Title: HANDLING A REQUEST IN AN AUTOMATION SYSTEM

Abstract: The invention relates to a method for handling requests within an automation system and to an automation system having means for carrying out a method of this type. To allow efficient handling of requests sent to the automation system within the automation system, the following method steps are proposed: - at least one mobile software agent is generated using as a first component of the automation system on the basis of a request which is sent to the automation system and received by the first component, - the software agent is being forwarded to first execution means, and - the request is at least partly handled by executing the software agent using the first execution means.
Description

Handling a request in an automation system

The invention relates to a method for handling requests within an automation system and to an automation system having means for carrying out such a method.

Such a system or method is used in automation engineering, for example for a man-machine interface in the form of a remote station to interact with various components of an automation system. Here and in the whole document, the term automation system is to be understood to mean a system which uses technical means to form particular operations without any human influence, partly or entirely according to prescribed programs. In this context, operations denote, by way of example, production processes, assembly processes, transport processes, processes for quality control etc. taking place in the industrial environment in particular.

In today's product and process automation systems, process-engineering installations are normally controlled using programmable logic controllers, which have a CPU suitable for executing a control program. Depending on the complexity of the process to be controlled, an automation system may also have a plurality of programmable logic controllers (PLCs) coupled to one another. To control the various automation processes, a plurality of CPUs are therefore provided and hence the hardware resources are increased.

The various automation devices normally communicate with the programmable logic controllers using input/output assemblies. The input/output assemblies convert the field signals, detected at the installations using sensors, for example, into a form suitable for a system bus or for a programmable logic controller connected to the system bus. Input/output assemblies may also be local, where they are linked not to a system bus but rather to an intranet or to the Internet.
put/output units frequently themselves have their own CPU.

If a plurality of CPUs are present in the automation system and are involved in controlling the automation process, the control and process data are normally also distributed locally in the system. In the event of the automation system being reconfigured, the distribution of the computer and memory utilization within such a system changes. If the system configuration is altered, the whole system therefore needs to be reconfigured.

The invention is based on the object of allowing efficient handling of requests sent to an automation system within the automation system.

This object is achieved by an automation system having at least one first component, which has:

- generation means for generating at least one mobile software agent on the basis of a request which is sent to the automation system and received by the first component, and
- forwarding means for forwarding the software agent to first execution means which are suitable for at least partly handling the request by executing the software agent.

This object is also achieved by a method for handling requests within an automation system, having the following method steps:

- at least one mobile software agent is generated using a first component of the automation system on the basis of a request which is sent to the automation system and received by the first component,
- the software agent is forwarded to first execution means, and
- the request is at least partly handled by executing the software agent using the first execution means.

In the inventive automation system or method, a request sent to the automation system results first of all in the mobile
software agent being produced. A software agent is to be understood to mean a computer program which has at least the following properties:
- the program operates autonomously, i.e. largely independently of user interventions,
- the program operates proactively, i.e. it initiates actions on the basis of its own initiatives,
- the program operates reactively, i.e. it reacts to changes in the environment,
- the program is social, i.e. it is able to communicate with other software agents.

In addition, a software agent may have the properties of being able to learn, the capability of drawing conclusions and the capability of changing its behavior.

A mobile agent additionally has the ability to change the execution means for executing it. By way of example, a mobile software agent is able to migrate from one hardware component to another. Alternatively, it can "migrate" from one software component to another, for example from a virtual machine to a web server, so as to be successively executed by both software components. By way of example, when a request is received by the first component which requires handling by a web server, the mobile software agent is forwarded to the execution means which, by way of example, are in the form of a web server in this case. In this context, the web server may be installed on the first component or on another component, networked to the first component.

Production of the software agent allows the request underlying it to be handled extremely flexibly. Many industrial processes and also many industrially produced products are frequently subject to a change in specifications, on the basis of which the corresponding control tasks also need to be frequently changed. In such cases, the automation system normally needs to be reconfigured. The control programs running on the CPUs need to be updated and, depending on the utiliza-
tion level of the existing hardware resources, redistributed over the various intelligent components of the automation system. Failure of a component in the automation system also frequently requires redistribution of the control tasks over the various intelligent components which are present in the system.

When the hardware structure of the automation system has been altered, the memory locations and hence the access paths to the various data present in the system are also altered. To allow a user to access such an automation system, known systems therefore also require reconfiguration of the communication system.

By contrast, such reconfiguration of the data access operations is not necessary when the inventive system and method are used. When a request is made to the automation system or to a component of the automation system, the component receiving this request first of all generates the mobile software agent. Particularly if the first component does not have means for fully processing the request, an embodiment of the invention in which the automation system comprises at least one second component which is networked to the first component and which has the first execution means is advantageous.

In this context, a further refinement of the invention is advantageous in which the first component has second execution means for partly handling the request by executing the software agent. When the request has been partly handled, the software agent is forwarded to the second component or to the first execution means present on the second component.

Full processing of the request is not inevitably ensured even after subsequent execution of the software agent using the first execution means. Rather, in a further advantageous refinement, the second component may have further forwarding means for forwarding the software agent to third execution means, the third execution means being suitable for at least
partly handling the request by executing the software agent and, in particular, being part of a third component of the automation system, which component is networked to the first and/or second component. The mobile software agent is thus able to change from one component to the other component within the automation system and thus to allow the request which forms the basis of generation of the software agent to be handled on the various components of the automation system.

The use of mobile software agents for handling a request sent to the automation system has the advantage that the precise hardware configuration of the automation system does not need to be known in full. A software agent generated on the basis of a request sent to the automation system migrates within the networked automation system from one component to the next until the associated request has been processed fully. By way of example, communication with other software agents advantageously allows the software agent generated on the basis of the request even to specifically look up a component which is suitable for execution.

In many cases, particularly efficient handling of the request can be achieved if, in a further advantageous refinement of the invention, the second component has further generation means for generating at least one further software agent after the software agent has been forwarded to the first execution means. The further software agent (8) can be regarded as subagent (9) produced on the basis of the received software agent. Next, the subagents are forwarded within the automation system and are executed on various components. By splitting the software agent into subagents, the underlying request can be performed in parallel using various execution means, which are sometimes also distributed over various components.

The handling of a request sent to the automation system using one or more software agents is expedient, particularly in one
advantageous embodiment of the invention, when the automation system has more than one networked component with a real-time-compatible processing unit for executing control programs. Since a request to the automation system is normally handled by such an intelligent component, the use of software agents for a plurality of intelligent components in the automation system is particularly advantageous. This applies particularly in one advantageous embodiment of the invention, in which the automation system has means for dynamically distributing control tasks among the networked components with a real-time-compatible processing unit. In such an automation system, the control tasks or the corresponding control programs are dynamically distributed within the automation system, for example by a central entity, in line with the resources and the utilization level of the individual networked components. Such adapting of the distribution of the control tasks is also effected when a component of the automation system is removed or added. When the hardware configuration of the automation system is altered, the resources and the utilization level within the system are frequently also altered, which can be compensated for by dynamically redistributing the control tasks. In a distributed automation system of this type, in which the computation power can change dynamically, the handling of a request sent to the system using mobile software agents is extremely advantageous. When a user sends such a request to the automation system, for example, the underlying hardware configuration does not need to be known for this. Rather, the request is taken as a basis for configuring a software agent, which automatically looks for the execution means suitable for handling the request within the automation system.

In a further advantageous refinement of the invention, the first component has a man-machine interface which allows the request to be initiated by a user. In such a case, the component is in the form of a PC, for example, with the PC being used by the user to operate and/or observe the components in the automation system. If the user wishes to use the PC to
access data in the automation process, it is not necessary to know at this point where and using what interfaces the requested data are accessible within the system. On the basis of the request, a software agent is generated which is used to collect the data from the automation system.

Also advantageous is an embodiment of the invention in which the first component can be accessed via the Internet or via an intranet. In the case of such an embodiment, for example, a user using a PC at a remote location can access the automation system via the Internet in order to read process values or to configure individual components of the networked automation system. Such access can be implemented, by way of example, by the remote PC in the form of "Remote Method Invocation" (RMI). Such method invocation executes a method for a remote Java object running on a virtual machine on a computer which is remote from the PC. In addition, a "Remote Procedure Call" (RPC) can be initiated for such a call. RPC is a network protocol which allows function calls on remote computers. If the PC is now used to send an RPC or an RMI to the first component in the automation system, the first component generates a software agent which is suitable for such a call. Such a software agent then migrates through the networked automation system and is executed using the execution means of the components in the automation system in order to handle the appropriate request in this way.

In a further advantageous refinement of the invention, the first execution means and/or, if present, the second and/or, if present, the third execution means are in the form of a, in particular real-time-compatible, virtual machine. The use of a virtual machine for executing a software agent has the advantage that a virtual machine can be used to implement platform-independent execution of the software agent. Real-time compatibility of the virtual machine allows synchronization to be implemented when executing the software agent using individual control processes.
Web-based access to the networked automation system is particularly simple to implement when, in a further advantageous refinement of the invention, the software agent contains Java objects. In such an embodiment of the invention, it is expedient to provide a Java Virtual Machine for executing the software agent in the component(s) of the automation system.

To make the handling state of the software agent transparent, an embodiment of the invention is advantageous in which the software agent has at least one state variable for indicating the handling state of the request. Such a state variable can be used to decide, for example, whether the software agent should be forwarded within the automation system for the purpose of handling the request.

To communicate the result of the handling to a remote station, for example, following full handling of the requests, a refinement of the invention is advantageous in which at least one of the components of the automation system has means for producing response data for the request following full handling of the request.

To ensure real-time compatibility for handling the request, an embodiment of the invention is advantageous in which the automation system has means for synchronizing the at least one software agent to a system cycle in the automation system.

To allow web-based access to the components of the automation system, an advantageous refinement of the invention is characterized in that at least one of the components of the automation system has an interface based on an OPC (OLE for Process Control) standard. A conceivable example in this context is a form as an OPC-XML interface or OPC-UA (United Architecture) interface.

The invention is described in more detail and explained below using the exemplary embodiments illustrated in the figures,
in which:

FIGURE 1 shows a schematic illustration of a method for handling a request sent to an automation system, FIGURE 2 shows a first embodiment of an automation system with networked components, and FIGURE 3 shows a second embodiment of an automation system with networked components.

FIGURE 1 shows a schematic illustration of a method for handling a request 4 sent to an automation system. The request 4 has been generated by a user, for example, using a PC connected to the Internet in order to read in data from the automation system. The request 4 reaches a first component 24 of the automation system via the Internet. The first component 24 of the automation system generates a mobile software agent 3 on the basis of the request 4.

The aim is to handle the request 4 fully by executing the mobile software agent 3 using various execution means 25, 27, 29 which are present within the automation system. First execution means suitable for this purpose are located within a second component of the automation system 26, second execution means 27 which are suitable for this purpose are located within the first component 24, and third execution means 29 which are suitable for this purpose are located within a third component 28 of the automation system. Before the mobile software agent 3 is forwarded to the first execution means 25, which are located within the second component 26, it is executed using the second execution means 27. By way of example, these execution means are, in particular real-time-compatible, virtual machines which can be used to execute the mobile software agent 3 independently of the platform.

Executing the mobile software agent 3 using the second execution means 27 does not result in full handling of the request 4, as is to be assumed in this example. Therefore, the mobile software agent 3 is forwarded to the second component 26 or
to the first execution means 27 accommodated within the second component 26 after execution within the first component 24.

It will be assumed here that executing the software agent 3 using the first execution means 25 also does not yet result in full handling of the request 4. Therefore, the mobile software agent 3 is forwarded from the second component 26 to a third component 28, which has the third execution means 29 for executing the mobile software agent 3. Executing the mobile software agent 3 on the third component 28 of the automation system finally results in the request 4 having been handled to the full extent.

While the mobile software agent 3 was being handled on the various components 24, 26, 28 of the automation system, the mobile software agent 3 collected various data, which are finally returned to the user by the third component 28 of the automation system in the form of response data 22, again via the Internet or an intranet, for example.

An advantage of the system presented here is that the request 4 was able to be sent to the automation system without the need to know which of the components 24, 26, 28 has suitable means for handling the request 4. Rather, the first component 24 receiving the request 4 generates a mobile software agent 3, which largely independently takes care of handling the request 4. In the example illustrated here, all three components 24, 26, 28 are involved in handling the request 4. This is naturally not necessarily the case and is just one particular embodiment of the invention. By way of example, it might also have been that just executing the software agent 3 using the third execution means 29 is sufficient for fully handling the request 4. In this case, the mobile software agent 3 would have been forwarded directly from the first component 24 to the third component 28. Depending on the intelligence which the mobile software agent 3 has, it automatically looks for the appropriate execution means 25, 27,
29 within the automation system. By way of example, it finds its target by communicating with other software agents migrating around within the automation system.

FIGURE 2 shows a first embodiment of an automation system with networked components. The networked automation system comprises a first subnetwork 7 and a second subnetwork 8, which are connected to one another via an industrial Ethernet. To network the two subnetworks 7, 8 and also the individual components of the automation system, the PROFINET standard is used for the industrial Ethernet. The test below will explain an interaction within such an automation system by way of example.

Within the first subnetwork 7 there is a remote station 9 in the form of a PC. It is now assumed that a user wishes to use the PC 9 to effect read and/or write access to data in the automation system, particularly to the second subnetwork 8. The automation system is a distributed automation system which is distinguished in that the control programs required for process control are distributed over various intelligent components of the automation system. In this context, the distribution of the computation load can be dynamically adapted. Both the resources of the individual components and the utilization level thereof can be taken into account in this context.

To allow a user of the remote PC 9 to interact with such an automation system without this requiring detailed knowledge of the hardware configuration of the automation network, a request 4 generated by the PC 9 to the automation system is converted into a mobile software agent 3. The PC 9 has an installed web browser 20 which is used to generate the request 4, for example in the form of an RMI (Remote Method Invocation) or ORB (Object Request Broker), using an applet and a non-real-time-compatible virtual machine 10.

An Internet Service Provider (ISP) 11 and a Switch and Secu-
Security Gateway 12 route the request 4 to a first component of the automation system. The first component is an input/output unit 1, to which industrial field transmitters such as sensors and/or actuators are connected. The input/output unit 1 has a real-time-compatible virtual machine 6 and an OPC-XML server 13 on it.

As illustrated in enlarged detail, the real-time-compatible virtual machine 6 executes a demon process 14 which waits for requests at a port of the input/output unit 1. Such a demon 14 may be programmed using Java, for example, in which case the real-time-compatible virtual machine 6 is a Java Virtual Machine. When the request 4 is registered by the demon 14, a mobile agent server 2 (likewise running on the real-time-compatible virtual machine 6) is used to generate a mobile software agent 3 on the basis of the request 4. In addition, the real-time-compatible virtual machine 6 runs a Distributed Service Provider 15 providing an infrastructure in order to connect various services provided within the distributed automation system.

The software agent 3 generated using the first component in the form of an input/output unit 1 is first of all executed within the input/output unit 1 using the real-time-compatible virtual machine 6, and hence part of the request 4 is handled. Information or data which need to be added to a response to the request 4 are collected by the mobile software agent 3, which migrates to various other networked components of the automation system which are likewise executed using a real-time-compatible virtual machine 6 and, altogether, can handle the omitted parts of the request 4. By way of example, the mobile software agent 3 first of all migrates to a second component, which is a programmable logic controller 5. The programmable logic controller 5 likewise comprises an OPC-XML interface 13 and also a real-time-compatible virtual machine 6. The mobile software agent 3 is executed on the real-time-compatible virtual machine 6 of the programmable logic controller 5 and, in so doing, obtains access to the data of
further field transmitters 17 which are connected to the PROFINET via a further input/output unit 18 in the second subnetwork 8. In this way, the mobile software agent 3 is able to collect data from field transmitters which are located in a subnetwork which is remote from the remote station 9.

In this context, the mobile software agent 3 has the necessary intelligence to look for the data largely independently within the automation system. A precise configuration for the hardware of the automation system therefore does not need to be known at the location of the remote station 9. It is also possible to change the configuration of the automation system, particularly also of the second subnetwork 8, without needing to notify a central component of this in order to allow a user to interact with the automation system or its components from the remote station 9.

Within the second subnetwork 8 there is a further PC 30 on which a web browser 20 and a non-real-time-compatible virtual machine 10 are likewise installed. The further PC 30 can also be used to generate a further request 19 and, by way of example, to send it to the PLC 5. Using the Mobile Agent Server 34 running on the real-time-compatible virtual machine 6 of the PLC 5, an additional software agent 21 is generated on the basis of the further requests 19. In this case too, it is again the task of the additional software agent 21 to migrate from component to component in the automation system and to handle the underlying further request 21 through execution of the additional software agent 21 on the respective virtual machines 6 of the networked components in the automation system. In this context, it is also conceivable for the software agent 3 and the additional software agent 21 to communicate with one another so as to ensure optimum processing of the respective underlying requests 4, 19. In some situations, the software agents 3, 21 generically also produce further software agents, known as subagents, which are likewise involved in handling the requests 4, 19.
As soon as the software agents 2, 21 or their underlying requests 4, 19 have been handled fully, at least one component of the automation system generates appropriate response data 22, 23 and sends them to the PC 9 or to the further PC 30. In this way, the respective user receives a response to his initiated request 4, 19 sent to the automation system.

FIGURE 3 shows a second embodiment of an automation system with networked components. In this case too, the control programs required for carrying out the automated process have also been distributed over various components of the automation system. There is no classical programmable logic controller as in the case of the automation system known from the prior art. Instead, various input/output units 1, 18 are equipped with a suitable execution unit for executing the control programs, for example. The control tasks can be dynamically distributed on the basis of the available resources of the individual components and their utilization level in the automation system.

A remote station 9 can send a request 4 to the automation system. In the embodiment illustrated here, directly on the remote station 9 there is a real-time-compatible virtual machine 6 which can be used to generate a mobile software agent 3 on the basis of the request 4. This mobile software agent 3, which can also be called an intelligent software agent on the basis of its learning capability and adaptation capability, is then sent through a high-speed data network via a Switch & Security Gateway 12 using an Internet Service Provider 11. If the software agent 3 reaches an input/output unit 1 on which a real-time-compatible virtual machine 6 is likewise installed, for example, then the real-time-compatible virtual machine 6 can be used to execute the mobile intelligent software agent 3 and hence to handle the request 4 at least partly. Should the handling of the request 4 by the real-time-compatible virtual machine 6 of the input/output unit 1 not be sufficient to handle the request 4
fully, the mobile software agent 3 will look for further input/output units 18 connected to the high-speed data network and will enforce execution there by the real-time-compatible virtual machines 6 installed on these units. Once the intelligent mobile software agent 3 or its underlying request 4 has finally been handled fully, appropriate response data are sent to the remote station 9.

The collection of data by the intelligent mobile software agents 3 can be synchronized to a special cycle in the automation system. This is very simple to do, particularly by using real-time-compatible virtual machines 6.

The approach of using intelligent mobile software agents for the networked components of an automation system to interact with a remote station can also be extended to the implementation of control functions within the system. The intelligent mobile software agents can be used to implement communication among the individual networked automation components in order to distribute the computation load for the control tasks dynamically in this manner. The approach described can also be used to implement expert systems in order to optimize the performance of such a distributed automation system.

The approach described, of using mobile software agents for a remote station to communicate with the components of the automation system or for the automation components to communicate with one another, requires a smaller bandwidth than known methods. The network load can be reduced, since the mobile software agents can migrate autonomously from component to component. In addition, relatively simple processors can be used within the system, since the computation load can be distributed better by the approach described. The autonomous and asynchronous way in which the software agents work allows application in an environment with heterogeneous hardware engineering. The system is extremely robust, since failure of a mobile software agent or of a node or of a component of the automation system does not necessarily result in an incorrect
response from the overall system. If a component of the automation system fails, the relevant request is automatically routed to a different component of the system. In addition, it is conceivable for the mobile software agents to provide algorithms for system analysis so that failure of the system can be foreseen as early as possible.

An important property of the intelligent mobile software agents described is the ability to communicate with one another. Such communication is understood to mean communication similar to that of humans, which is based on a word framework and rules and provides a high level of flexibility for presenting information. The ability to communicate allows the agents to exchange knowledge and intentions with one another and to make rational decisions on the basis of the current situation in their environment. The meaning of the words which are used for this type of communication and their individual relationships with one another are specified in ontologies. Using the same ontology ensures that the software agents understand one another.

The communication among the agents also allows the following scenario: each agent is responsible for a specific part of the system and holds the information which is relevant to this. In this context, the software agents ideally have a hierarchical structure. By way of example, a software agent on the highest level is asked by an external agent for the status of the part of the total system which is associated with the agent on the highest level. This high-level software agent asks all its subordinate subagents for the relevant status information, which the subagents monitor. On the basis of the individual status information items, the software agent forms a status variable presenting the overall status of said software agent's subordinate part of the total system and sends this status variable to the external agent.

The collection of data by software agents can be used, by way of example, for quality control, for performance analysis for
the total system, for maintenance management, for documentation purposes, for monitoring the production status of an individual product or else for logistical purposes, for example for supplier management.

Since the software agents described are autonomous and they themselves know how they need to behave, the complexity of the overall system is reduced. The agent approach is suitable for breaking down automation tasks into smaller subtasks. This is an advantage for programmers who have the task of programming appropriate control programs.

The network utilization level can be reduced by the agent concept. Agents can group similar data and requests and send them together. In addition, they can carry out preprocessing of the data before they send them, in order to reduce the total volume of the data.
List of reference symbols

1 Input/output unit
2 Mobile agent server
3 Mobile software agent
4 Request
5 Programmable logic controller
6 Real-time-compatible virtual machine
7 First subnetwork
8 Second subnetwork
9 Remote station
10 Non-real-time-compatible virtual machine
11 Internet Service Provider
12 Switch
13 XML-OPC server
14 Demon
15 Distributed Service Provider
16 Field transmitters
17 Further field transmitters
18 Further input/output units
19 Further request
20 Web browser
21 Further mobile software agent
22, 23 Response data
24 First component
25 First execution means
26 Second component
27 Second execution means
28 Third component
29 Third execution means
Patent claims

1. An automation system having at least one first component (24), which has:
   - generation means for generating at least one mobile software agent (3) on the basis of a request (4) which is sent to the automation system and received by the first component (24), and
   - forwarding means for forwarding the software agent (3) to first execution means (25) which are suitable for at least partly handling the request (4) by executing the software agent (3).

2. The automation system as claimed in claim 1, where the automation system comprises at least one second component (26) which is networked to the first component (24) and which has the first execution means (25).

3. The automation system as claimed in claim 2, where the first component (24) has second execution means (27) for partly handling the request (4) by executing the software agent (3).

4. The automation system as claimed in claim 2 or 3, where the second component (26) has further forwarding means for forwarding the software agent (3) to third execution means (29), the third execution means (29) being suitable for at least partly handling the request (4) by executing the software agent (3) and, in particular, being part of a third component (28) of the automation system, which component is networked to the first and/or second component (24, 26).

5. The automation system as claimed in one of claims 2 to 4, where the second component (26) has further generation means for generating at least one further mobile soft-
ware agent after the software agent (3) has been forwarded to the first execution means (25).

6. The automation system as claimed in one of the preceding claims, where the automation system has more than one networked component (24, 26, 28) with a real-time-compatible processing unit for executing control programs.

7. The automation system as claimed in claim 6, where the automation system has means for dynamically distributing control tasks among the networked components (24, 26, 28) with a real-time-compatible processing unit.

8. The automation system as claimed in one of the preceding claims, where the first component (24) has a man-machine interface which allows the request to be initiated by a user.

9. The automation system as claimed in one of the preceding claims, where the first component (24) can be accessed via the Internet or via an intranet.

10. The automation system as claimed in one of the preceding claims, where the first execution means (25) and/or, if present, the second (27) and/or, if present, the third execution means (29) are in the form of a, in particular real-time-compatible, virtual machine (6).

11. The automation system as claimed in one of the preceding claims, where the software agent (3) contains Java objects.

12. The automation system as claimed in one of the preceding claims,
where the software agent \((3)\) has at least one state variable for indicating the handling state of the request \((4)\).

13. The automation system as claimed in one of the preceding claims, where at least one component \((24, 26, 28)\) of the automation system has means for producing response data \((22, 23)\) for the request following full handling of the request \((4)\).

14. The automation system as claimed in one of the preceding claims, where the automation system has means for synchronizing the at least one software agent \((3)\) to a system cycle in the automation system.

15. The automation system as claimed in one of the preceding claims, where at least one component \((24, 26, 28)\) of the automation system has an interface based on an OPC standard.

16. A method for handling requests within an automation system, having the following method steps:
- at least one mobile software agent \((3)\) is generated using a first component \((24)\) of the automation system on the basis of a request \((4)\) which is sent to the automation system and received by the first component \((24)\),
- the software agent \((3)\) is forwarded to first execution means \((25)\), and
- the request \((4)\) is at least partly handled by executing the software agent \((3)\) using the first execution means \((25)\).

17. The method as claimed in claim 16,
where the software agent (3) is forwarded to a second component (26) which is networked to the first component (24) and which has the first execution means (25).

18. The method as claimed in claim 17, where the software agent (3) is executed using second execution means (27) in the first component (24) for partly handling the request (4).

19. The method as claimed in claim 17 or 18, where the software agent (3) is forwarded to third execution means (29) and is executed using the third execution means (29) for at least partly handling the request (4), the third execution means (29), in particular, being part of a third component (28) of the automation system, which component is networked to the first and/or second component (24, 26).

20. The method as claimed in one of claims 17 to 19, where the second component (26) is used to generate at least one further software agent after the software agent (3) has been forwarded to the first execution means (25).

21. The method as claimed in one of claims 16 to 20, where control programs are executed within the automation system using more than one networked component (24, 26, 28) with a respective real-time-compatible processing unit.

22. The method as claimed in claim 21, where control tasks are distributed dynamically among the networked components (24, 26, 28) with a real-time-compatible processing unit.

23. The method as claimed in one of claims 16 to 22, where the request (4) is initiated by a user using a man-machine interface.
24. The method as claimed in one of claims 16 to 23, where the first component (24) is accessed via the Internet or an intranet.

25. The method as claimed in one of claims 16 to 24, where the first execution means (25) and/or, if present, the second (27) and/or, if present, the third execution means (29) are in the form of a, in particular real-time-compatible, virtual machine (6).

26. The method as claimed in one of claims 16 to 25, where the software agent contains Java objects.

27. The method as claimed in one of claims 16 to 26, where at least one state variable in the software agent (3) is set in order to indicate the handling state of the request (4).

28. The method as claimed in one of claims 16 to 27, where at least one component (24, 26, 28) of the automation system is used to produce response data (22, 23) for the request (4) following full handling of the request (4).

29. The method as claimed in one of claims 16 to 28, where the at least one software agent (3) is synchronized to a system cycle in the automation system.

30. The method as claimed in one of claims 16 to 29, where at least one component (24, 26, 28) of the automation system has an interface based on an OPC standard.
### Document Information

**International application No**
PCT/RU2006/000114

**INTERNATIONAL SEARCH REPORT**

**A. CLASSIFICATION OF SUBJECT MATTER**

INV. G06F9/46

According to International Patent Classification (IPC) or to both national classification and IPC.

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, IBM-TDB, INSPEC, COMPENDEX

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document with indication where appropriate, of the relevant passages</th>
<th>Relevant to claim</th>
</tr>
</thead>
</table>

Further documents are listed in the continuation of Box C

See patent family annex

**Date of the actual completion of the international search**

10 January 2007

**Date of mailing of the international search report**

23/01/2007

**Name and mailing address of the ISA/Office**

European Patent Office, P B 5318 Patentlaat 2 NL- 2280 HV Rijswijk Tel (+31-70) 340-2040, Tx 31 651 epo nl Fax (+31-70) 340-3016

Authorized officer

Lo Turco, Salvatore
<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>WO 99/05593 A (BRITISH TELECOMM [GB]; LEE LYNDON CHI HANG [GB]; NWANA HYACINTH SAMA. () 4 February 1999 (1999-02-04) page 1, line 1 - page 6, line 20</td>
<td>1-30</td>
</tr>
<tr>
<td>Patent document cited in search report</td>
<td>Publication date</td>
<td>Patent family member(s)</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6859931 B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 8548898 A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CA 2296061 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DE 69008632 D1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DE 69008632 T2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP .2001511555 T</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6314555 B1</td>
</tr>
</tbody>
</table>