processes. In order to achieve this efficiency, a high value has to be set on the quality of the industrial equipment as well as on the industrial process itself. Monitoring and diagnostic systems (M&D systems) support this objective by predicting failures, or if a failure occurred, by identifying the reason for this fault. Thereby it is possible to reduce down-time costs of the production process. To achieve an overall reduction of the production costs, the development expense for a powerful M&D system has to be minimized. For this purpose, a modular concept was realized that is based on a distributed multi-agent approach.

A complex industrial application is built by a set of different physical units. These units may be provided by different vendors, having detailed knowledge about the behavior of their unit. Furthermore, there are often different diagnostic methods for the same unit available. To realize a powerful diagnosis, the knowledge about the different physical units and about various diagnostic methods should be merged together within an overall framework. Therefore, new strategies were required to treat these supplementing diagnostic knowledge about an industrial process.

This paper presents the generic aspects of the underlying infrastructure and describes the multi-agent framework. It is neither deemed to explain any diagnostic algorithms that may be applied nor to present the methodologies in detail that are employed to handle different diagnostic results in parallel.

2 Related work

Interest in recent research on distributed approaches for diagnostic purposes can currently be seen in Europe, Japan and in the United States. A general overview about distributed artificial intelligence in industry is given in [Par94]. This paper reviews the industrial needs for Distributed Artificial Intelligence, giving special attention to systems for manufacturing, scheduling and control. It gives case studies of several advanced research applications, actual industrial installations and identifies steps, need to be taken to deploy these technologies more broadly.

In [Fro96] there is a distinction between semantically distributed diagnosis and spatially distributed diagnosis. Semantically distributed diagnosis refers to a heterogeneous group of agents, in which each agent has its own view of the system. This can either mean that each agent focuses on a specific area of the system or that the agents model different aspects of the system or use different diagnostic methods, e.g. one agent models the structure of the system and another one models the performance. Spatially distributed diagnosis refers to a group of agents which jointly monitor and diagnose a spatially distributed system with relationships between those equipments. Each agent has detailed knowledge about a small part of the system.

Different concepts to realize a distributed approach are proposed in the literature, ranging from classical client server application over blackboard technologies to a few multi-agent frameworks. Most distributed applications employ a classical client server approach with distributed clients, communicating with a central server. This well known technology is continuously advanced and applied to distributed management applications. The standardi-

zation of distributed management tasks like information exchange, monitoring and diagnostics is aimed by the Common Diagnostic Model (CDM), developed within the "distributed management task force" (DMTF) [DMTF]. This framework is based on software clients which perform their tasks nearly automatically and report information to a central server.
Another concept is based on a blackboard technology. A central blackboard is used to store all available information about an industrial process. These data can be accessed by other software units, possibly software agents, in order to perform a diagnosis or to visualize the results. [Lee97] outlines the conceptual foundations for next generation industrial remote diagnostics and product monitoring systems. It extends the multi-agent framework research to include new classes of product population and diagnostic agents within a distributed Embedded Web and Electronic Commerce infrastructure. The products to be diagnosed are for example printers, copiers and vehicles.
[Ben97] introduces a KQML-CORBA (Knowledge Query and Manipulation Language) [KQML] based architecture to implement a multi-agent system for network and service management. The paper adopts this architecture and applies it to diagnosis and monitoring of machines and components in the production environment by using a multi-agent approach, combined with semantically distributed diagnosis. Up to now there are only few other implementations of KQML, one over TCP/IP in C which has been developed Lockheed-Martin [Fin99] and one in JAVA [Fro96]. The use of the agent communication language developed by FIPA [FIPA98] together with the CORBA standard for distributed computing is a widely used strategy to realize an agent interaction [Orf97].

3 Units of Monitoring and Diagnostic System

High value and cost effective diagnosis system can be developed by distinguishing between the tasks that have to be performed within a M&D system. Some tasks are general for all monitoring and diagnostic systems, some are specific for the employed production equipment and others are specific for the considered production process. These three units should to be treated in different ways:

3.1 Generic M&D units

Independent to specific requirements of an application area, a set of tasks are generic for all monitoring and diagnostic systems. Functionalities like interfacing different units, storing data, formatting measurements and diagnosis results, configuring the system, managing the interaction and some more have to be performed in all M&D systems. It is obviously feasible to reuse a set of existing software units whenever a specific Monitoring and Diagnostic system has to be built. This allows the integration of highly developed and well tested units extremely fast and low priced. Together with a set of well defined interfaces, a general architecture for monitoring and diagnosis becomes realized.

3.2 Specific to production equipment

Some parts of monitoring and diagnostic software are specific to the used production equipment. The way how to access sensor values and how to use these data to diagnose a physical component is specific to the production environment. The component manufacturer is interested to equip its component with monitoring and diagnostic capabilities that are compatible with a generic architecture. The usage of software agents allows to encapsulate legacy diagnostic tools in order to become interoperable to the overall system (see section 5). These parts are reusable whenever the same production equipment is used and should not be developed each time a M&D system is built from scratch.

3.3 Specific to production process

All remaining parts of the complete monitoring and diagnosis system are specific to the production process. The interaction between different physical components, and adoptions of the operator interface are examples for units that have to be develop individually.

4 DIAMOND – distributed multi-agent architecture

The architecture, proposed in this paper, tries to merge the three different parts to a consistent monitoring and diagnosis system. All generic units of a M&D system are realized by the utilization of software agents, each responsible for a specific task. An integration of all parts that are specific to the production environment or to the production process itself is supported by encapsulating these functionalities into different agents, able to interact with the overall framework. The DIAMOND architecture specifies the information that are required to integrate these parts and supports the development process by offering a set of tools (see section 5).
The structure and the interaction of the software agents that are able to perform all generic tasks and that are used to encapsulate all specific tasks are described in the following chapter.

4.1 Structure of DIAMOND Agent

All agents that are used within DIAMOND are built of two different software units. One is responsible for the communication with other DIAMOND agents within the M&D framework. This unit is called 'Wrapper' and is identical for all agents. The second software unit is called 'Brain'. This part performs all tasks that are specific for the type of the agent.
The Wrapper is responsible for handling the communication of this agent with other agents in the M&D system by applying the CORBA middleware. Information can be exchanged by applying CORBA events or by using

CORBA call-backs. The wrapper registers its CORBA objects at the ORB to become visible for other agents. However, the wrapper does not initiate or analyze any information exchange with another agent by itself. This is handled by the brain part.
The agents brain performs all tasks that are specific to the type of the agent. Some of the agents that are utilized within the framework perform exclusively generic tasks that are similar for all industrial environments. The implementation of the brain for these agents is uniform for each application where DIAMOND is applied. The tasks that have to be performed by the monitoring and diagnostic agents (see next chapter) are partly independent on the application. One part handles generic monitoring or diagnostic capabilities. This part of the monitoring agent brain and the diagnostic agent brain is provided by the DIAMOND development toolkit (see section 5.4 and section 5.5). All further capabilities depend on the specific application and have to be implemented afterwards individually by accessing well defined interfaces. This absence of an explicit implementation enables the integration of legacy monitoring and diagnostic tools inside the brain of a monitoring agent or inside a diagnostic agent.

4.2 Agent interaction

The Wrapper of each agent uses CORBA to exchange messages with other agents. It supports a synchronous call-back communication and an asynchronous event-based communication.
The wrapper exposes a CORBA remote interface to other agents that can use it, whenever they want to send a message to this agent by using a CORBA call-back. The cure, if an unknown protocol was received, if the agent is to busy to handle the message or if the agent will continue to process the message. Only in the last case, the message will be stored in an internal buffer that is part of the Wrapper. This buffer allows to sort incoming (and outgoing) messages according to their priority, their timestamp or in respect to a given time-out. The brain removes the message from the buffer as soon as it is able to process it. After processing the message, it specifies the answer, corresponding to the used protocol. This answer is stored in the internal buffer of the agent and forwarded to the remote CORBA object.
The message that is sent by an agent contains a set of pre-defined parameters as it is specified by FIPA. These parameters are stored within a XML structure. Since a conversation must not only handle information exchange but also the exchange of attitude about the information, a 2-layered protocol is applied. The outer layer of a message represents the attitude about the information. These data are processed by the Wrapper. The information itself is part of an inner layer which is stored in the content parameter of a message. The message content, which is also encoded in the XML syntax, is processed by the brain of the agent.
If an agent wants to supply data quickly to the overall multi-agent framework without taking care about the receiving agents, an asynchronous event-based communication is more feasible. This mechanism is mainly used by the monitoring agents to supply measurements to all diagnostic agents that are interested in. Every agent is able to supply and to consume events, structured in XML, by connecting to different event channels. This CORBA functionality is accessed by the Wrapper.

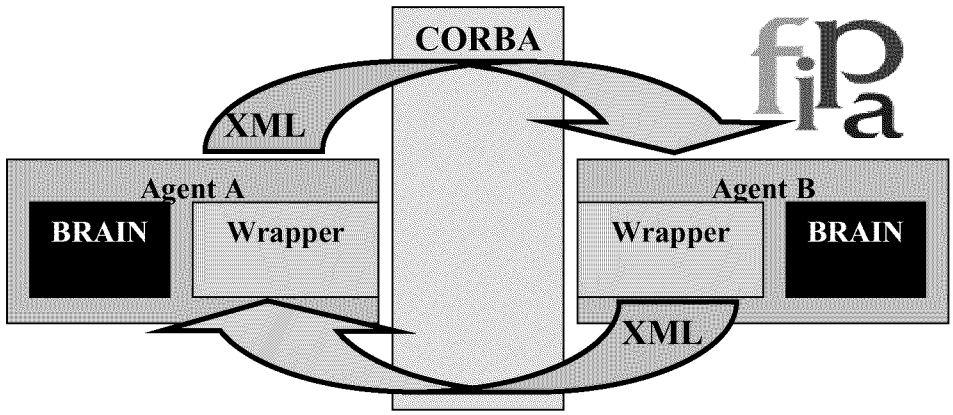

Figure 1 Agent Interaction

FIPA-Agent Communication Language (FIPA-ACL) [FIPA98] is used to restrict the interaction between communicating agents (see figure 1).The REQUEST, the QUERY-REF the SUBSCRIBE and the CANCEL protocol were identified to cover all required agent interactions. The call-back CORBA concept allows the receiving agent to return an integer value instantly if the structure of the message is invalid, if the message is not se-

4.3 Monitoring Agent

The interface between the physical state of the industrial application and the DIAMOND system is realized by a Monitoring Agent. This type of agent handles the measurements of the physical components and prepares them to be treatable by other agents within the framework. Each Monitoring Agent has to be adjusted to the sensors

of the industrial equipment that will provide the measurements. Furthermore, the Monitoring Agents are able to initiate a diagnosis of a component as soon as they have identified an irregular state of a measurement.

4.4 Diagnostic Agent

Different aspects of distribution are handled within the DIAMOND framework. First of all, the different tasks that have to be performed within the monitoring and diagnostic process are distributed to different agent types, each responsible for its specific task. The task that has to be performed by the Diagnostic Agents is also distributed again. The Diagnostic Agents are handling the measurements that are provided by the Monitoring Agents to identify the functional state of the physical components. This diagnosis may be performed by different Diagnostic Agents, each having a different view of the industrial application.

- This variation may be related with different temporal aspects of the behavior of the plant (temporal distribution).
- Often, there are different diagnostic algorithms available to identify the state of an industrial process. A development tool for a flexible M&D system has to be able to handle various diagnostic mechanisms in parallel. This is identified as a semantical distribution of the diagnosis.
- The entire diagnostic knowledge about the behavior of the plant is split to a set of smaller knowledge units, each associated with a physical part of the plant, called component. A single Diagnostic Agent does not know about the behavior of the complete plant, but about a single component. This knowledge may be provided by the manufacturer of the component. In this manner, the diagnostic task is spatially distributed.

When distributing the overall diagnostic task regarding temporal, semantical and spatial aspects, a flexible and clear framework is feasible. For diagnosing the overall process, the various diagnostic results, reported by different Diagnostic Agents have to be merged together. This additional task is performed by the Conflict Resolution Agent.

4.5 Conflict Resolution Agent

A conflict resolution mechanism is required to investigate, whether the diagnostic results, reported by different Diagnostic Agents are contradicting or completing each other. The Diagnostic Agents do not communicate with each other to merge their knowledge, but do report their diagnosis to a Conflict Resolution Agent. According to the different types of distribution, temporal, semantical and spatial conflicts have to be considered. For this purpose, the relations between the components and between the possible failures which may be related within the components have to be well known (section 5.1 and 5.3). The knowledge is represented by a Graph. An adjacent vector, where each element represents a component is used to build the graph [ALG94]. Each node (component) consists of:

- Vector of topological arcs with other components
- Vector of relationships between the same failure in different components
- Vector of relationships between different failures in different components.

The overall conflict resolution process is divided into different sequential steps:

- The reported failures are assigned to the nodes (components) of the graph conformed by the information specified in the structural knowledge base (chapter 5.1). This allows to identify semantical conflicts.
- Following, spatial and temporal conflicts are investigated in three different levels. Level 1 works with the topological information specified in the structural knowledge base, level 2 works with the relations between the same failure in different components (i.e. similar to level 1 but specific for a failure) and level 3 works with the relations between different failures in different components.

Details about these generic algorithms for handling temporal, semantical and spatial conflicts can be found in [DIAMOND].

4.6 Facilitator Agent

The Facilitator Agent is responsible for networking and mediating between the agents in the Multi-Agent framework. Large industrial applications may be federal and hierarchical structured. This structure is adopted to different "domains". A domain is a subsystem of the DIAMOND architecture that is responsible for a part of the industrial application. Each "domain" is associated with a facilitator agent to facilitate the networking within this domain and with other Facilitator agents of other domains. Thus a diagnosis of a single domain as well as a diagnosis of the complete industrial application is feasible. Furthermore, the Facilitator agent is the mediator to the Graphical User Interface Agent.

4.7 Blackboard Agent

All diagnosis results that were reported within a well defined timeframe are stored in a blackboard that is implemented in a Blackboard Agent. Each domain has its own Blackboard Agent that is mediating with the Conflict Resolution Agent. The Blackboard Agent provides the results, reported by the Diagnostic Agents and triggers the conflict resolution process. The resolved diagnostic result that cover the state of all components that are part of the domain are forwarded to the Facilitator Agent. The Blackboard Agent is also in charge of storing all reported diagnostic results permanently.

4.8 Graphical User Interface Agent

The Graphical User Interface Agent is the human gateway to the DIAMOND system. The operator uses this

interface to get information about the state of the industrial application, to provide human accessible information to the Diagnostic Agent and to initiate diagnostic processes.

4.9 Overall Architecture

The hierarchical and federal structure of the industrial environment that has to be monitored and diagnosed is transferred to a hierarchical and federal structure of the software architecture. For this purpose, industrial components are grouped together to form a logical superior unit. A set of agents that are responsible for diagnosing this set of components are grouped within a "domain". Only the Facilitator Agent of each domain is able to communicate with other Facilitator of other domains or with the Graphical User Interface Agent.
The main concepts of this DIAMOND architecture are summarized in figure 2.

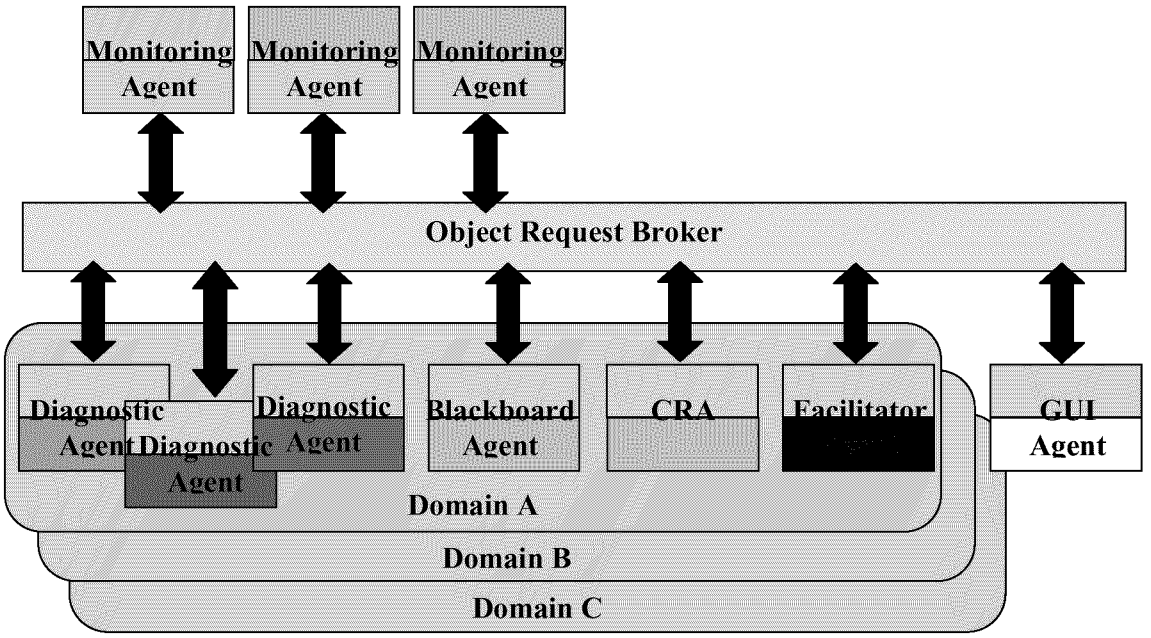

Figure 2 DIAMOND Architecture

All DIAMOND agents that are interacting are pictured as two colored boxes with the type of the agent written inside. The light green box indicates the Wrapper that is responsible for communication. This part is unique for all agents. The second box represents the Brain which is specific for the agents type. The agents are interacting by using the Object Request Broker (CORBA). This middleware is pictured as the yellow bar in the middle of the figure.
The figure indicates three different domains (Domain A, B and C). These domains are grouping the Diagnostic Agents, a Blackboard Agent, a Conflict Resolution Agent and a Facilitator Agent together. The Monitoring Agents are not associated to one single domain. They are able to provide measurements that may be used by the Diagnostic Agents of different domains.

5 How to build a monitoring and diagnosis system

This chapter describes the steps that are required to build up a complete monitoring and diagnosis system by using the results provided by the DIAMOND architecture.

5.1 Identify semantic structure of the industrial application

The first step while building up a monitoring and diagnosis system is to define all physical components and their relations of the automated industrial system that have to be supervised. There should be no overlap between components, nor should there be "white spaces" of the system being diagnosed not covered by a component at all.

The components of the industrial application may be hierarchical or federal related with each other. If there is a set of components that are building a logical unit which is widely self-contained, they have to be grouped together into a domain. The knowledge about the components and domains is fixed for a specific industrial application and will not change during runtime. DIAMOND provides an ontology that defines the structure and the possible attributes of any component. This knowledge is stored in the structural knowledge base in the XML format.
The possible relations between components are expressed by the attributes of each "COMPONENT" element:

- INPUT_CONNECTED_TO specifies functional or logical output of another component which effects the behavior of this component.
- EXCLUSIVE identifies that the faults of both related components are mutual to each other.
- BELONGING_TO is used to express the topological relation between "parent" and "child" components.

Further attributes are the certainty of the identified relation and a possible time delay which describes the temporal behavior of related components.

5.2 Identify measurements

It has to be investigated whether there are any existing monitoring sources available which are able to access measurable states of the plant. These sources may be accessed by a Monitoring Agent. It has to be investigated, which information about the measurable state of the industrial application is accessible and how to obtain them. In the case of integrating a legacy application for accessing the system variables, the interface to these applications have to be identified. All measurable states that are practical for a diagnosis have to be described as a measurement according to a well defined ontology for MEASUREMENT.

All measurements that will be used have to be associated with a Monitoring Agent that is able to access it. It is reasonable to associate all measurements that are provided by a single data acquisition source or by a specific mechanism how to access them with the same Monitoring Agent.

5.3 Identify failure modes

The M&D system is able to identify those potential faults of the industrial application that were specified in advance. Therefore, it has to be investigated, which legacy diagnostic tools may be used and which faults these modules are able to detect.

Attributes for each failure are specifying the conditions that have to be fulfilled, the names of rules that are feasible to identify the fault and potential recovery actions. Another attribute identifies whether the occurrence of this failure is related with another failure, either in the same or in another component. This allows to state whether different faults are contradicting or complementing each other, how they are temporally related and how a failure propagates.

All these information are stored within an XML structure. These data are mainly processed by the Conflict Resolution Agent to solve diagnostic conflicts. The diagnostic mechanisms and algorithms to identify the failure are specific to the industrial process and will be used by the diagnostic agents.

5.4 Build Monitoring Agents and connect with industrial environment

For building the Monitoring Agents, a shell is provided by DIAMOND that enables the creation of this agent. The connection with the monitoring and diagnostic infrastructure is automatically realized. The connection with the sensors of the industrial application has to be done afterwards. This task is specific for each application and for each sensor.

There are several predefined configuration parameters for a Monitoring Agent. These parameters may be set by using a DIAMOND toolkit.

5.5 Build Diagnostic Agents and connect with diagnostic engines

The measurements are used by the Diagnostic Agents to perform a diagnosis. For this purpose, each Diagnostic Agent needs to have a diagnostic engine. This may be a commercial expert system or any other kind of diagnostic engine to identify failures of the related component. The connection of the diagnostic engine with the M&D system is realized by using a development shell for creating the Diagnostic Agents. The interface to the diagnostic engine has to be implemented afterwards individually. There are only two methods that have to be implemented for interfacing:

One method enables the diagnostic engine to get a measurement value which is accessed from an internal buffer of the Diagnostic Agent. The engine does not keep care where to get the measurement from. All measurements that were identified in chapter 5.2 are accessible. After the engine has performed its diagnosis, it provides the diagnosis result to the Wrapper of the agent by using the second method of the interface. The Wrapper makes the result available for the infrastructure for further processing.

Integrating the diagnostic engine of a legacy diagnostic tool is possible, if this clear interface is realized. No further modifications, neither to the DIAMOND framework, nor to the diagnostic engine are required.

6 Evaluation Examples

The functionality of the presented multi-agent architecture was verified by integrating a specific monitoring and diagnosis system into two operational prototypes.

The first was concerned with the functional process of an automated welding cell, containing a control system, a robot with gas-metal arc welding equipment and a positioning device. To simulate faults that may occur in the welding system, a simulator was used that emulates the behavior of the welding equipment for different faulty situations. The measurements were accessed by using an ODBC interface and a DCOM interface. Several legacy case based reasoning engines, each responsible for another component, were applied to identify faulty components. This integration was suitable to present the capability to integrate different data accessing methods and various diagnostic engines within an integrative monitoring and diagnostic system easily. This M&D system was able to identify spatial conflicts and recognize the propagation of faults from one component to another one.

The second evaluation example took an existing expert system for diagnosing the water-steam cycle chemistry of a coal fired power plant (called SEQA, based on G2, Gensym) and re-worked it to operate in a modern diagnostic framework. To verify the behavior of the M&D system outside the power plant, a simulator based on a neural network model was used to either generate offline artificial anomalies overlapped to normal patterns or online to provide a set of normal behavior values against which measurements should be compared. The assimilation of a complete legacy expert system into a distributed M&D framework illustrated a complex tasks since there were many interfaces necessary for accessing data and for using the legacy diagnostic engines.

7 Conclusion

This paper describes a concept for building a distributed architecture for monitoring and diagnosing a complex industrial application. The presented M&D system uses a multi agent approach which warrants the flexibility, the extendibility and a cost effective development of the system.

One main extension to existing solutions is the possibility to integrate legacy diagnostic tools into the overall diagnosis system. This requires an extensive and exact specification of all components, measurements and possible failures of the industrial application as well as a specification of their relations to each other. This was realized by introducing a set of ontologies for the monitoring and diagnostic system.

Furthermore, several diagnostic engines can be utilized in parallel. They may refer to different components of the industrial application and they may apply different diagnostic mechanisms. By using different Diagnostic Agents, related to different components, the diagnostic knowledge can be provided by the component manufacturer. For applying different diagnostic methods, algorithms were developed to handle different diagnosis results in parallel and to investigate whether they are completing or contradicting each other.

Acknowledgment

This research work has been performed at the Institute for Process Control and Robotics, Prof. Dr.-Ing. H. Wörn and Prof. Dr.-Ing. R. Dillmann, Department of Computer Science, University of Karlsruhe (Germany) in collaboration with Union Fenosa Generación, S.A. (Spain), KUKA Roboter GmbH (Germany), GenRad Ltd (UK), Instituto de Investigación Tecnológica - Universidad Pontificia Comillas (Spain) and S.A.T.E. s.r.l.(Italy). The research was partly funded by the European Commission in the DIAMOND project under the Contract No. EP 28735.

References

[ALG94] "Introduction to Algorithms", T.H. Cormen, C.E. Leiserson, R.L. Rivest, the MIT Press, 1994.

[Ben97] Benech D., Desprats T., Raynaud Y., "A KQML-CORBA based Architecture for Intelligent Agents Communication in Co-operative Service and Network Management", IFIP/IEEE International Conference on Management of Multimedia Networks and Services, Montréal, Canada (1997)

[CORBA] http://www.corba.org/ 2002 „Common Object Request Broker Architecture", object management group.

[DIAMOND] http://wwwipr.ira.uka.de/~kamara/diamond/ 2001 "Distributed Architecture for Monitoring and Diagnosis". EU Esprit Project No. 28735.

[DMTF] "Distributed Management Task Force", http://www.dmtf.org/ , develops standards for distributed management tasks, 2002

[Fin99] Yannis Labrou, Tim Finin, Yun Peng, University of Maryland, Baltimore County: "Agent Communication Languages: The Current Landscape". IEEE March/April (1999)

[FIPA98] http://www.fipa.org/ 1998 "FIPA 98 Specification", Foundation for intelligent Physical Agents (1997-2001).

[Fro96] Froehlich, P., Nejdl W., "Resolving Conflicts in Distributed Diagnosis", ECAI '96, 12th European Conference on Artificial Intelligence, John Wiley & Sons, Ltd., (1996)

[KQML] T. Finin, R. Fritzson, D. McKay, R. McEntire, "A language and protocol for Information Exchange", Technical Report CS-94-02, Computer Science Department, University of Maryland, UMBC

[Lee97] Lee B. H., "Agent-based Remote Diagnostics of Product Populations Across the Full Product Life Cycle An Industrial Multi-Agent System Approach in an Electronic Commerce Framework", Proceedings of The Seventh International Workshop on Principles of Diagnosis, October 13-16, 1996, Val Morin, Quebec, Canada (1997)

[Nej94] Nejdl W., Werner M., "Distributed Intelligent Agents for Control, Diagnosis and Repair", Technical Report, Informatik V, RWTH Aachen, Germany, (1994)

[Orf97] Orfali, R., Harkey D., "Client/Server Programming with Java and CORBA", John Wiley & Sons, Ltd., (1997)

[Par94] Parunak V., "Applications of Distributed Artificial intelligence in Industry", O'Hare and Jennings, eds., Foundations of Distributed Artificial Intelligence, Wiley Inter-Science, 1996, (1994)

[XML] 2001, „Extensible Markup Language", http://www.xml.org/