

R. Plösch, R. Weinreich

An Agent-Based Environment for Remote Diagnosis, Supervision and Control

TR-SE 99.13

Copyright

© Springer Verlag, see <http://www.springer.de/comp/lncs/index.html>. Proceedings of the International Computer Science Conference, ICSC 99, Hong Kong, Dec. 13-15, 1999, Lecture Notes in Computer Science, Springer-Verlag, 1999, pp 385-392.

An Agent-Based Environment for Remote Diagnosis, Supervision and Control

Reinhold Plösch and Rainer Weinreich
Department of Business Informatics, Software Engineering
Johannes Kepler Universität Linz, Austria
{ploesch, weinreich}@swe.uni-linz.ac.at

Abstract

Mobile agent technology offers a number of characteristics that make it well suited for distributed information retrieval and monitoring tasks. We present an agent-based environment that provides facilities and tools for remote diagnosis, supervision and control of process automation systems. The described environment allows remote and dynamic configuration of diagnosis and supervision components, offers a framework for observing and controlling existing (legacy) software, and is also suited for other application domains than process automation.

1: Introduction and Overview

In this paper we describe an agent-based environment for remote diagnosis, supervision and control of process automation systems. The term agent is currently being used for a wide variety of software systems. Franklin [5] distinguishes, e.g., adaptive agents, autonomous agents, communicative agents, and mobile agents. The environment described in this paper is based on mobile agents. Mobile agents are software components with a certain degree of autonomy that have the ability to transport themselves from one location in a network to another (see [2], [5], [10]).

The work described is part of a research cooperation with Siemens Germany. The major aim of this cooperation is to explore the potential of agent-based architectures for remote diagnosis, supervision and control of steel automation systems. The reason for investigation was the realization, that the general problem frame has a number of characteristics including distributed information retrieval, monitoring and notification, as well as parallel processing, which make the problem a typical one suitable for mobile agent technology (see [2], [8], [9] and [10]). Agent-based systems have been developed in other domains as well, e.g., scientific computing, internet information retrieval, simulation, and e-commerce. A recent overview of such systems can be found in [13].

Although our investigations concentrate on the process automation domain, the developed components, frameworks, and tools, are also useful in other domains. The work addresses issues like dynamic configuration and integration of native system interfaces, which are important for mobile-agent systems in general.

In the next section we describe the primary application domain to provide a general understanding of the problem frame. In Section 3 we outline important requirements on the architecture of our environment. In Sections 4 we present the overall architecture with respect to the requirements described in Section 3. In Section 5 we describe some examples of the

currently implemented components for diagnosis, supervision and control. Section 6 gives an overview of the provided configuration tools. We conclude by summarizing the main aspects of the presented environment and by describing further work.

2: Application Domain

The initially targeted application domain is steel plant automation systems. From the business perspective it is desirable to supervise and control steel plants remotely, as it enables the company to proactively recognize problems that might affect the production process. Typical problems are:

- Decreasing performance of the automation system because of memory leaks or unplanned processor consumption.
- System deadlock because of full or corrupted file systems.
- Decreasing steel quality because of badly parameterized models.

A typical steel plant automation system consists of a number of connected hosts, often with different operating systems. The aim of the project is to implement an environment that supports the continuous supervision of such a system and that allows the specification of actions in case of problems. The two main areas of functionality to be implemented are *remote system diagnosis* and *remote system supervision and control*.

2.1: Remote System Diagnosis

In remote system diagnosis certain aspects of an automation system are observed remotely and the collected data is visualized in order to enable responsible plant supervisors to foresee problems and to timely take adequate actions.

The aspects to be monitored depend on the specific automation system and on specific customer needs. Basic requirements on the environment described in the following are the diagnosis of file system capacity, file system integrity, memory and CPU consumption of specified operating system processes and the observation of quality attributes provided by the process control software. A general requirement is that the architecture should be flexible enough to allow monitoring of arbitrary aspects of an automation system.

To facilitate diagnosis, emphasis is on a convenient presentation of the collected and condensed data in a Web browser using standard presentation techniques (HTML) or using more flexible techniques like Java applets.

2.2: System Supervision and Control

While remote system diagnosis aims at long term aspects, system supervision and control aims at supervising aspects of an automation system that require instant action in case of problems.

To facilitate the supervision and control tasks of plant supervisors, critical aspects, e.g., quality parameters, are continuously supervised. In case the specified limits for the supervised aspects are exceeded (e.g., quality parameters are out of the tolerable range), autonomous actions have to be taken by the supervising software. The actions to be taken can be specified by an administrator. Typical types of actions are automatic notification of plant supervisors, messages to local operators or autonomous changes of parameters for models like temperature or geometry models by the supervising software.

3: Requirements

A number of different requirements on the environment's implementation and design are imposed either by our cooperation partners or by the described application domain. The main requirements are:

- (R1) Portability, since a system may consist of multiple hosts with different operating systems
- (R2) Communication with existing (legacy) systems and usage of operating system services
- (R3) Dynamic configurability with tool support
- (R4) Visualization of system state
- (R5) Measurement-specific requirements
- (R6) Basic distribution issues like security, bandwidth, etc.

As described in Section 2, a typical steel plant automation system consists of multiple hosts which may have different operating systems installed. This means that portability of the supervision and diagnosis software to be installed is of importance.

The environment is intended to be used for the supervision and diagnosis of already existing plant automation systems. To obtain status information from such a system or to control it from the outside, often lower-level communication mechanisms provided for interfacing such systems have to be used. This includes mechanisms like shared memory, files and sockets, in addition to more advanced communication mechanisms like CORBA- or Java-RMI-style remote method invocation. The main communication mechanism used among the components of the environment itself, is the one provided by the underlying agent platform. The communication mechanisms used for interfacing with the existing automation software, however, are often provided by the host operating system. (This has implications on the system design to preserve portability and dynamic configurability as described in Section 4.2.) Another reason for the need of using native operating system APIs is the need for collecting data about the usage of system resources like hard-disk space and memory- or CPU-consumption of certain automation tasks as described in Section 2.

The third main requirement is the need to configure the environment remotely and dynamically (i.e., at run time). This includes dynamic installation and de-installation of system components, and dynamic configuration of component interactions and properties via an internet connection. Configuration of a running system over time may impose versioning problems which have to be handled properly, also. Further, it has to be considered that administration may take place using temporary connections (established, e.g., via a modem) and that the administration node may be a notebook which may even shut down completely. Both temporary connected communication lines and shutdown of the administration host have significant influence on the design of the environment. An example is the issue of transferring code, which cannot be fetched on demand if the connection might be terminated at some point of time in the future.

In addition to remote and dynamic configuration the environment has to provide information about its state to the administrator. This does not mean information about the automation system (which is the central function of the environment anyway), but information about whether subsystems of the automation system are currently supervised or whether some measurement is currently active.

To collect data about the state of the automation system, measurements have to be active either sequentially or in parallel. For example, the system has to be able to measure CPU-

consumption of certain automation tasks at multiple hosts at the same time. In addition, since some measurements may be active for hours or days it must always be possible to obtain the results of incomplete measurements and to generate temporary reports. This has implications on the communication structure of certain parts of the system (e.g., whether communication is synchronous or asynchronous).

The last main requirement concerns basic distribution issues like security and bandwidth use. The administration and configuration of the system via an arbitrary internet connection makes a number of demands on security issues, like ensuring the authenticity of the administrator and preventing the alteration of data and code in transit. For a more complete list of security issues to be considered, see, for example, Farmer [4]. Another aspect that has to be taken into consideration is that code and data for system configuration and control has to be transferred over sometimes very slow communication lines. Thus reduction of bandwidth use where possible is desired.

4: System Architecture

To provide an understanding of the chosen architecture and of how the requirements listed in the previous section are handled in the current implementation, we will first describe the implementation platform used and then give an overview of the architecture of the two main functional areas, remote diagnosis, and system supervision and control.

4.1: Implementation Platform

The decision for using a mobile agent platform was made before the project was started, because—as stated in the introduction—one of the project aims was to investigate the usefulness of agent technology in this domain. We chose IBM's Aglets SDK [8] as the base platform, because of both technical and practical reasons. From a technical perspective, the Aglets system is implemented in Java (see [1] and [7]) which offers a certain degree of portability and eases the integration of (Java) third party components because of the standardization provided by the Java platform. As stated in Section 3, portability is an important requirement since an automation system may consist of hosts with different operating systems like Solaris and Windows NT (see R1 in Section 3). The agent platform also provides a number of security options (although not completely sufficient for our application) and, of course, basic functionality for mobile agents, like object migration and code transport. From a practical perspective, the Aglets SDK is available for free (at the time of writing) and has attracted the attention of a significant number of engineers and scientists.

4.2: Remote System Diagnosis

Figure 1 gives an overview of the architecture of the part of our environment realizing remote system diagnosis. We group the hosts that are part of an automation system into units. In fact, a unit is just a collection of hosts, which means that not only hosts that are part of automation systems may be grouped into units. Typically the members of one unit are connected by a local area network. One host per unit that serves as an entry point for all externally arriving agents is called the unit gateway. The distinguishing feature of the unit gateway is that it hosts a special agent, called the Unit Agent, which maintains information about the unit in general and about the installed automation system in particular. For example, the Unit Agent knows about the hosts that are part of the automation system and which operating systems are installed on these hosts. The Unit Agent is also able to provide information about how to

communicate with the automation system, i.e., which (potentially operating system dependent) communication service to use.

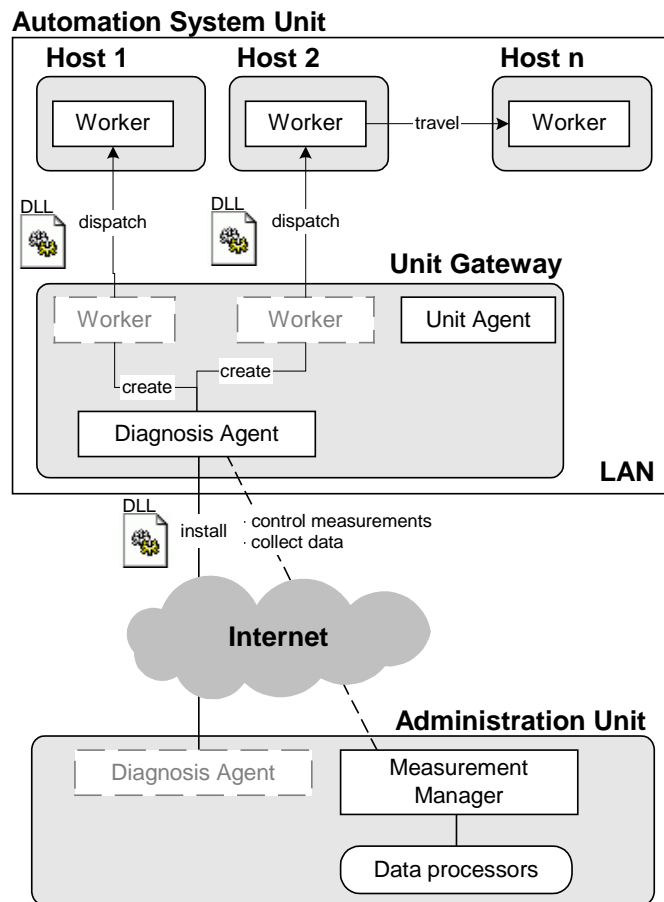


Figure 1: Architecture for Diagnosis Agents

An administrator at a remote administration unit, which is connected with the automation system unit via a (potentially temporary) internet connection, may create and configure an arbitrary diagnosis agent, and install it at the automation system. Some diagnosis agents may need additional operating-system specific code (see R1 and R2 in Section 3) to perform their diagnosis task, e.g., for measuring CPU-consumption of specified processes. During the installation process, first the agent code and sometimes the code for agents performing subtasks on behalf of the diagnosis agent are transferred from the administration unit to the automation unit. On arrival at the administration unit gateway the agent determines which platform specific libraries are needed by contacting the responsible Unit Agent. Any platform specific native libraries (DLLs) are only transferred from the administration unit if the proper version for a specific operating system is not available at the automation system unit (see R6 in Section 3).

After the installation phase the connection between the administration unit and the automation system unit may be terminated. The diagnosis agent rests in an idle state until it is told to fulfill its measuring task. For starting and terminating measurements and for processing measurement results we provide a tool called Measurement Manager. The measurement process itself depends on the diagnosis agent. Sometimes data has to be determined at a

specific host at a certain point in time, sometimes data has to be collected from several hosts in parallel over a longer period (see R5 in Section 3). In the latter case, the diagnosis agent creates a number of worker agents which it sends to each host where certain system characteristics are to be determined. Each worker agent carries the proper platform specific library for his target host and immediately starts the measurement after arrival at the target host. In cases where parallel measurement is not necessary the diagnosis agent may only create one worker agent which visits all hosts sequentially. Measurement results are sent automatically to all registered measurement managers, which may be located at different units. Data can also be obtained manually as described below. Workers that are not continuously measuring dispose themselves from the target host after the measurement is finished. Workers for continuous measurements are supplying the diagnosis agent at predefined rates with temporary results and rest on the target host until measurement is explicitly stopped by the administrator.

An administrator can obtain the collected data from an arbitrary diagnosis agent at any time by means of the Measurement Manager tool. If a measurement is not yet complete, temporary results are delivered if available. Each Measurement Manager can be configured with data processor components, which are used for analyzing and processing the selected data. We provide data processors that are able to generate HTML-pages for visualizing the results. New data processors, e.g., for performing statistic evaluations or for storing the data in a database can be loaded at runtime (see R3 in Section 3).

The system is implemented in a way that the administration unit may be disconnected or may shutdown at any time. Connections with diagnosis agents are regained automatically upon system startup.

4.3: System Supervision and Control

System supervision and control provides facilities for observing critical aspects of the automation system continuously and for automatic reactions in case of problems. Like diagnosis agents, agents for supervision and control are created and configured at a remote administration unit and installed at an automation system unit afterwards. In contrast to diagnosis agents, which may only be installed at the unit gateway, agents for supervision and control may be installed at any host of the automation system as depicted in Figure 2.

After installation supervision agents are continuously observing a certain aspect of the automation environment. We have implemented agents for supervising resource consumption (like disk space, memory usage and CPU time) and process control data (like quality parameters). The latter are determined by analyzing log files and by communicating with the automation system via shared-memory segments and sockets. The supervision agents do not initiate any action directly when some critical system state has been detected. Instead, they may be connected to an arbitrary number of control agents, which finally perform the specified action.

Control agents are just waiting to be notified about some critical system state by a supervision agent. Examples for control agents are an agent sending emails to plant administrators, an agent for placing a message at local operator's control panel, and an agent switching the process automation task to another (more stable) version. Some examples of implemented agents are given in Section 5.2. Each supervision agent may be connected to an arbitrary number of control agents and vice versa. The connections between supervision and control agents are either set up remotely using the configuration tool described in Section 6 or negotiated dynamically using a trader agent that is located at every host (see Figure 2).

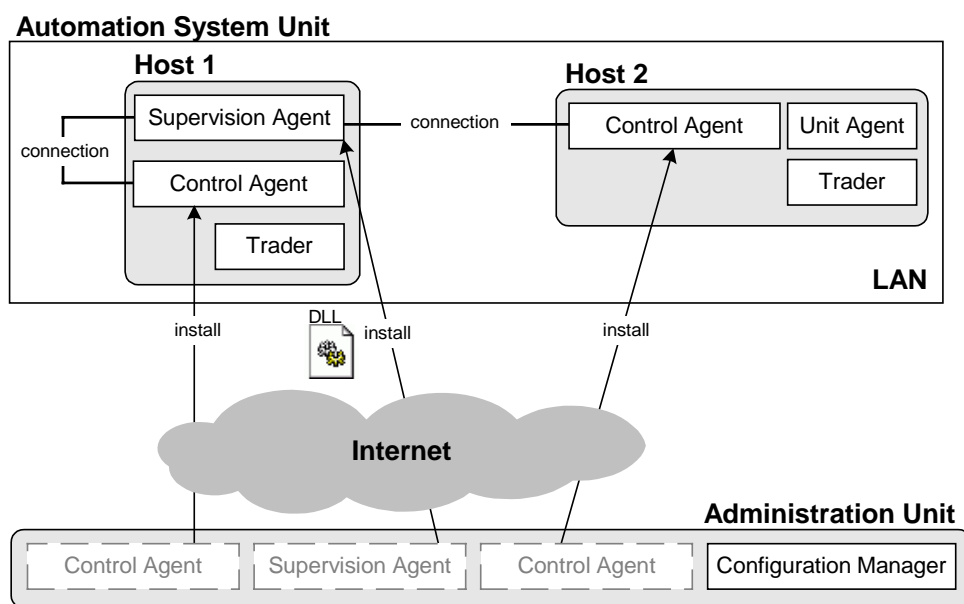


Figure 2: Architecture for System Supervision and Control

5: Sample Components

In the following we describe some examples of the currently implemented components. In Section 5.1 we illustrate system diagnosis and report generation. In Section 5.2 we give an example for agents used for supervision and control.

5.1: Continuous Memory and CPU-Measurement

A common task when supervising process automation systems is to monitor memory and CPU-consumption of selected operating system processes. Figure 3 depicts a dialog for configuring a diagnosis agent for memory and CPU measurement. The general settings (hidden in Figure 3) allow the configuration of the name of the agent and the address of the unit gateway. In addition, a list of hosts where measurement should take place may be provided. If no host is specified the performance measurements take place at every host of the automation system.

Beside these general settings the measurement process itself can be parameterized. In the example depicted in Figure 3 the memory and CPU-consumption of the specified operating system processes (in our example only a process named tracker) is measured every 60 minutes (Time between measurements). Ten measurements are taken over a period of one minute, which means that every six seconds measurement data is collected for the specified operating system process. The average values of these ten measurements is a temporary result that is sent to the diagnosis agent at the unit gateway by each of its workers (see Section 4.2). In the example above twenty-four average values can be stored, i.e., the measurements of a whole day.

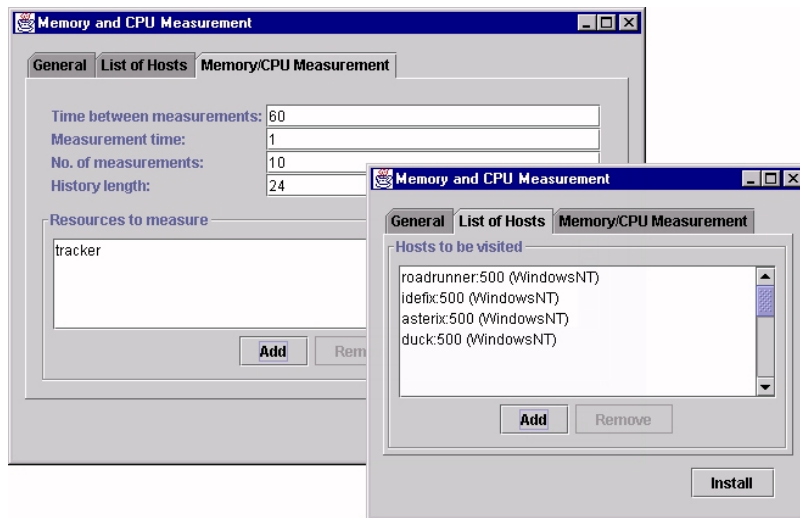


Figure 3: Memory and CPU Measurement

Whenever measurement results are transferred from a diagnosis agent to a Measurement Manager a report like the one depicted in Figure 4 is updated. The report in this figure shows real memory and virtual memory usage of the monitored processes at the different hosts on its left hand side, and their CPU consumption in percent on the right hand side.

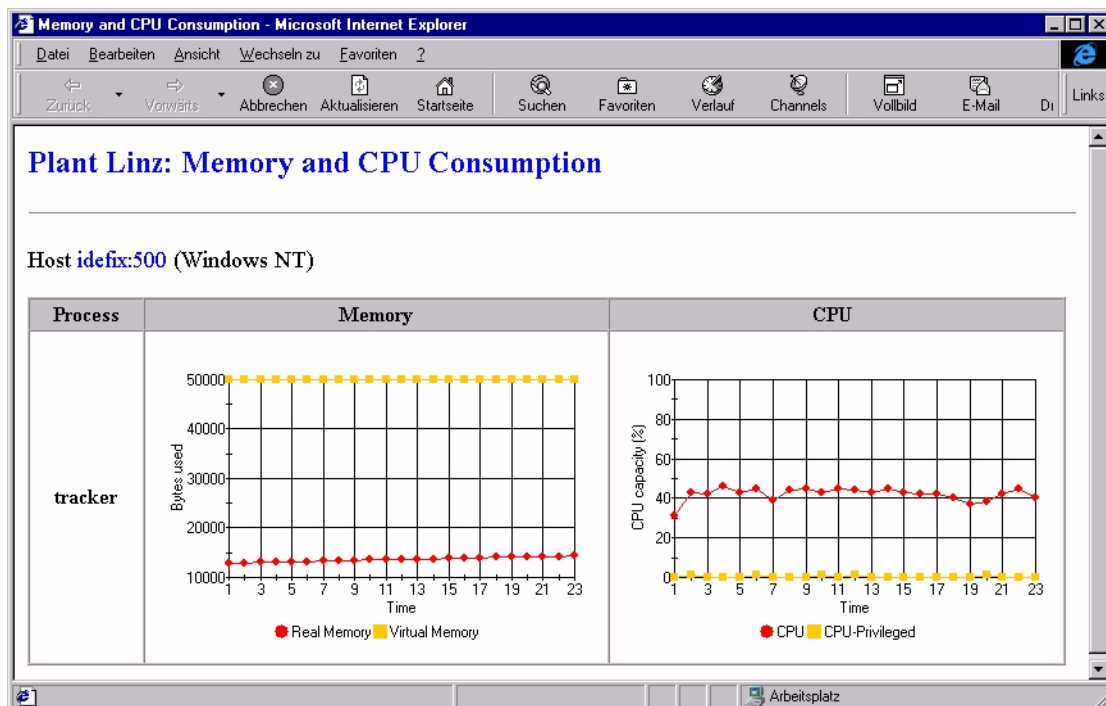


Figure 4: Report for Memory and CPU Measurement

5.2: Supervising Quality Attributes of the Production Process

The rolling mill automation system we are dealing with in our cooperation project measures profile and planeness tolerances for every slab or strip and places this data in a shared-memory segment. These quality attributes, which are of course only a small fraction of the quality attributes that would have to be supervised in a full fledged quality supervising system, are continuously supervised by a Quality Supervising Agent whose configuration user interface is depicted in Figure 5.

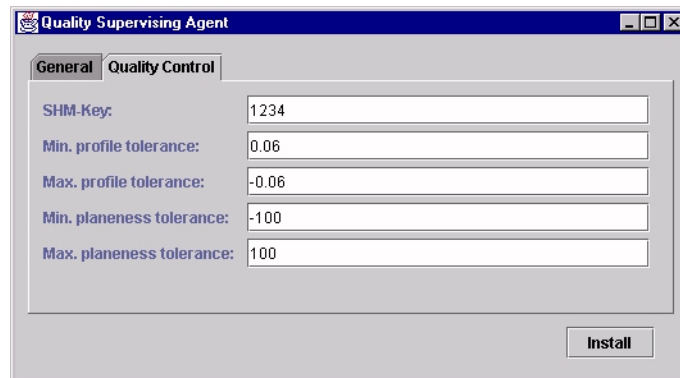


Figure 5: Quality Supervising Agent

Besides general settings (hidden in Figure 5) like the name of the agent, the name of the host and the checking interval, a shared-memory key (which is used by the supervised process automation system) and upper and lower limits for the profile and planeness tolerance may be specified. After successful installation (see Section 4.3) the agent checks in defined intervals whether the limits of the quality attributes are exceeded. In case of violated limits the specified control agents (see Section 4.3 and Section 6) are informed.

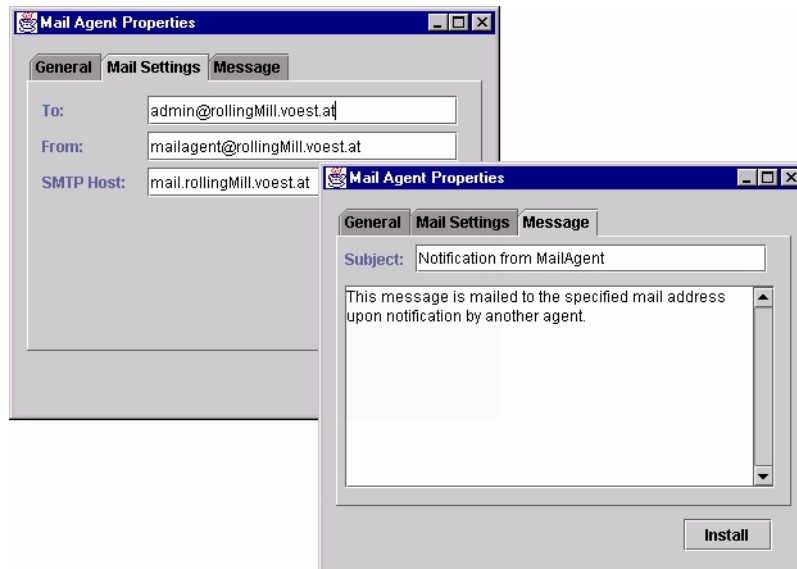


Figure 6: E-Mail Agent

A proper reaction for violations of planeness and profile tolerance is to inform a human person, the plant supervisor, by means of electronic mail. An e-mail agent which performs the e-mail sending task is depicted in Figure 6. It allows the configuration of receiver, sender and SMTP-host for an e-mail message. It is also possible to compose a standard message. The Quality Supervising Agent (or any other supervising agent connected to an e-mail agent) may simply send a general notification to the e-mail agent or alternatively pass a more specific message to it. In the first case a predefined message is sent, in the latter case the provided alternate message is mailed to the specified receiver. Additional or alternate reactions can be specified dynamically using the configuration tools described in the next section.

6: Configuration Tools

An important aspect of our system is the ability for remote and dynamic configuration (see R3 in Section 3). We currently provide two tools for the different functional areas described in the sections above. The Measurement Manager depicted in Figure 7 is used for controlling (long-term) measurements and report generation. It gives an overview of all diagnosis agents installed at a particular unit and also indicates their current state (see R4 in Section 3), e.g., measurement active or terminated.

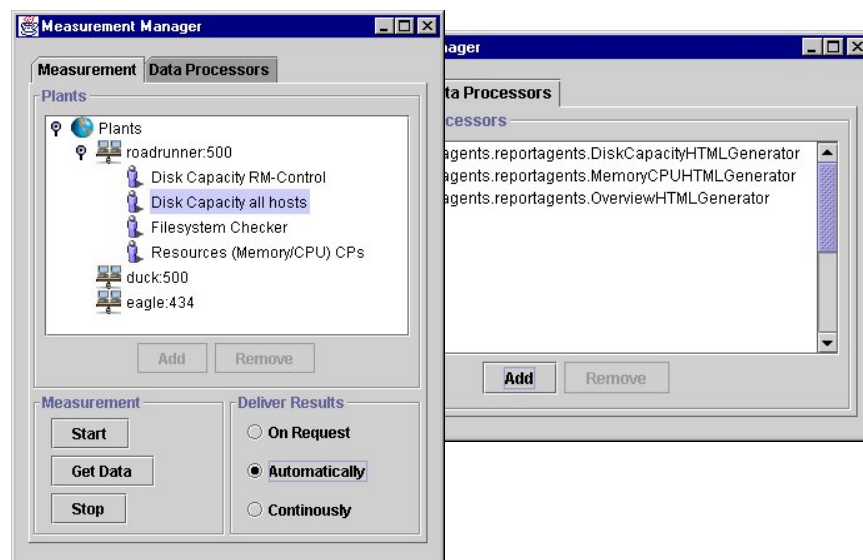


Figure 7: Measurement Manager

The units to be controlled—which are different automation plants in our case—are represented by the DNS names of the respective unit gateway hosts, each hosting a number of diagnosis agents. Each of these agents can be selected and asked to start a new measurement, to terminate the currently active measurement or to deliver temporary results of the currently active measurement. The view is updated if agents are disposed or if new agents are installed (possibly from some other remote location). The Measurement Manager uses data processors for processing measurement results. Data processors can be provided as pluggable components, which may be added or removed to/from the tool dynamically. Most of the data processors that are currently available are HTML generators for reports that are to be displayed in a Web-Browser.

The tool used for managing connections among supervision and control agents remotely and dynamically is called Configuration Manager and is depicted in Figure 8. The main window of the Configuration Manager displays not only the different units (again represented by the DNS name of the unit gateway) but also all available hosts at each particular unit and all agents at each particular host.

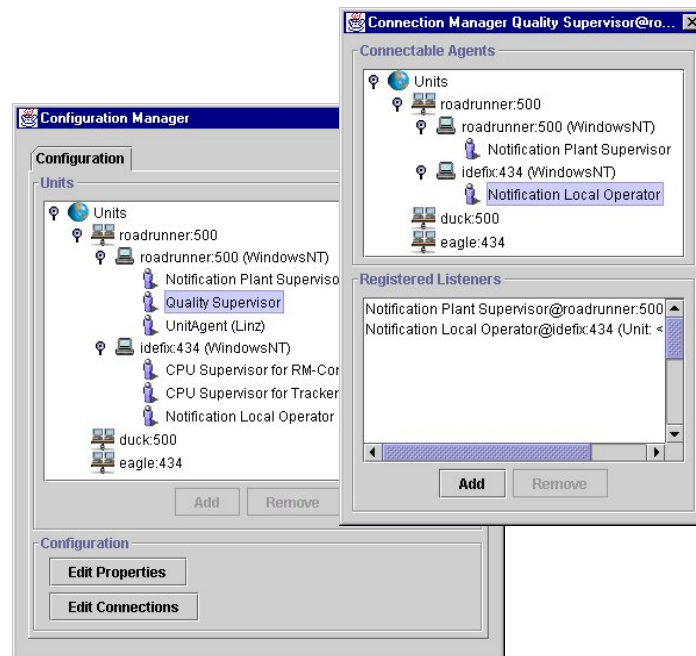


Figure 8: Configuration Manager

If a supervision agent is selected, its connections to control agents can be modified with the Connection Manager also shown in Figure 8. The Connection Manager only shows agents that can be connected to the specified supervision agent. In the example above two e-mail agents are connected to an agent signaling significant changes of certain quality parameters. The available connections and views are correctly updated if agents are disposed, new agents are installed, and if an agent changes its location. Both tools are using caching strategies for reduced bandwidth use and increased performance.

7: Status and Further Work

A focus of current work is on improving the framework aspect of our environment. The aim is to reduce the development effort for new agents requiring different strategies for installation (especially code and DLL management), data collection, measurement tasks, communication, etc. Framework design requires a number of redesign cycles to reach a somewhat mature state (see [6]). At the time of writing, a number of new requirements demand additional redesign cycles to increase the reusability of the existing code. We are also still working on native code management for different operating systems.

Some topics of future work include better management of an agent's user interface code for configuration (currently each agent carries the UI code along when migrating to another location), improved version management, and especially improved security. At present we use the security mechanisms of the underlying agent platform, which provides security only for communication among the hosts of the same internet domain, which are all to be trusted. Since

the current version of the agent platform is not yet based on the Java 2 platform, we expect the available security options to be improved in future versions.

The development of the system has also made the lack of tools for debugging and monitoring multiagent systems obvious. We have developed monitoring tools for distributed applications in previous projects (see [12]) and thus plan to investigate techniques and tool support for the additional demands imposed by mobile agent systems (like object migration and code mobility).

8: Conclusion

White [10] describes increased performance (because of local instead of remote calls after object migration), the support of temporary connections and improved customization as main advantages of mobile agent technology. These characteristics make mobile agents well suited for the implementation of supervision and control tasks in automation systems. The software agents for system supervision and diagnosis are de-coupled from the core automation system, can be designed independently and (de)installed dynamically. Only in cases where the agent environment needs data that cannot be requested via some kind of interface to the automation system, the observed system itself has to be modified to provide hooks for determining the required data.

Main issues of our environment are the flexible support for interfaces to already existing automation software on different operating systems, frameworks for measurement activities and agent interactions, and tool support for remote and dynamic configuration. Further work includes improved native code management, better management of an agent's user interface code and improved configuration support. We also plan to investigate appropriate monitoring and debugging support for agent-based systems.

9: Acknowledgements

The work presented has been sponsored and supported by Siemens AG Germany. We would like to thank Christa Schwanninger, Thomas Heimke, Joachim Höhne and Einar Bröse for their cooperation and support.

References

- [1] Arnold K., Gosling J.: "The Java Programming Language", Second edition, Addison-Wesley Longman, Reading, MA, 1998
- [2] Cabri G., Leonardi L., Zambonelli F.: "Mobile Agent Technology: Current Trends and Perspectives", AICA 98, Napoli, Italy, Nov. 98.
- [3] Chang D., Lange D.B.: "Mobile Agents: A new paradigm for distributed object computing on the WWW", in Proceedings of the OOPSLA96 Workshop: Toward the Integration of WWW and Distributed Object Technology, San Jose, California, October 1996, ACM Press, New York, 1996, pp 25-32
- [4] Farmer W.M., Guttman D., Swarup V.: "Security for Mobile Agents: Issues and Requirements", in Proceedings of 19th National Information Systems Security Conference (NISSC96), 1996
- [5] Franklin S., Graesser A.: "Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents", in Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996
- [6] Gamma E., Helm R., Johnson R., Vlissides J.: "Design Patterns", Addison Wesley Longman, Reading, MA, 1995

- [7] Java: Information on Java releases and online documentation can be found at java.sun.com, Sun Microsystems, Inc.
- [8] Lange D.B., Oshima M.: "Programming and Deploying Java Mobile Agents with Aglets", Addison-Wesley Longman, Reading, 1998
- [9] Lange D.B., Oshima M.: "Seven Good Reasons for Mobile Agents", Communications of the ACM, Volume 42, Number 3, March 1999, pp 88-89
- [10] White J.: "Mobile Agents", in "Software Agents", J. Bradshaw, Editor, MIT Press, 1997, pp. 437-472
- [11] White T., Pagurek B., Bieszczad A., "Network Modeling For Management Applications Using Intelligent Mobile Agents", accepted for publication in a special issue on Mobile Agents of the Journal of Network and Systems Management, to be published in September, 1999
- [12] Weinreich R., Kurschl W.: "Dynamic Analysis of Distributed Object-oriented Applications", Proceedings of the 31st Hawaii International Conference on System Sciences (HICSS-31), Software Technology, Big Island of Hawaii, USA, January 6-9, 1998, IEEE Computer Society Press 1998.
- [13] ACM: "Multiagent Systems on the Net and Agents in E-commerce", Communications of the ACM, Volume 42, Number 3, March 1999.