(12) UK Patent Application (19) GB (11) 2 322 719 (13) A

(43) Date of A Publication 02.09.1998

(21) Application No 9726965.8

(22) Date of Filing 19.12.1997

(30) Priority Data
(31) 08780409 (32) 09.01.1997 (33) US

(51) INT CL^6
G06F 9/44

(52) UK CL (Edition P)
G4A APX
U1S S2202

(56) Documents Cited
WO 94/23360 A1 US 5129083 A

(58) Field of Search
UK CL (Edition P) G4A APX
INT CL^6 G06F 9/30 9/44 9/445 9/46
Online: COMPUTER, INSPEC, WPI

(71) Applicant(s)
Mitel Corporation
(Incorporated in Canada - Ontario)
P O Box 13089, Kanata, Ontario K2K 1X3, Canada

(72) Inventor(s)
Thomas A Gray
Deborah L Pinnard

(74) Agent and/or Address for Service
Brookes & Martin
High Holborn House, 52-54 High Holborn, LONDON, WC1V 6SE, United Kingdom

(54) Abstract Title
Establishing a process using software agent control

(57) A method of establishing a process agent comprises storing a library of first programs (resource agents) representing system resources; storing a second program representing a service agent which includes pointers to associated ones of the first programs represented by the service agent; storing a program for invoking a general process, including the steps of: requesting a usage right for functional services represented by the service agent, sending pointers to system resources required for the general process from the service agent to the program for operating the general process and storing the pointers sent from the service agent in association, 3, with the program for operating the general process; and executing the program for operating the general process using the system resources identified by the stored pointers.

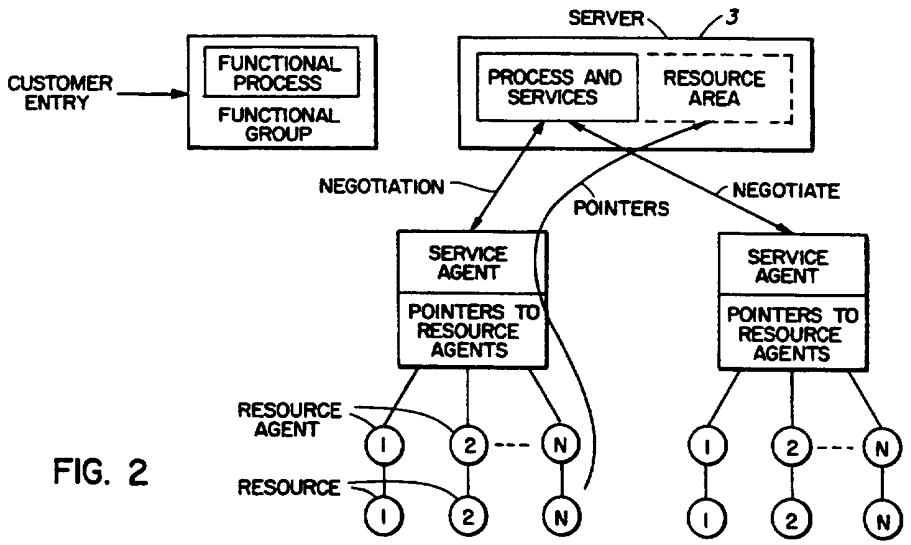

FIG. 2

GB 2 322 719 A

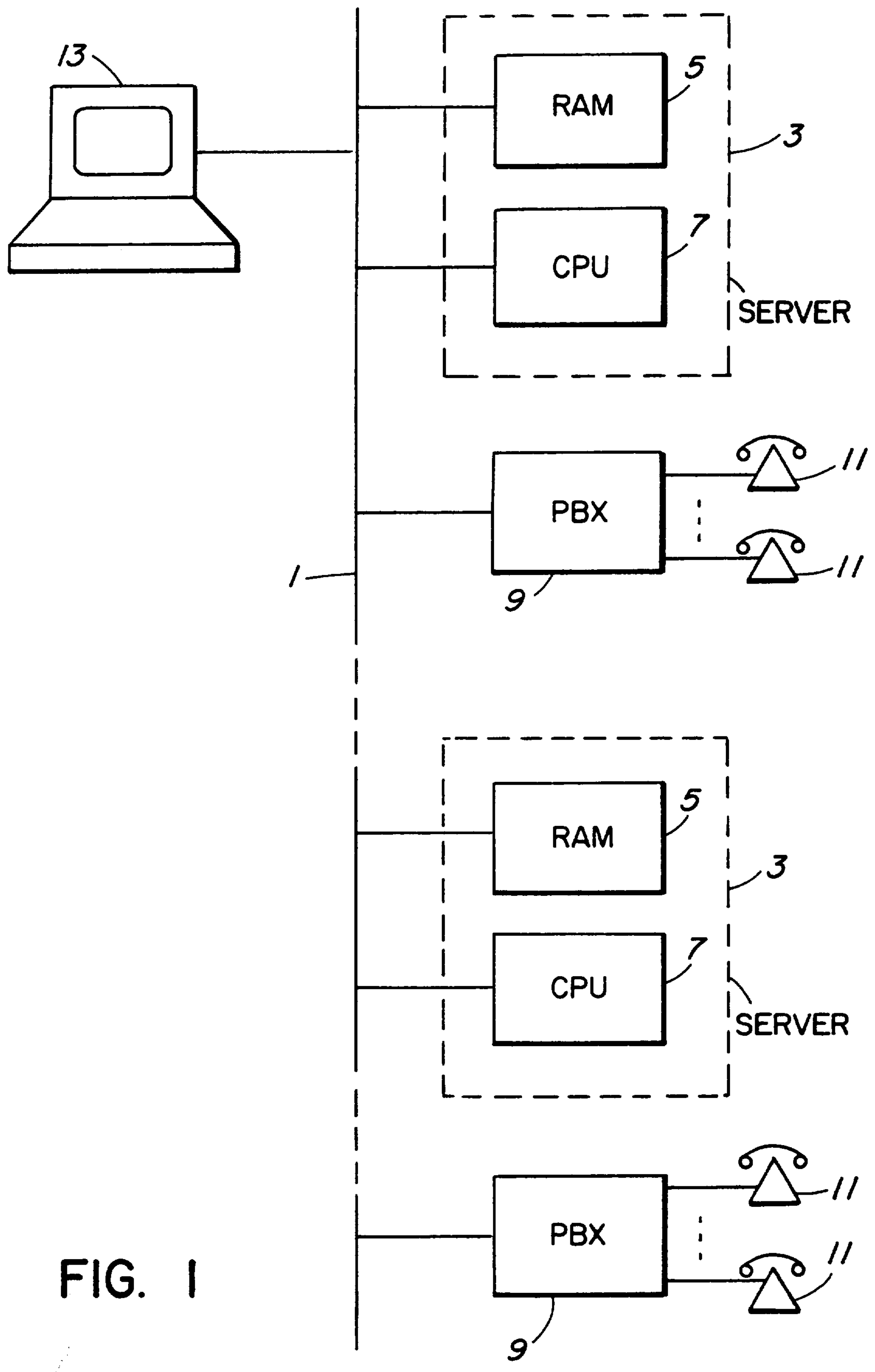

FIG. 1

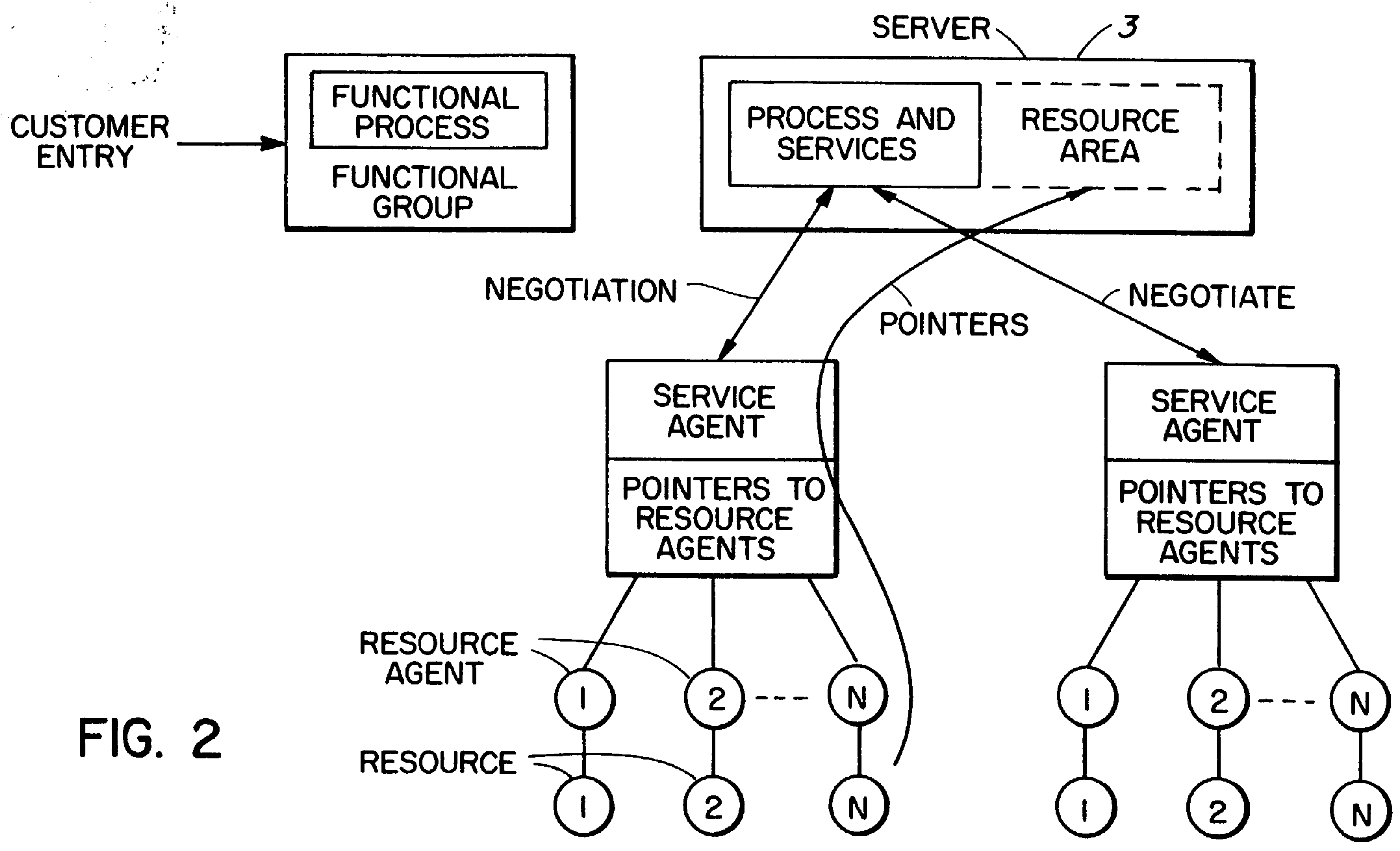
CUSTOMER ENTRY
FUNCTIONAL PROCESS
FUNCTIONAL GROUP
SERVER
3
PROCESS AND SERVICES
RESOURCE AREA
NEGOTIATION
POINTERS
NEGOTIATE
SERVICE AGENT
POINTERS TO RESOURCE AGENTS
SERVICE AGENT
POINTERS TO RESOURCE AGENTS
RESOURCE AGENT
RESOURCE
1
2
N
1
2
N
1
2
N
1
2
N

FIG. 2

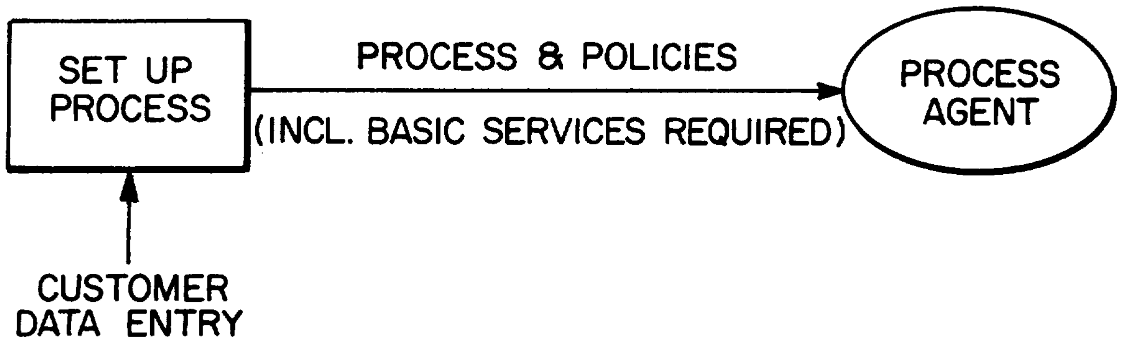
SET UP PROCESS
PROCESS & POLICIES
(INCL. BASIC SERVICES REQUIRED)
PROCESS AGENT
CUSTOMER DATA ENTRY

FIG. 3

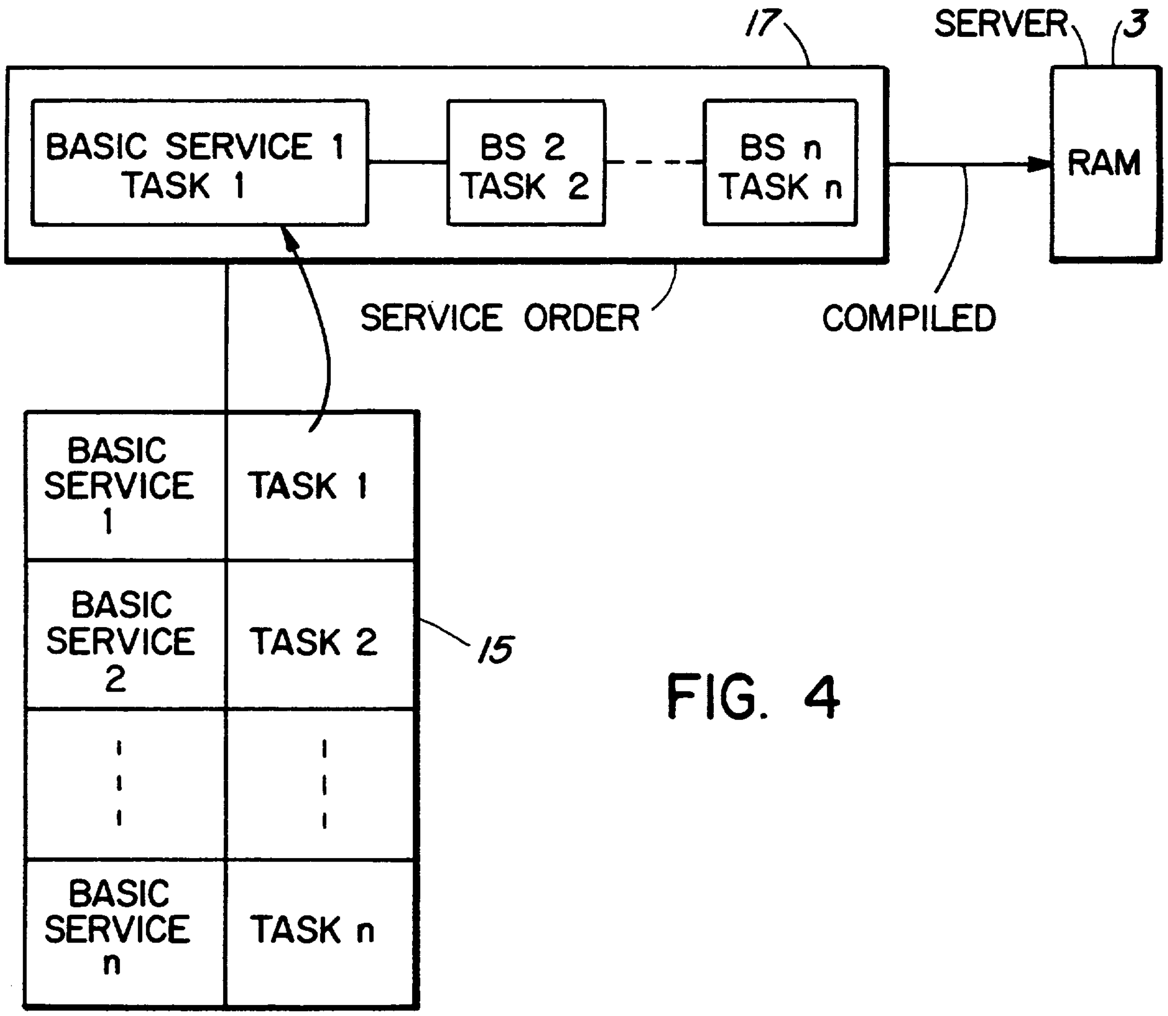
17
SERVER
3
BASIC SERVICE 1 TASK 1
BS 2 TASK 2
BS n TASK n
RAM
SERVICE ORDER
COMPILED
BASIC SERVICE 1
TASK 1
BASIC SERVICE 2
TASK 2
BASIC SERVICE n
TASK n
15

FIG. 4

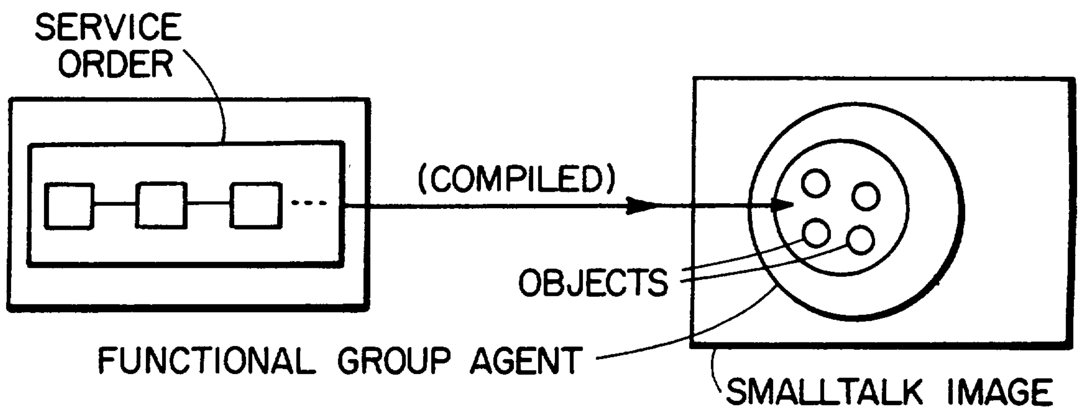

FIG. 5

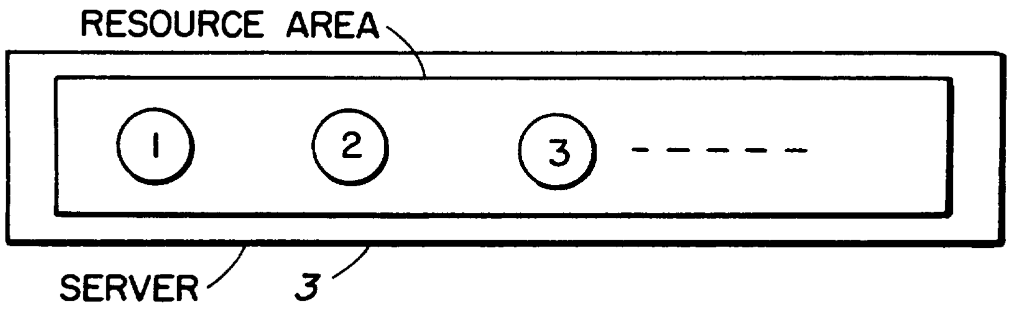

FIG. 6

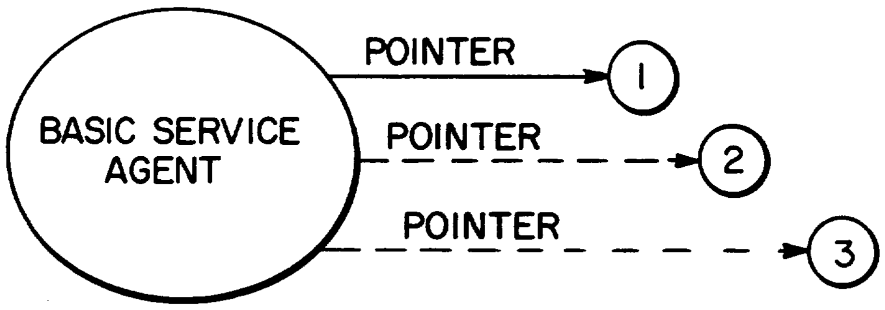

FIG. 7

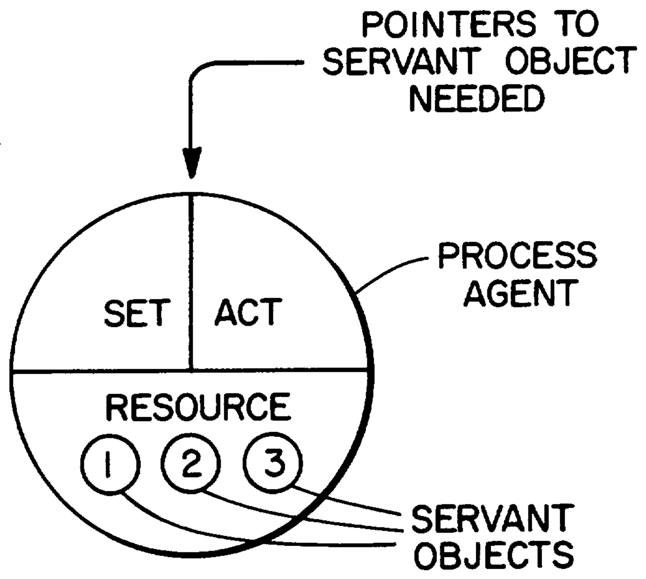
POINTERS TO SERVANT OBJECT NEEDED
SET
ACT
PROCESS AGENT
RESOURCE
1
2
3
SERVANT OBJECTS

FIG. 8

FIELD OF THE INVENTION

This invention relates to methods of controlling communication or other systems and in particular to a method of establishing a process using software agent control.

SUMMARY OF THE INVENTION

The present invention is an improvement and relates to an earlier inventions described in U.S. patent applications 08/257,917 filed June 10, 1994 invented by Deborah Pinard et al and 08/367,821 filed January 3, 1995 invented by Thomas Gray, which is incorporated herein by reference. In an embodiment of the present invention, software objects for operating resources are stored in a random access memory (RAM), in a resource area of a functional process for carrying out the process, which resources are identified during a blackboard bidding process by service agents which have pointers to resource agents controlling the resources.

In accordance with another embodiment of the invention, a method of establishing a process agent is comprised of: (a) storing a library of first software programs representing system resources, (b) storing a second software program representing a corresponding service agent which includes pointers to associated ones of the first software programs which are represented by the service agent, (c) storing a software program for invoking a general process, including the steps of: (i)requesting a usage right for functional services represented by the service agent required by the general process, (ii) sending pointers to system resources required for the general process from the service agent to the software program for operating the general process, and (iii) storing the pointers sent from the service agent in association with the software program for operating the general process, and (d) executing the software program for operating the general process using the system resources identified by the stored pointers.

BRIEF INTRODUCTION TO THE DRAWINGS

A better understanding of the invention will be obtained by considering the detailed description below, with reference to the following drawings, in which:

Figure 1 is a block diagram of a structure in which the present invention can be implemented,

Figure 2 is a diagram illustrating the relationships of various software structures used in the invention,

Figure 3 illustrates process relationships used in a portion of the process,

Figure 4 illustrates the process relationships of Figure 3 in more detail,

Figure 5 illustrates the process relationships of Figure 3 in a different form,

Figure 6 illustrates a portion of the contents of the process agent of Figure 3,

Figure 7 illustrates other process relationships used in another portion of the process, and

Figure 8 illustrates a process agent of Figure 6 in a different form.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The aforenoted patent application describes the nature of software agent processing in a communication system. The present invention is a preferred method by which a process agent can avail itself of various resources without having knowledge of their specific characteristics.

Figure 1 illustrates a network 1, which need not be a local area network, and could be several networks which communicate with each other by various well-known structures and techniques. At least one server 3 is connected to the network 1; several can be distributed and connected to the network 1 at locations which are established by e.g. traffic patterns, availability of physical locations, etc. Each server is comprised of at least a random access memory (RAM) 5 and at least a central processor (CPU) 7. The central processors 7 communicate via the network 1 or via a local subnetwork, with the associated RAM 5, and can communicate via network 1 with other CPUs 7.

For the sake of illustration, PBXs 9 also are connected to the network 7, and can communicate with each other via asynchronous transfer mode (ATM) cells, via data channels, and/or via separate trunk or data links (not shown). Telephones 11 are connected to the PBXs in a well known manner. However, it should be recognized that instead of, or in addition to, the PBXs, various other systems could be connected to the network 1, such as video phone systems, computers, etc.

The particular equipment connected to the network and how they communicate with each other is not the subject of the present invention, and is believed to be within the skill of a person skilled in the art.

A representative computer terminal 13 is also connected to the network 1, for communication with any of the CPUs.

Turning now also to Figures 2 and 3, a customer enters a process into the system, using terminal 13. This can be comprised of filling in a form on the terminal 13, which is produced by a database program. By filling in the form related to a service, shown on form as "basic service", an associated database program is executed, which relates to a particular type of function or task, shown on the form as "task 1", "task 2", etc.

Upon execution of the database program, a particular service task order for a service is created by the terminal 13, which is related to the particular task requested. For example, the task requested could be the establishment of speech generation. The order would include input and output parameters, such as the monitoring of a voiced input (e.g. by a user) for "yes" or "no", and the synthetically voiced response "you have selected 'yes' (or 'no')". The created service order, formed of a series of tasks is shown as reference 17.

Each of the service tasks (i.e. the output parameters of those service tasks) is saved using an unique name, so that it can be used as an input parameter for other basic service tasks later in the implementation of the service. These names are implemented as global variables.

It is preferred that the configuration and service setup so far described should be implemented in an object oriented computer language, such as Smalltalk. As is well known, a program in object oriented language is comprised of a self-contained routine and associated data, which can be "plugged into" various other routines and operate relatively independently, but which can be controlled by the other routines. Smalltalk is a language which has objects which include methods that can be invoked by sending the object a message.

The tasks in the configuration and setup described above are represented by methods inside of servant objects. The parameters (input and output) that are programmed in the configuration and setup should be translated into parameters for the methods.

As illustrated in Figure 5, once a service has been created, i.e. the parameters are defined (and, it is preferred, an icon representing the service drawn on the terminal display), a script should be compiled which preferably takes the form of smalltalk code. This compiled script should then be loaded into a functional group of tasks 17, which thus describe a complete function for the service, for example, the detection of various sounds received from a user and machine creation of speech responses. These are shown in group 17 as task 1, task 3... task n. The functional group is stored in a memory at the terminal 13.

Once the functional group has been defined, then this process and the policies which govern it (derived from the task object programs selected via the entries to the database) are loaded into the RAM 5 of any server 3 connected to the network 1, as a process agent, as shown in Figures 2 and 3, and as a Smalltalk image, as shown in Figure 5.

The servers also store service agents. Each service agent contains pointers to resource agents. Each resource agent is dedicated to a single resource. A resource can be, for example, a voice synthesizer.

It is preferred that the server RAM should contain a servant object library, which is formed of resource programs (objects, containing circled numbers) whose functions are to interface associated resource agents using the correct identifier for the resource agents with which it interfaces, as shown in Figure 6. Thus each object in the library is able to communicate with one particular resource.

Thus each of the service agents has pointers to all of the objects in the servant object library which it uses to provide the basic service, as shown in figure 7.

When a process is initialized, it must negotiate usage rights for each basic service that is required to fulfill the objective of the process. It does this by establishing a blackboard bidding procedure as described in the aforenoted patent application. The service agents which can fulfill the requirements of the process bid on the request, and one service agent is selected, completing the negotiation. The selected service agent then passes pointers to the resource agents, i.e. the servant objects (i.e. methods), that the process agent needs to fulfill the requirements of the process, to the process agent.

The process agent, having received the pointers, accesses the resource agents pointed to and copies of these resource agents are installed in the RAM in a resource area associated with the process agent, as shown in Figure 8.

The process agent, now having direct access to the process agents required to fulfill the tasks associated with the function, can carry out the process when required by the server.

It should be noted that the resource agents can be invoked from any functional group agent on the network. Thus service agents in RAM 5 of one server can be accessed, and can bid on a blackboard process established in another server, and the resulting resource agents transferred to the function process agent stored in RAM 5 of the other server.

It should be noted that once a process agent has been initialized, and contains all of its servants (resource agents), it then can fulfill its function (set of tasks) for any other process. Thus it adds itself to the object library which represents itself, and adds a pointer to itself to one of the service agents. In this way it makes itself available to other processes as a basic service.

In a similar manner, resource brokers (service agents) can reside in a library in RAM 5, and the process agent can download the code for the brokers it needs, to pick the appropriate resource for a process it is responsible for.

With the resources coded in an object code language, they can be easily and dynamically be added to the agent code. Once added to the agent code, the general process program can be operated upon request from other processing programs which control the operation of the system, such as the functioning of an auto attendant in a PBX, or the connection of a pair of telephones via one or plural PBXs.

As an example, when speech is to be generated, let us assume that the service can be provided by two different hardware devices (resources), each with a different identifier. Two service agents (objects) are written which translate the "generic" language of the instruction "generate speech" to the identities of the two resources. These objects are stored in a common library.

An agent is written which represents the basic service of speech generation, and which has knowledge of the storage location of the two service agents.

When a process is created which requires speech generation (e.g. an auto attendant), the process agent for the auto attendant obtains the pointers to the two service agents from the speech generation basic service agent, and copies the servant code into its own resource area.

Thus if a new hardware or software device which is added can perform speech generation, the servant code for it can be written and added to the library, the speech generation basic service agent is informed of the additional resource, and the next time the resource is required, the new code is automatically available. This facilitates availability of the various resources to processes required by the system in an efficient manner.

A person understanding this invention may now conceive of alternative structures and embodiments or variations of the above. All those which fall within the scope of the claims appended hereto are considered to be part of the present invention.

We claim:

1. A method of establishing a process agent comprising:

(a) storing a library of first software programs representing system resources,

(b) storing a second software program representing a corresponding service agent which includes pointers to associated ones of the first software programs which are represented by the service agent,

(c) storing a software program for invoking a general process, including the steps of:

(i) requesting a usage right for functional services represented by the service agent required by the general process,

(ii) sending pointers to system resources required for the general process from the service agent to the software program for operating the general process, and

(iii) storing said pointers sent from the service agent in association with the software program for operating the general process, and

(d) executing the software program for operating the general process using the system resources identified by the stored pointers.

2. A method as defined in claim 1, including providing plural second software programs representing plural corresponding service agents each including pointers to particular associated ones of said first software programs, and in which step (c)(i) includes requesting a usage right for functional services represented by said plural service agents.

3. A method as defined in claim 2 in which the step of requesting a usage right is performed by posting a service requirement to a blackboard, said second software program bidding on the posted service requirement, and in which step (c)(ii) includes sending pointers from a service agent which succeeded in the bidding.

4. A method as defined in claim 3 in which each of the first software programs is a software object.

5. A method as defined in claim 5 in which the step of storing a software program invoking a general process is comprised of a user indicating a service requirement on a user terminal, establishing a group of tasks to fulfill the service requirement as a service order, compiling the group of tasks in an object oriented language code, defining objects, and loading the objects into a functional group agent in a server in which said software program for invoking a general process is to be stored.

6. A method as claimed in claim 1 substantially as hereinbefore described with reference to the accompanying drawings.

The Patent Office

12

Application No:	GB 9726965.8	**Examiner:**	Geoffrey Western
Claims searched:	1-6	**Date of search:**	25 June 1998

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.P): G4A (APX)

Int Cl (Ed.6): G06F 9/30 9/44 9/445 9/46

Other: Online : COMPUTER, INSPEC, WPI

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
A	WO 94/23360 A1 (TALIGENT)	-
A	US 5129083 A (CUTLER et al)	-

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
		E	Patent document published on or after, but with priority date earlier than, the filing date of this application.
&	Member of the same patent family		