GLOBAL CONDITION MONITORING SYSTEM
Implementing MATLAB®-Based Analysis Services

Henri Helanterä, Mikko Salmenperä, Hannu Koivisto
Institute of Automation and Control, Tampere University of Technology, P.O. Box 692,FIN-33101 Tampere, Finland

Keywords: Proactive maintenance, condition monitoring, fault diagnostics, distributed automation, Java, MATLAB®

Abstract: Proactive maintenance is a solution to increase the availability of the production equipment in the process industry. It involves online condition monitoring of field devices and reliably diagnosing the reason behind any abnormal behaviour, thus helping to rationalise maintenance operations. Making the information of different industrial sites available for analysis, significant improvements could be made to the predicting capabilities of condition monitoring and to the accuracy of fault diagnostics. The global condition monitoring system described in this paper is based on distributed agent-architecture and employs data communication networks to connect the industrial sites to one or more service centres. Many successful methods used in condition monitoring and fault diagnostics often require advanced tools. MATLAB® software is the de facto standard in numerical computing but integrating MATLAB® as a computing server to the J2EE-based condition monitoring system is a laborious task as no all-purpose and easy-to-use methods exist. However, this paper introduces some strategies to overcome the integration problem. The most important solution presented here is so called inverted calling scheme. Also two other approaches are discussed: using MATLAB® engine functions via C-language native methods and deployment of stand-alone MATLAB® COM components. All the above strategies have their strengths and weaknesses. Implementing the inverted call requires more effort from the programmer but is standard-compliant. Exploiting engine functions or COM components is easier as some ready-made software can be employed but the emerging solutions are not pure-Java.

1 INTRODUCTION

The role of proactive maintenance in process industry is growing in importance, since there is an ever-increasing demand for improvement of productivity and reliability of the plants. At the same time there are requirements for reducing the overall maintenance costs and carrying out the maintenance operations during the planned stoppages. Meeting these requirements—conventionally seen as opposite alternatives—means that any device failures have to be predicted reliably and the diagnosis of faulty devices has to be performed quickly and efficiently. This is not possible without highly automated condition monitoring and diagnostic systems. Individual factories, however, don't have sufficient resources to meet the demanding goals set to the maintenance. Thus, outsourcing is a potential option here.

If the huge amount of information available at the different industrial sites were available for analysis, significant improvements could be made to the predicting capabilities of condition monitoring and the accuracy of fault diagnostics. This paper first introduces the overall structure of the global condition monitoring system designed to answer the needs of predictive maintenance. Then we examine how this system can be used in data-analysis. Finally, techniques to integrate MATLAB® software as a computing server to the J2EE-based condition monitoring system are presented, including a nouvelle approach that—unlike the others—is pure-Java.

2 STRUCTURE OF THE GLOBAL CONDITION MONITORING SYSTEM

The overall structure of the global condition monitoring system is studied in (Salmenperä, 2000), (Nikunen, J. et al., 2001) and (Salmenperä, M., Koivisto, H., 2001). The system consists of devices,

Helanterä H., Salmenperä M. and Koivisto H. (2004).
GLOBAL CONDITION MONITORING SYSTEM - Implementing MATLAB® -Based Analysis Services.
In *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*, pages 300-305
DOI: 10.5220/0001134303000305

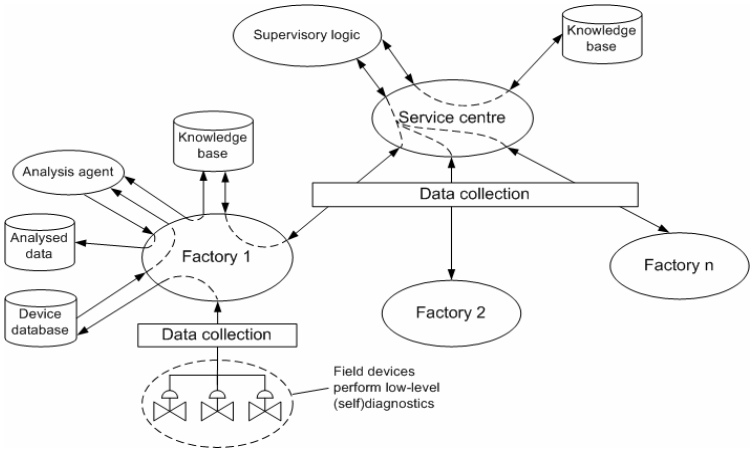

Figure 1: The analysis components of the global condition monitoring system.

databases, software components and communications links that connect all the above together. Factory-specific subsystems, sites, are connected to one or more central systems, service centres.

The system works at two levels. Automated agent-based level is meant to be the framework of the system. All continuous data acquisition, generation of regular reports and alerts takes place at this level. More specific operations are carried out by human operators. The agents provide interfaces, through which users and other agents can access system resources without having to be aware of their implementation. (Salmenperä, 2000)

Agent is a somewhat vague term. According to Franklin & Graesser (1996), the most significant characteristics of software agents are independence and goal-orientation. These also necessitate that the agents are capable of communicating with each other. In the global condition monitoring system analysis agents are used to act on behalf of human experts in routine data-analysis and fault reasoning.

Administration of the scattered large-scale system requires that all the essential information is stored in one place, in the service centre, where it can be easily accessed (Salmenperä, 2000). Employing the agent-architecture makes it possible to distribute the routine analyses to factory-level thus preventing unnecessary transfer of raw data. As a result only the most important information ends up to the service centre for further scrutiny and refinement. Extracting the raw data into interesting and valid information is one of the key tasks of the condition monitoring system.

The information within the system is of great importance for the companies and thus to be held confidential. Yet, making use of existing communication networks, mainly Internet, is essential when implementing a global system. Consequently, network security is a key issue. Possible threats include external factors but also other parts of the system, such as other industrial sites, can be seen as potential risks, since the maintenance service provider might well cooperate with rival companies. (Nikunen, 2001)

Network speed and the amount of transferred data have to be considered when designing the system in order to make it robust to possible delays and breaks in the communication. The sites are independent units that have to be able to perform basic maintenance operations without intervention of the service centre. The system has to work even if the connection to the service centre broke.

3 GLOBAL ANALYSIS SCHEME

Analysis services in the global condition monitoring system are routine analysis automatically run by the analysis agents, or more specific—but still predefined—operations put into effect by a user. The analysis services need three different kinds of databases: namely device, knowledge and analysis databases. The device database consists of measurements and diagnostic information. The knowledge base contains expert knowledge of the devices accumulated in the long run. The analysis database includes the results of analyses for later use and scrutiny.

The analysis components of the global condition monitoring system are shown in figure 1. The field devices perform low-level self-diagnostics by monitoring their own state and condition. The analysis agent responsible for the site's automated condition monitoring integrates the device

diagnostic information with the information in the site databases to carry out more thorough analyses. The information gathered at the sites is further processed and analysed by the high-level analysis services (supervisory logic) in the service centre.
The analysis operations are mainly carried out automatically by the site subsystems. However, if the site's analysis agent cannot identify the reason for the abnormal behaviour of a device—i.e., it discovers a situation that does not match to its existing deduction rules—it sends a warning to the system. Experts of the service centre then examine the device and, if the cause can be worked out, the new deduction rule is updated to the knowledge base. The experts can use, in addition to their own knowledge, the data from all the other sites, and therefore have a better chance to clear up the problem than the local staff at the site. (Helanterä, 2004)
The global condition monitoring system has to work in a diverse environment of different field devices, automation systems, database systems etc. Thus, the architecture of the system has to be flexible to accommodate the variety of industrial sites (Nikunen, J. et al, 2001). Modularity is one of the reasons behind choosing agent-architecture as the general framework of the system and modularity requirements have to be taken into account in the design of the analysis services as well. The advantage of the modular structure is that the implementation of the computational module can be changed without having to modify the analysis agent.
The computation can be transferred to a separate computing server or distributed among various machines. The information about the computing server(s) can be included in system configuration files. Using a separate computing server or various distributed servers should be considered when the amount of data to be processed increases and thus the load caused to the agent server becomes intolerable. Separating the computation from the agent server is relevant when considering the overall system stability.

4 COMPUTATIONAL IMPLEMENTATION OF THE ANALYSES

The computational implementation of the analyses means in this context the tools and mathematical libraries that are needed to perform the analyses. The way of implementation is affected by the general architecture of the system, the systems integration issues and the analysis methods to be used.
Two alternative technologies have been studied as a framework of the global condition monitoring system—namely Sun's J2EE (Kero, 2004) and Microsoft's .Net (Haavisto, 2001) and (Salmenperä, M. et al., 2003). In this paper we concentrate on the J2EE architecture and thus, the analysis components have to be compatible with Java in some way. The main driving force behind choosing the Java technology is its platform independence, which is one of the central requirements of the global condition monitoring system and has to be taken into account with the analysis services as well. On the other hand, the condition monitoring system also has to deal with legacy systems, thus making systems integration necessary. Therefore we may well ask if it is possible to use something else than a Java-based solution, provided that it is otherwise more convenient.
It is a great advantage if the analysis components could be developed and implemented with the same tool. In practise, there are two different alternatives: using Java tools both in development and implementation, or using non-Java tools in development and then integrating the emerged components to the system. In the field of numerical computing MATLAB® is the de facto standard and the development of analyses would most probably be carried out with it. Then again, the J2EE framework of the condition monitoring system makes using pure-Java applications desirable.
Basically MATLAB® consists of the computational engine, its own programming language and a variety of toolboxes that contain a combination of functions for special tasks. In the context of analysis services e.g. the Statistical, Neural Network, Fuzzy Logic and Database toolboxes provide a solid base for both developing and implementing the analyses. In the Java world there are tools for these tasks as well but they are not as advanced as those of MATLAB® and also the experience of using them is not as extensive.

5 JAVA-MATLAB® INTEGRATION

MATLAB® is a competent tool for analysis development but in terms of systems integration it is problematic. Especially the integration with Java-based systems is difficult, even though the more recent versions of the software (since version 5.3 R11) include the Java virtual machine, which makes it possible to call Java classes from MATLAB® code but the contrary is not possible. However, we present here some ways to use MATLAB® from a

Java application. The first method makes use of MATLAB®'s Java virtual machine. The latter two use MATLAB®'s external C-language interface and the ability to compile MATLAB® programs into stand-alone COM components.

5.1 Inverted call

Java Runtime class enables executing commands in the command line of the operating system. Thus, starting the MATLAB® process from a Java application is possible. As a command line option the name of an m-file—a program written in MATLAB®'s own programming language—that MATLAB® executes upon start-up can be given. Then, within the m-file we can call a Java class that listens to analysis requests from the analysis agent. The analysis request can consist of e.g. the name of an m-file to be used to perform the analysis, and the data to be analysed, or the parameters with which the data can be fetched. In the inverted calling method, which is illustrated in figure 2, the Java application that listens to the analysis requests is run in the MATLAB®'s Java virtual machine. The analysis agent, on the other hand, is run in another virtual machine. The virtual machines can reside on different physical machines. Thus, a method for these two applications to communicate over the network is needed. The simplest solution is to use Java sockets. With the help of sockets it is possible to read from and write to a defined network connection that is described by an IP address and a port number. Here, the client and the computing server exchange information through an auxiliary class called SharePoint.

The inverted call was presented by Helanterä (2004) and it works as follows (have a look at the numbers in figure 2):

The ServerStarter class starts the intermediary server Server.

The JMatlabStarter class starts the MATLAB® process from the operating system command line and gives the name of the m-file (located in the m-file repository) to be run upon the start-up.

MATLAB® calls the GetCommand class to get analysis requests from Server.

GetCommand uses a socket connection to poll Server for any analysis requests.

The analysis agent (referred as Commander in the figure 2) calls the GetArray class to get analysis result from Server.

GetArray uses a socket connection to poll Server for available results.

The analysis agent uses the SetCommand class to send a request for an analysis to be performed in the computing server.

SetCommand sends the request to Server using a socket connection.

ServerThread sends the analysis request to listening GetCommand through the established socket connection.

GetCommand returns the analysis request to MATLAB®.

MATLAB® gets the appropriate m-file from the m-file repository and performs the analysis.

MATLAB® calls the SetArray class to return the results.

SetArray sends the result to Server through a socket connection.

ServerThread sends the result to listening GetArray through the established socket connection.

GetArray returns the result to the analysis agent.

The socket-based solution is simple but probably not suitable to be used in the real-world global condition monitoring system. The analysis requests are likely to occur simultaneously and the problem with the socket-based approach is how to bind the right answer to the right request. A more elegant solution is to replace the socket server with JMS-based (Java Message Service) message queue that handles the ties between the client and the server, the request and the answer. Using JMS to implement messaging in the global condition monitoring system is studied by Kero (2004). Other alternatives, in addition to Java sockets and JMS, are e.g. making use of Java RMI (Remote Method Invocation) or CORBA (Common Object Request Broker Architecture) to do the interoperation needed.

The main advantage of the inverted call is its conformity to standards. No additional software is needed but the solution relies solely on J2EE standard libraries. The Runtime class is partly platform dependent because command syntax is system specific. Considering the minor role of the use of system commands—they are only used to start the MATLAB® process—this cannot be viewed as a stumbling block as choosing the correct syntax for each system is programmatically easy to implement.

5.2 MATLAB® engine functions

MATLAB® provides engine functions with which it is possible to use MATLAB® as a computing engine from other software. The engine functions enable executing MATLAB® commands and transferring data and variables between MATLAB® and the calling application. Engine functions are C or FORTRAN programs that communicate with the MATLAB® process via COM interface in MS Windows and named pipes in UNIX/LINUX environment. (The Mathworks, 2003)

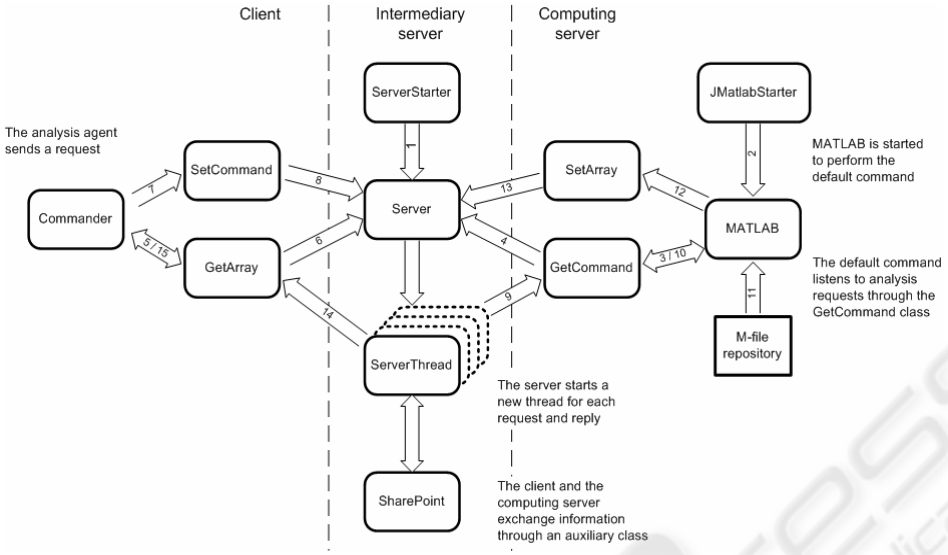

Figure 2: Calling MATLAB from Java using inverted call

Engine functions do not support Java (The Mathworks, 2003) but it is possible to use so called native methods in Java via JNI (Java Native Interface) (Sun Microsystems, 2003). Native methods are functions programmed in another language, e.g. in C, that are embedded in the Java program. JNI acts as a bridge between Java and the native program code. It is worth noting that by using JNI we lose the platform independence of Java, since native methods are operating system specific. MATLAB® engine functions can be used from a Java program by implementing engine function calls in C language native methods. There is an existing implementation called JMatLink (Müller, 2002) that uses engine functions with C native methods in Java.

5.3 MATLAB® COM Builder

COM (Component Object Model) is Microsoft's component model that makes it possible for programs to use each others' resources. The latest version of MATLAB® (version 6.5 R13) comes with the COM Builder tool that can be used to convert m-files to stand-alone COM components. A stand-alone COM component does not need MATLAB® to function, since all the required libraries are packed within the installation program of the component. The analysis modules (m-files) developed with MATLAB® can be made stand-alone COM components with the help of COM Builder. (The Mathworks, 2002)

COM components cannot be used directly from a Java program and therefore some kind of intermediation is needed. Fortunately such Java-COM bridges do exist. One useful implementation is JIntegra that uses proxy classes to enable two-way communication between Java programs and COM components. The proxy classes are pure Java applications that cast COM data types as Java data types, thus avoiding the need to use any platform-dependent code. The JIntegra software generates the proxy classes automatically for each COM component. (Intrinsyc Software International ,2004)

MATLAB® licence allows the free use of COM components and consequently the low-cost is one of the most important aspects in favouring them. On the other hand COM technology binds the implementation strictly to MS Windows environment whereas the other integration techniques presented here are also suitable to all the other operating systems supported by MATLAB®.

6 CONCLUSIONS

This paper first described the general structure and requirements set for the global condition monitoring system that is designed to achieve predictive maintenance in process industry. We also outlined the way in which the information management and analysis in the system should be done. Then three methods to integrate MATLAB® software to perform computations required by condition monitoring and fault diagnostics were introduced.

It was proposed that distributed agent-architecture is needed to guarantee modularity as well as automated operation—essential requirements for such a large-scale system. Distribution applies also to the

analysis services that are scattered to the site subsystems. On top of that, data pre-processing and device self-diagnostics are done at the field level to further decrease the system workload and to avoid unnecessary data transfer.

Systems integration has grown in importance especially in medium and large-scale industries where it is difficult to launch unified, comprehensive information solutions. The global condition monitoring system has to deal with legacy systems and some basic tools, like MATLAB®, may need to be integrated to the system as well. Modularity and generality requirements have to be considered when making the decision about the integration solution to allow for future changes to the system.

Combining MATLAB® and a Java-based system is not a straightforward task. Choosing the integration solution is a matter of deciding between ease-of-use against generality and portability of the solution. Out of the three approaches introduced in this paper only the inverted call can guarantee platform independence. The disadvantage, however, is the more complicated integration process and the more extensive programming effort compared to the other two solutions. Moreover, the simple socket-based implementation detailed in section 5.1 is not applicable to a full-scale production system. However, by employing more sophisticated inter-process communication methods, like JMS, Java RMI or CORBA, to implement the analysis request broking, the inverted call can be improved and can be seen as the most promising alternative to integrate MATLAB® software with a Java-based system.

REFERENCES

Franklin, S. & Graesser, A., 1996. Is it an agent or just a program?: A Taxonomy of autonomous Agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*. Springer-Verlag, Berlin.

Haavisto, H., 2001. *Kunnonvalvontajärjestelmän viestinvälityksen toteutusmahdollisuudet .NET tekniikoin*. MSc Thesis, Tampereen teknillinen korkeakoulu.

Helanterä, H., 2004. *Analyysipalvelut globaalissa kunnonvalvontajärjestelmässä*. M.Sc. Thesis, Tampereen teknillinen yliopisto.

Java Native Interface, 2003. [Internet]. Sun Microsystems, Inc. Available from: <http://java.sun.com/j2se/1.4.2/docs/guide/jni/>. [Accessed 24 February, 2004].

J-Integra Documentation, 2004. [Internet]. Intrinsyc Software International, Inc. Available from: <http://j-integra.intrinsyc.com/j-integra/doc/>. [Accessed 24 February, 2004].

Kero, J., 2004. *Advanced Messaging in Enterprise Scale Maintenance System*. MSc Thesis, Tampereen teknillinen korkeakoulu.

MATLAB® COM Builder User's Guide, 2002. [Internet]. Version 1. Natick, MA. The Mathworks, Inc. Available from: <http://www.mathworks.com/access/helpdesk/help/pdf_doc/combuilder/combuilder.pdf>. [Accessed 24 February, 2004].

MATLAB® External Interfaces/API, 2003. [Internet]. Version 6. Natick, MA. The Mathworks, Inc. Available from: <http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/apiext.pdf>. [Accessed 24 February, 2004].

Müller, S., 2002. *JMatLink – MATLAB® Java Classes*. [Internet]. Available from: <http://www.held-mueller.de/JMatLink/>. [Accessed 24 February, 2004].

Nikunen, J., 2001. *Security Considerations on Wide Area Networking Industrial Solutions*. M.Sc. Thesis, Tampereen teknillinen korkeakoulu.

Nikunen, J. et al., 2001. Global Condition Monitoring Network. In *Automaatio2001*, Helsinki, September 2-9. Suomen Automaatioseura ry.

Salmenperä, M., 2000. *E-speak in Enterprise Scale Condition Monitoring Network*. MSc Thesis, Tampereen teknillinen korkeakoulu.

Salmenperä, M. & Koivisto, H., 2001. Using E-speak in Condition Monitoring Network. In *Automaatio2001*, Helsinki, 2-9 September. Suomen Automaatioseura ry.

Salmenperä, M. et al., 2003. Applying .NET Framework to Conditioning Monitoring in Globally Distributed Environment. In *Automaatio2003*, Helsinki, 9-11 September. Suomen Automaatioseura ry.