DIAMOND - Distributed Multi-Agent Architecture for Monitoring and Diagnosis

H. Wörn, T. Längle, M. Albert
Institute for Process Control and Robotics,
University of Karlsruhe, D-76128 Karlsruhe,
Germany
Phone: 721 608 4263; Fax: 721 608 7141
(wörn, laengle, albert)@ira.uka.de

A. Kazi
KUKA Roboter GmbH
Blücherstrasse 144. D-86165 Augsburg,
Germany
ArifKazi@kuka-roboter.de

Attilio Brighenti
S.A.T.E. S.r.l
Systems & Advanced Technologies Engineering s.r.l
Santa Croce 664/a, 30135 Venice - IIALY
attilio.brighenti@sate-italy.com

Christopher Senior
GenRad Ltd.
Orion Business Park, Birdhall Lane
Stockport, Cheshire, SK 0XG UK
c.j.d.senior.2000@cranfield.ac.uk

Santiago Revuelta Seijo
Unión Fenosa Generación
Capitán Haya 53, 28020 Madrid – SPAIN
id-int-anllares@pop.uef.es

Miguel Angel Sanz Bobi, Jose Villar
Universidad Pontificia Comillas
Alberto Aguilera 23, 28015 Madrid - SPAIN
(Jose.Villar, masanz)@iit.upco.es

1. Abstract

This paper presents a concept for building up a distributed monitoring and diagnosis system for complex industrial applications. For this purpose, a hierarchical organised model with distributed, co-operating agents was developed. The hierarchical aspect guarantees a predictable behaviour of the system with a high performance and the flexibility of the system is ensured by the federal distribution [Bongaerts 98]. By using this approach, a modular component diagnosis and monitoring (CDM) system is realised that enables the integration of legacy monitoring and diagnostic tools, specific to the application area. Universal applicable mechanisms were found to perform diagnostic processes and to improve the quality of a diagnosis by handling different diagnostic mechanisms in parallel and by applying conflict resolution algorithms.

This software architecture for monitoring and diagnosis was developed by the University of Karlsruhe in co-operation with three industrial partners and one research institute within the framework of the EU Esprit Program: “DIAMOND: DIstributed **A**rchitecture for **MON**itoring and **D**iagnosis” [DIAMOND 02].

Keywords: Multi-Agent-System, distributed architecture, monitoring, diagnosis, conflict resolution, CORBA, FIPA-ACL, XML

2. Problems in Industrial Monitoring and Diagnosis

The need to compete within industry demands for more and more efficient productive processes. To achieve this efficiency, high value has to be set on the quality of the industrial equipment as well as on the industrial process. Monitoring and diagnosis systems support this target objective by predicting failures, or if a failure occurred, by helping to identify the reason for the fault. Thereby it is possible to reduce down-time of the production process [Edgar 00]. Efficiency of the overall production can be achieved by minimising the additional costs required for building a powerful monitoring and diagnosis system.

The increasing complexity of modern industrial plants determines a growing complexity of the monitoring and diagnosis system. To keep the CDM system “clear”, it is essential to encapsulate different tasks and to define strict interfaces between plant components and between components of the monitoring and diagnosis system. To guarantee flexibility to changing needs in case of an industrial application, the monitoring and diagnosis system must not be a “monolithic” software application, but rather has to be configurable and expandable without the need of modifying any line of code [Lüder 01].

The diagnostic knowledge about an industrial process is available on different parties (process specialists, component manufacturers, etc.). A modern monitoring and diagnostic system should be able to integrate the diagnostic knowledge from all available sources, even if different diagnostic mechanisms are applied. To achieve an overall diagnosis of an industrial process, several diagnostic tasks have to be performed in parallel. This requires new strategies to handle diagnostic conflicts that might occur between different diagnostic results.

A general overview about distributed artificial intelligence in industry is given in [Parunak 94]. This paper reviews the industrial needs for Distributed Artificial Intelligence, giving special attention to the needs arising from systems for manufacturing, scheduling and control.

3. Related Work

Different distributed approaches for monitoring and diagnosis are proposed in the literature, ranging from classical client server application over blackboard technologies to multi-agent frameworks. Most distributed applications in the manufacturing domain employ a classical client server architecture, with distributed clients communicating with a central server. The standardisation of distributed

management tasks like information exchange, monitoring and diagnostics is aimed by the Common Diagnostic Model, developed within the “distributed management task force” (DMTF) [DMTF 02]. This framework is based on software clients performing their tasks in a nearly automatic manner and reporting the resulting information to a central server.

Other concepts are based on the 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 visualise the results.

Several descriptions can also be found in literature about flexible and powerful structures of multi-agent systems. [Lee 97] 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. Nej 94] discusses the use of distributed intelligent agents to build a system that incorporates control, diagnosis and repair tasks. A hierarchy within a distributed system of co-operating agents was identified by [Bongaerts 00] to meet requirements originated by industrial processes. A specific evolution of this concept is described by “holonic” systems, where software units are able to reach a common goal by exchanging knowledge and by performing “negotiation”.

As an example of a multi-agent system, [Ben 97] introduces a KQML-CORBA (Knowledge Query and Manipulation Language) [Finin 94] based architecture 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, combined with “semantically distributed diagnosis”. Actually, several further activities are trying to bring the agent technology into industrial process control. The PABADIS model (Plant Automation based on Distributed Systems, funded by the European Union under the IMS and IST programmes) [IMS 02], [Lüder 01] distributes the resource planning and production flow control in single-piece production plants by means of multi-agent systems. This model is composed by “stationary” software agents, each representing a production resource (tooling machines, database servers, transport systems) and by “mobile” agents, representing the work pieces of the industrial application, carrying product and production information.

4. DIAMOND - Multi-Agent Architecture

The distributed M&D (Monitoring and Diagnosis) system referred to as DIAMOND, with hierarchical and federal organised software agents that are responsible for different tasks within a monitoring and

diagnostic process is presented in the following, covering different aspects of distribution, the structure of the agents and their interactions.

4.1. Distribution of the Diagnostic Knowledge

The knowledge about the structure and the behaviour of the industrial application to be monitored and diagnosed is distributed. There are different sources of knowledge about the physical units of the plant, which have to be merged together to perform a diagnosis of the overall system. In [Fro 96] there is a distinction between semantically distributed diagnosis and spatially distributed diagnosis. This approach is supplemented by a temporal distribution to cover different temporal aspects of the same diagnostics.

- **Semantically distributed diagnosis**: Different views of the system are available, each suitable to identify a fault of the plant. This can either mean that each diagnosis focuses on a specific aspect of the system or uses different diagnostic methods, e.g. one diagnosis models the structure of the system and another one models the performance. Integration of several diagnostic algorithms, even running in parallel, must not be limited by the structure of the monitoring and diagnosis system. This is the only way to enable an integration of legacy diagnostic engines, i.e. commercially available expert system development tools or other artificial intelligence software.
- **Spatially distributed diagnosis**: Each reasoning process focuses on a small physical component of a larger system. These spatially distributed pieces of software are related with other components of the industrial equipment. The knowledge that is handled within different diagnostic processes may be supplied by different sources (component manufacturer, line builder, process experts, etc.)
- **Temporal distributed diagnosis**: During the diagnostic process of a varying system, different temporal aspects of several diagnostic results have to be merged together. The temporal propagation of a fault from one component to another one has to be considered as well as the occurrence of different diagnosis, covering diverse time frames.

Flexibility of the M&D system and the economical perspective to reuse existing monitoring and diagnostic software components, require the adoption of a modular software architecture. Handling distributed diagnostic knowledge in a semantical, spatial, and temporal manner makes conflicting diagnoses inevitable. Solving these conflicts may either be achieved by mutual information exchange and negotiation, as it is described for holonic systems [Bongaerts 00], or by introducing an additional,

hierarchically superior software unit with the capability to resolve these conflicts (see section 5.3). The latter approach is more suitable for a flexible M&D system, since a single diagnostic engine is able to handle only a small part of the overall diagnostic knowledge. It is not in a position to deal with the diagnostic results, provided by other diagnostic units, applying different diagnostic methods or concerning different physical components of the plant.

4.2. A Hierarchically organised M&D System

Production plants can be highly complex as well as their components (e.g. industrial robots, computer controlled numerical machines, etc.). It is desirable to use the same M&D architecture both on a large scale (i.e. on the plant level) and on a small scale (i.e. on the plant component level). A complete multi-agent based diagnostic system for an industrial robot, for example, appears as a single part of a superior M&D system on the plant level. Diagnostic components supplied by the manufacturers of robot components are used for the diagnosis of the robot, while in turn the diagnosis system of the robot participates on the whole in the diagnosis of the production plant. In this way, the monitoring and diagnosis system becomes hierarchically organised. The overall M&D system may consist of several inner M&D systems, which may cover again several smaller M&D systems, as a nested set. Each of the inner, self-consistent M&D systems is called “domain”. The hierarchical structure of the plant is adapted to a hierarchical structure of the monitoring and diagnosis system and the logical physical entities (production cell, production line, machine, etc.) become monitored and diagnosed by a domain.

The diagnosis for a specific physical entity is completely performed by the agents that are responsible for this domain. Each domain is hierarchically organised by the “facilitator agent”, the “blackboard agent” and the “conflict resolution agent”, whereas the monitoring and diagnostic tasks are distributed to several “monitoring and diagnostic agents” (see section 5). This hierarchical organisation of distributed systems is similar to an holonic approach. Unlike holonic systems, the agents in the DIAMOND architecture do not co-operate while making decisions after mutual information exchange, negotiation and agreements. This approach prevents the occurrence of an unpredictable behaviour, guarantees the overall performance, ensures the convergence of the agent communication, and provides a clear control of the system. All such requirements are extremely important in case of industrial applications [Kraus 95].

4.3. Structure of the DIAMOND Agents

All agents used within DIAMOND are made of two different software units. One part is responsible for the communication with other DIAMOND agents. This unit, called “Wrapper”, is the same for all agents. The second software unit, called “Brain”, performs all tasks that are specific for the type of the agent.

The Wrapper is responsible for handling the agent communication with other ones in the M&D system by accessing CORBA functionalities. However, it does not initiate or analyse any information exchange with another agent by itself. This is handled within the Brain part of each agent.

The Brain is responsible for performing all tasks that are specific for the type of the agent. Some perform exclusively generic tasks that are the same for all industrial environments that are monitored and diagnosed. The implementation of the Brain for these agents is uniform for each application. The tasks that have to be performed by the monitoring and diagnostic agents (see section 5) are partly independent on the application. One part handles generic monitoring or diagnostic capabilities. This part of the Monitoring and the Diagnostic Agent Brain is provided within the DIAMOND development toolkit. All further capabilities depend on the specific application and have to be implemented afterwards individually. The absence of an explicit implementation is a requirement to encapsulate legacy monitoring and diagnostic tools inside the Brain of a Monitoring Agent or inside a Diagnostic Agent.

4.4. Agent Interaction

The Wrapper of each agent 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 [Orf 97]. The FIPA-

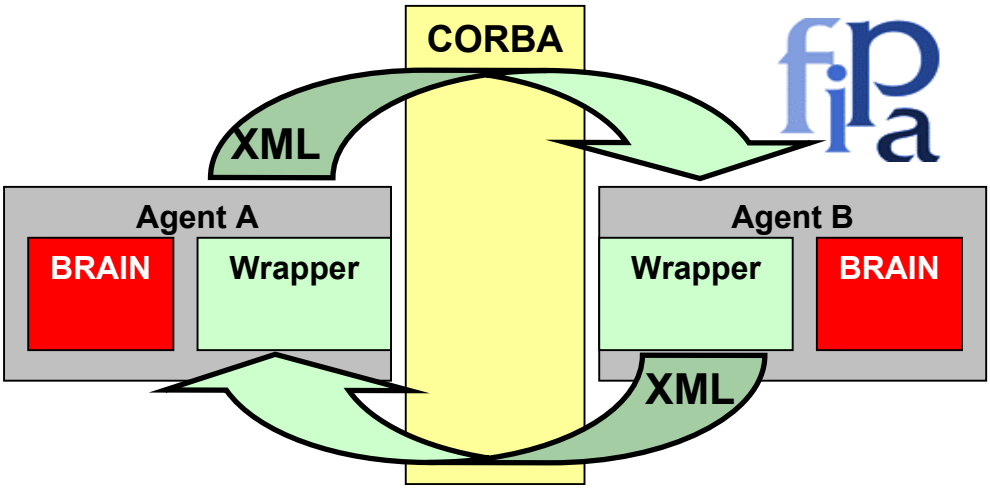

Figure 1 Agent Interaction

Agent Communication Language (FIPA-ACL) [FIPA 98] [Labrou 99] 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 corresponding to the following cases: if the structure of the message is invalid, if the message is not secure, if an unknown protocol was received, if the agent is too busy to handle the message or if the agent will be able 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 parameter. The Brain removes the message from the buffer as soon as it is able to handle it. After processing the message, it specifies the answer, corresponding to the result and the used protocol. This answer is stored in the internal buffer of the agents Wrapper and forwarded to the remote CORBA object.

The message that is sent contains a set of pre-defined parameters as it is specified by FIPA. These parameters are stored within a XML structure [Kesselring 98]. 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, 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 of 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. All agents are able to supply or to receive and process events, structured in XML, by connecting to different CORBA event channels. This CORBA functionality is accessed by the Wrapper.

5. Agent Species

The monitoring and diagnostic process can be distributed to a set of co-operating tasks. Each task is associated with a specialised agent. The clear structure and the distribution of the computational load over various nodes is beneficial especially in large plants with complex diagnostic reasoning mechanisms.

5.1. The Monitoring Agent

The interface between real-world objects of the industrial application and the diagnosis system is realised by Monitoring Agents. In many cases, input comes from simple sources such as a database, but it can also come from other sources such as electronic sensors. This concept is in many ways similar to the “sensor analysis agents” proposed by [Lee 97]. The Monitoring Agent accesses the data, wraps them to the other agents in the architecture and provides access to it through a standardised “interface”. This access would either be performed using an agent communication language or by supplying the data via a CORBA event mechanism. Beyond this, the Monitoring Agent may filter the sensor data or can perform calculations, useful for a diagnostic process.

The concept of a Monitoring Agent may prove even more important for future applications. Indeed the ongoing developments in the so-called “smart sensors” technology aim at providing processing capabilities on the same microchip of a conventional sensor. This allows sensors with additional embedded software functionalities to be built from a single device. There is no reason why this software should not be a monitoring agent, providing an extremely neat solution for capturing monitoring data [Senior 99].

5.2. The Diagnostic Agent

The Diagnostic Agents are handling the measurements provided by the Monitoring Agents to identify the functional state of the plant. This state contains an unique identifier, a set of conditions under with the state was identified, rules and “meta-rules” used to identify the state, a conclusion, a timestamp and a certainty factor to identify the reliability of the identified functional state. This diagnosis may be performed by different Diagnostic Agents, each one having a different view of the industrial application, since the diagnostic knowledge is distributed (see section 4.1). By distributing the overall diagnostic tasks regarding temporal, semantical and spatial aspects, a flexible and clear framework is feasible. But for diagnosing the overall process, the various diagnostic results, reported by different Diagnostic Agents have to be merged together. In DIAMOND, the task of conflict resolution is not performed by mutually exchanging information and negotiation, but by a specific agent, called Conflict Resolution Agent.

5.3. The Conflict Resolution Agent

A conflict resolution mechanism is required to investigate whether the diagnostic results, as 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 (sections 6.1 and 6.3). The knowledge is typically represented by a Graph. An adjacent vector, where each element represents a component is used to build the graph [Cormen 94]. Each node (component) consists of:

- Vector of topological arcs with other components to describe the relationships among different 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 (section 6.1). This allows to identify first of all semantical conflicts.
- Following the identification of semantical conflicts, spatial and temporal conflicts are investigated at 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 02], [Sanz 98], [Sanz 99], [Gertler 91].

5.4. The Facilitator Agent

The Facilitator Agent is responsible for networking and mediating between different agents. Therewith, it represents the hierarchical aspect of DIAMOND. Each Domain (section 4.2) is associated with a Facilitator agent to facilitate the networking between the agents 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, while by these means the system flexibility is managed by the appropriate design and implementation of the Facilitator Agent.

All knowledge about the semantic structure of a Domain, the capabilities of the registered Monitoring and Diagnostic Agents and the available CORBA functionalities are managed by the various Facilitator Agents. Furthermore, the Facilitator Agent is the mediator to the Graphical User Interface Agent.

5.5. The Blackboard Agent

The diagnoses, 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 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 covers 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 diagnoses permanently. Conflict Resolution Agent, Facilitator Agent and Blackboard Agent are working closely together and are responsible for a hierarchical organisation of the agent interaction within a Domain.

5.6. The 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. This agent is on top of the multi-agent hierarchy.

5.7. The Overall DIAMOND Architecture

In a DIAMOND application implementation the hierarchical and federal structure of the industrial environment that has to be monitored and diagnosed is transferred to a corresponding hierarchical and federal structure of the software architecture. For this purpose, industrial components are grouped together in the way most suitable for the diagnostic process. The set of agents that are responsible for diagnosing a corresponding set of components are grouped within a Domain. The main concepts of this DIAMOND architecture are summarised in the following figure.

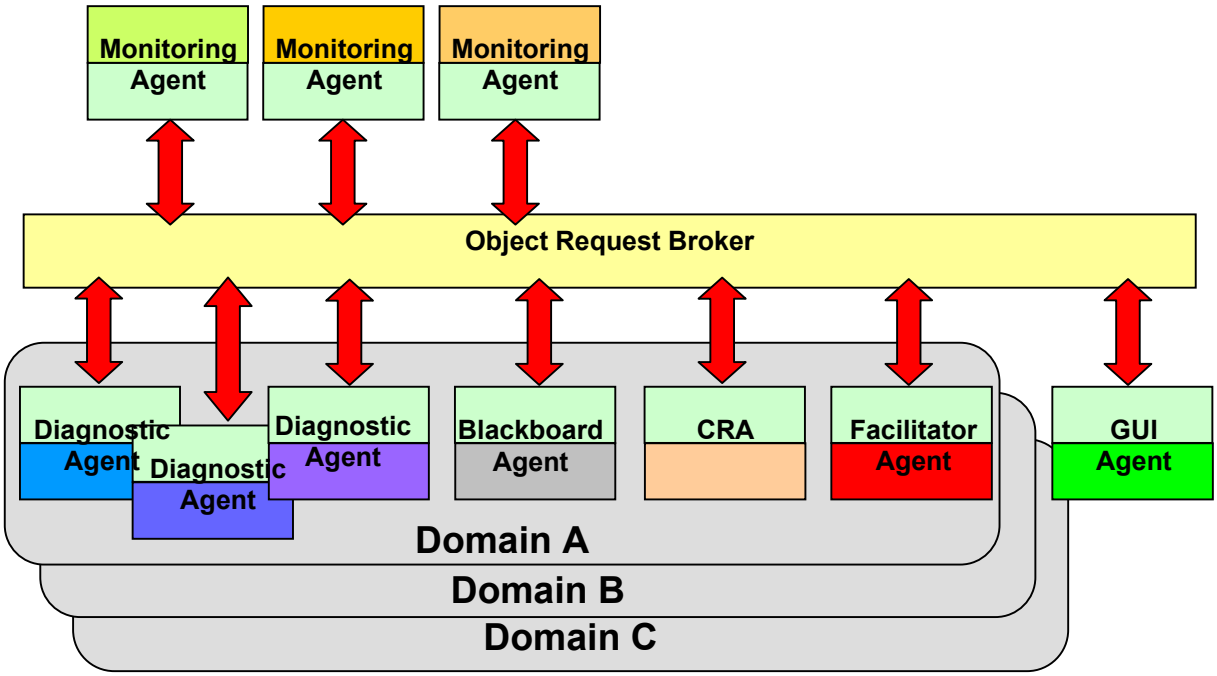

Figure 2 DIAMOND Architecture

All DIAMOND agents are pictured as boxes with the name of the agent inside. One part of each agent box indicates the Wrapper of each agent that is unique for all agents. The second box represents the Brain of an agent which is specific for the agents type. The agents are interacting by using the Object Request Broker (CORBA). This middleware is pictured as a big bar in the middle of the figure. It enables the communication between all agents that are connected to the system.

The figure shows an example concept with three different Domains (Domain A, B and C). These Domains are grouping the diagnostic agents, one blackboard agent, one conflict resolution agent and one facilitator agent together. The monitoring agents are instead not associated to a specific domain. They are able to provide the values of the measurements that are related to components, which may belong to different domains.

6. Development Process for specific M&D System

6.1. Identify Semantic Structure of the Industrial Application

The first step while building up a monitoring and diagnosis system is to define all components of the automated industrial system that has to be supervised. There should be no overlap between components, nor should there be “white spaces” of the system being diagnosed, which is left uncovered. Besides defining the components, their relations among each other must be known in advance.

The components of the industrial application may be hierarchically or federally related with each other. The knowledge about the components and the domains is fixed for a specific industrial application and will not change during runtime. DIAMOND provides an ontology [Schlenoff 99], [Williams 00] that defines the structure and the possible attributes of any component. This information is stored as a XML document. 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 affects the behaviour of this component.
- EXCLUSIVE identifies that the occurrence of different faults, associated with different components, are mutually exclusive.
- 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 behaviour of connected components.

6.2. Identify Measurements

It has to be investigated which information about the measurable state of the industrial application is accessible and how to obtain them (sensors, databases etc.). In case of integrating legacy applications for accessing the system variables, the interface to these applications has to be identified. All measurable states that are practical for a diagnosis must be described as a measurement according to a well defined MEASUREMENT ontology with attributes like name, scale, sensor type, minimum value, maximum value, etc..

6.3. Identify Failure Modes

A monitoring and diagnostic system is able to identify those functional states that were specified in advance. These potential functional states must be known before the diagnosis system is implemented. For this purpose, it has to be investigated which legacy diagnostic tools may be used and which functional states of the plant can be detected by the legacy diagnostic tools. Before integrating the legacy diagnostic tools into a Diagnostic Agent, the functional states must be formatted in accordance to the DIAMOND FAILURE-MODE ontology. The failure modes of the plant are uniquely associated to a component and this component is part of a specific domain.

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 may propagate.

This information is stored within an XML structure. The 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.

6.4. Build Monitoring Agents and connect with Industrial Environment

DIAMOND provides a shell that enables the creation of a Monitoring Agent. The connection of this agent with the multi-agent infrastructure is automatically realised. The connection with the sensors of the industrial application must 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 are set using the “DIAMOND Toolkit”.

6.5. 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, able to identify failures of a plants component. The connection of the diagnostic engine with the M&D system is realised by using the development shell. The interface to the diagnostic engine has to be implemented afterwards individually. Only two methods must 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 has not to take care where to get the measurement.

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 to the infrastructure for further processing.

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

6.6. Debug M&D system

After a monitoring and diagnosis system is built, it has to be tested whether it is working correctly. For this purpose, the interaction between the agents must be debugged and various faults of the industrial application must be simulated to investigate the behaviour of the M&D system.

For testing the agent interaction, a virtual model of a multi-agent network is provided by DIAMOND that allows finding out proper configurations of the communication parameters of all agents that will be used in the final M&D system. This model simulates the agent interaction, depending on their configuration. It allows preventing excessive time losses, odd or faulty diagnoses, and represents a tool for optimising the agents network. This model assists the system integrator to assess the influence of the design choices on the performance of the M&D system. The main scope of this virtual model is not to describe the physics or the lower level software underlying the communication itself, but rather to model the interactions among the various agents.

Beside the virtual model of the multi-agent network, it is necessary to verify and validate the implemented M&D system under real conditions. It has to be investigated how the M&D system reacts in the case of an occurring fault of the industrial process. However, in many application areas it is impossible to generate real faults. In these cases, the utilisation of a model that allows simulating the behaviour of a real environment is inevitable.

7. Evaluation examples

The functionality of the presented multi-agent architecture was verified by integrating the monitoring and diagnosis system into two completely different operational prototypes. The first investigates the diagnostic process in an automated welding cell, integrating different diagnostic reasoning engines to form an integrative diagnosis system. The second takes an existing expert system for diagnosis in a coal fired power plant and re-works it to operate in a modern diagnostic framework.

7.1. Automated Welding Cell

The automated welding cell contains a control system, a robot with gas-metal arc welding equipment and a positioning device. The welding equipment consists of an own control system, a wire feed control and a power supply. This system is used to demonstrate the functionality and the capabilities of the distributed multi agent architecture for monitoring and diagnosis.

The following monitoring and diagnostic agents were implemented:

- **System variable and I/O monitoring agent**: reads the input and output data of the robot controller. It is able to provide information about the current state of internal robot controller variables like operating mode or current position of the robot.
- **Logbook monitoring agent**: provides logged historical data of the robot controller that were stored within an internal database. Error messages, user accesses and internal warnings can be accessed by using an ODBC (Open Database Connectivity) interface.
- **Case based diagnostic agent for the robot**: uses a commercial case based diagnostic engine containing faults concerning the robot controller (i.e. hardware defects. loose or broken cables, etc.).
- **Case based diagnostic agent for the welding equipment**: uses a commercial case based diagnostic engine containing faults concerning the welding equipment. The required knowledge could be provided by a welding equipment manufacturer.
- **Case based diagnostic agent for the welding process**: uses a commercial case based diagnostic engine and contains different cases, assorted into 5 different groups, depending on the current phase of the welding process (pre-ignition, ignition, welding, welding termination or no active welding phase).

To simulate faults that may occur in the welding system, a simulator was developed that emulates the behaviour of the welding equipment for different faulty situations.

7.2. Water-Steam Cycle Chemistry of a Coal Fired Power Plant

The distributed architecture presented in this paper was also evaluated while monitoring and diagnosing the water-steam cycle chemistry of a coal fired power plant. Within this industrial environment, the integration of legacy diagnostic tools is important since the development of new tools is extremely time and money consuming. Therefore, an existing monitoring and diagnosis system called SEQA was integrated into the DIAMOND architecture. This monitoring and diagnosis system

supervises permanently the measurements of the water-steam cycle chemistry inside a coal fired power plant. SEQA performs an on-line monitoring of the most representative chemical variables in order to detect abnormal values in respect to the normal chemical behaviour of the plant.

The following monitoring and diagnostic agents were used to build up the overall DIAMOND system:

- **Monitoring agent for automatic measures**: is able to collect measurements provided by sensors and takes measurements provided by a simulation system.
- **Anomaly detection agent**: is a diagnostic agent being in charge of testing incoming measurements, analysing if they correspond to normal values and the current working condition. It applies threshold testing for each measurement, quality control techniques and black-box modelling based on neural networks.
- **Expert system diagnostic agent** receives the anomalies discovered by the anomaly detection agent and tries to detect the root cause of a detected anomaly.

To verify the behaviour of the M&D system outside the power plant, a simulator based on a neural network model is used to either generate offline artificial anomalies overlapped to normal patterns of functioning or, directly online, to provide a set of normal behaviour values against which measurements should be compared, to provide symptoms to be handled and analysed by the diagnostic system.

8. Conclusion

This paper describes a concept for building a monitoring and diagnosing system of a complex industrial process based on the application of a distributed multi-agent architecture. The multi-agent approach allows the flexibility, the extendibility, and a cost effective development of the system. Building up a M&D system by using the DIAMOND results, the design effort vanishes completely and the communication infrastructure can be used directly. A reduction of the development effort for several person months is feasible. This advantage could even be increased if component manufacturer would equip their physical units with DIAMOND compatible agents.

One main extension to existing solutions is the possibility to integrate legacy diagnostic tools into an overall diagnosis system. The DIAMOND demonstrator for the water steam cycle of a coal fired power plant illustrated this feature clearly. Major parts of an existing diagnostic system were integrated within a superior DIAMOND architecture. Additional features were added and an adaptation to user requirements were made.

An extensive and detailed specification of all components, measurements and potential failures of the industrial application as well as a specification of their relations has to be carried out when applying the DIAMOND results to a new application area. The structure of all information that are relevant for diagnostic purposes were identified and corresponding ontologies were introduced. This formal description of diagnostic relevant knowledge supports transferring the DIAMOND results to new application domains.

Another remarkable outcome of DIAMOND is its capability to utilise several diagnostic engines in parallel. They may refer to different components of the industrial application and they may apply different diagnostic mechanisms. Both concepts were implemented in the presented evaluation examples. By using different Diagnostic Agents, related to different components, the diagnostic knowledge could be provided by different sources (e.g. component manufacturer). To apply different diagnostic methods, generic algorithms were developed to handle different diagnosis results in parallel and to investigate whether they are complementing or contradicting each other. All relations between different components and failure modes that were identified in advance were recognised and resolved by the conflict resolution agent according to the developed procedures.

The DIAMOND technology proved to be feasible and conveniently applied in the demonstration areas selected for the project. Since these are very different from each other and the solutions provided by DIAMOND are based on a general approach, the extension also to other fields is feasible and capable to bring thereto the DIAMOND approach benefits.

9. Acknowledgement

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 is partly funded by the European Commission in the DIAMOND project under the Contract No. EP 28735.

10. References

[Ben 97] 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

[Bongaerts 00] Bongaerts L., Monostori L., Mac Farlane D., Kadar B. “Hierarchy in distributed shop floor control”, *Computers in Industry*. 43: 123-137, 2000

[Bongaerts 98] Bongaerts L, "Integration of Scheduling and Control in Holonic Manufacturing Systems", Ph.D. Thesis PMA / K.U.Leuven, Chapter 3, 1998

[Cormen 94] T.H. Cormen, C.E. Leiserson, R.L. Rivest, “Introduction to Algorithms”, the MIT Press, 1994

[DIAMOND 02] M. Albert, et al., “DIAMOND reference model”, EU Esprit Project No. 28735: DIAMOND: “Distributed Architecture for Monitoring and Diagnosis”. (Further information can be found in: http://wwwipr.ira.uka.de/~kamara/diamond/), 2002

[DMTF 02] “Distributed Management Task Force”, http://www.dmtf.org/, develops industrial standards for distributed management tasks, 2002

[Finin 94] Tim Finin, et. al., “Specification of the KQML Agent-Communication Language”, 1994

[FIPA 98] Foundation for Intelligent Physical Agents (FIPA) 97 Specification Version 2.0, Part 2, “Agent Communication Language”, (Available at *http://www.fipa.org/*), 1998

[Fro 96] Froehlich, P., Nejdl W., “Resolving Conflicts in Distributed Diagnosis”, *ECAI ’96, 12th European Conference on Artificial Intelligence, John Wiley & Sons, Ltd.*, 1996

[Gertler 91] Gertler, J., “Analytical redundancy methods in fault detection and isolation. Survey and synthesis”, IFAC Fault Detection Supervision and Safety for electrical processes, Baden-Baden, Germany, 1991

[IMS 02] “Intelligent Manufacturing Systems, an industry-led RTD international co-operation initiative, (Available at: http://www.ims.org/), 2002

[Kraus 95] Kraus S., Wilkenfeld J, Zlotkin G., “Multi-agent negotiation under time constraints”, *Artificial Intelligence*. 75: 297-345, 1995

[Labrou 99] Yannis Labrou, Tim Finin, Yun Peng, "Agent Communication Languages: The Current Landscape". *IEEE March/April* , 1999

[Lee 97] 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

[Nej 94] Nejdl W., Werner M., "Distributed Intelligent Agents for Control, Diagnosis and Repair*", Technical Report, Informatik V, RWTH Aachen, Germany,* 1994

[Orf 97] Orfali, R., Harkey D., "Client / Server Programming with Java and CORBA", *John Wiley & Sons, Ltd.,* 1997

[Lüder 01] Lüder A., et al., "Industrial Requirement and Overall Specification", prepared within the PABADIS IST research project no. "IST-1999-60016", (further information can be found in www.pabadis.org), 2001

[Parunak 94] Parunak V., "Applications of Distributed Artificial intelligence in Industry", *O'Hare and Jennings, eds., Foundations of Distributed Artificial Intelligence, Wiley Inter-Science, 1996*, (1994)

[Sanz 98] Sanz-Bobi, Munoz A., "An incipient fault detection system based on the probabilistic radial basis function network: application to the diagnosis of the condenser of a local power plant." *Neurocomputing*, **Vol. 23**, 177-194, 1998

[Sanz 99] Sanz-Bobi, M.A., Sanchez-Ubeda, E.F., Villar J. Revuelta S. Kazi A., "Multi-agent diagnosis systems in industry", 1st International IFAC Workshop on Multi-Agent Systems in Production (MAS'99), Vienna - Austria, 1998

[Senior 99] Senior JD, C., Dore, A., Laengle, T. and Albert, M., "An architecture for distributed diagnosis systems". *1st International IFAC Workshop on Multi-Agent Systems in Production (MAS'99),* Vienna – Austria, 1999

[Schlenoff 99] Schlenoff, C., R. Ivester, and A. Knutilla, "A robust process ontology for manufacturing systems integration", National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899. (Available at *http://www.ontology.org/*), 1999

[Edgar 00] Edgar T.F., Dixon D. A., Reklaitis G.V. "Vision 2020: Computational Needs of the Chemical Industry", University of Texas, 2000

[Williams 00] Williams Andrew B., "The Role of Multiagent Learning in Ontology-based Knowledge Management", *AAAI 2000 Spring Symposium on Bringing Knowledge to Business Processes*, Stanford University, 175-177, March 20-22, 2000.

[Kesselring 98] MacKenzie Kesselring, Inc, "XML As a Representation for Management Information – A White Paper", Version 1.0, *Distributed Management Task Force, Inc. (DMTF)*, (Available at http://www.xml.org/), 1998