

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
2 July 2009 (02.07.2009)

PCT

(10) International Publication Number
WO 2009/082385 A1

(51) International Patent Classification:
G06Q 50/00 (2006.01)

(21) International Application Number:
PCT/US2007/088334

(22) International Filing Date:
20 December 2007 (20.12.2007)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicants (for all designated States except US):
HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P. [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US). **BELROSE, Guillaume Alexandre** [FR/GB]; Filton Road, Stoke Gifford, Bristol BS34 8QZ (GB).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **EDWARDS, Nigel** [GB/GB]; Bristol Filton B3 (BUK02) 1 M138, Mail Stop

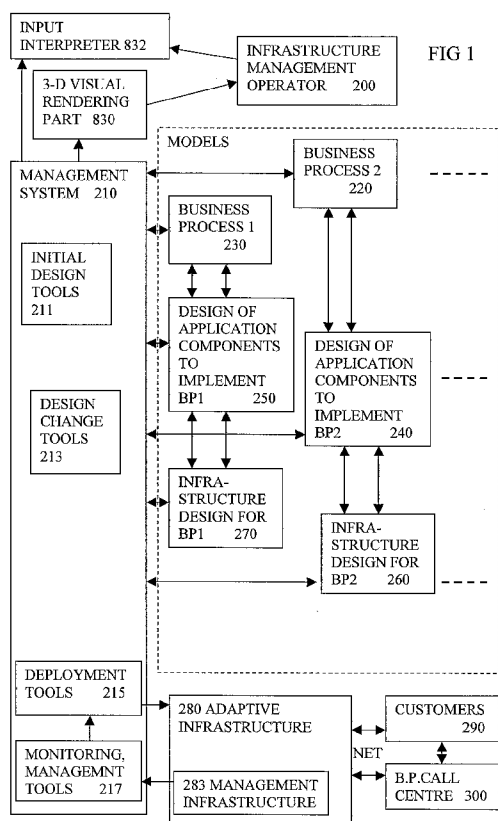
M36, Filton Rd., Stoke Gifford, Bristol BS34 8QZ (GB). **GRAUPNER, Sven** [DE/US]; B-3 1U H8, Mail Stop 1125, 1501 Page Mill Road, Palo Alto, CA 94304-1100 (US). **MANLEY, John** [GB/GB]; Filton Rd., Stoke Gifford, Bristol BS34 8QZ (GB). **ROLIA, Jerome** [CA/CA]; 100 Herzberg Rd., Kanata, Ontario K2K 2A6 (CA). **STEPHENSON, Bryan** [US/US]; B-1/1a 1U H10, Mail Stop 1125, 1501 Page Mill Rd., Palo Alto, CA 94304-1100 (US). **WILCOCK, Lawrence** [GB/GB]; Bristol Filton B1 (BUK01) 1 M25, Mail Stop HPLB, Filton Rd., Stoke Gifford, Bristol BS34 8QZ (GB).

(74) Agent: **HEWLETT-PACKARD COMPANY**; Intellectual Property Administration, Mail Stop 35, P.O. Box 272400, Fort Collins, CO 80527-2400 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW,

[Continued on next page]

(54) Title: VISUAL INTERFACE FOR SYSTEM FOR DEPLOYING COMPUTER BASED PROCESS ON SHARED INFRA-STRUCTURE



(57) Abstract: Deploying a computer based process on shared physical infrastructure (815) of a service provider, involves generating a model (827) of the process, suitable for automated deployment, to meet given non functional requirements, by generating for the model a representation (829) of an arrangement of software application components, for carrying out the functional steps, and a design (834) of computing infrastructure, for running the software application components. A visual interface (832, 834) for operators generates a three dimensional view of at least the computing infrastructure design of the model for operators. An input interpreter translates operator inputs associated with the three dimensional view into changes to corresponding parts of the model. The 3-D visual interface enables managing of more complex systems, or reduce the need for knowledgeable human operators.



MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

- (84) **Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL,

PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

- *as to the identity of the inventor (Rule 4.17(i))*
- *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

- *with international search report*

-1-

VISUAL INTERFACE FOR SYSTEM FOR DEPLOYING COMPUTER BASED
PROCESS ON SHARED INFRASTRUCTURE

Related Applications

5 This application relates to copending US applications of even date titled "MODEL
BASED DEPLOYMENT OF COMPUTER BASED BUSINESS PROCESS ON
DEDICATED HARDWARE" (applicant reference number 200702144), titled
"SETTING UP DEVELOPMENT ENVIRONMENT FOR COMPUTER BASED
10 BUSINESS PROCESS" (applicant reference number 200702145), titled
"MODELLING COMPUTER BASED BUSINESS PROCESS FOR
CUSTOMISATION AND DELIVERY" (applicant reference number 200702363),
titled "MODELLING COMPUTER BASED BUSINESS PROCESS AND
SIMULATING OPERATION" (applicant reference number 200702377), titled
15 "AUTOMATED MODEL GENERATION FOR COMPUTER BASED BUSINESS
PROCESS", (applicant reference number 200702600), and titled
"INCORPORATING DEVELOPMENT TOOLS IN SYSTEM FOR DEPLOYING
COMPUTER BASED PROCESS ON SHARED INFRASTRUCTURE", (applicant
reference number 200702601), and previously filed US application titled
20 "DERIVING GROUNDED MODEL OF BUSINESS PROCESS SUITABLE FOR
AUTOMATIC DEPLOYMENT" (serial number 11/741878) all of which are hereby
incorporated by reference in their entirety.

Field of the Invention

25 The invention relates to methods of modelling a process such as a business process
having a number of computer implemented steps using software application
components, to enable automatic deployment on computing infrastructure, and to
corresponding apparatus and software.

Background

30 Physical IT (information technology) infrastructures are difficult to manage.
Changing the network configuration, adding a new machine or storage device are
typically complicated and error prone manual tasks. In most physical IT
infrastructure, resource utilization is very low: 15% is not an uncommon utilization

-2-

for a server, 5% for a desktop. To address this, modern computer infrastructures are becoming increasingly (re)-configurable and more use is made of shared infrastructure in the form of data centres provided by service providers.

. Hewlett Packard's UDC (Utility Data Centre) is an example which has been applied

5 commercially and allows automatic reconfiguration of physical infrastructure: processing machines such as servers, storage devices such as disks, and networks coupling the parts. Reconfiguration can involve moving or starting software applications, changing allocations of storage space, or changing allocation of processing time to different processes for example. Another way of contributing more
10 reconfigurability, is by allowing many "virtual" computers to be hosted on a single physical machine. The term "virtual" usually means the opposite of real or physical, and is used where there is a level of indirection, or some mediation between the resource user and the physical resource.

In addition future computing fabrics are likely to allow the underlying hardware to be
15 reconfigured. In once instance the fabric might be configured to provide a number of four-way computers. In another instance it might be re-configured to provide four times as many single processor computers.

It is extremely complex to model the full reconfigurability of the above. Models of higher level entities need to be recursive in the sense of containing or referring to
20 lower level entities used or required to implement them (for example a virtual machine VM, may operate faster or slower depending on what underlying infrastructure is currently used to implement it (for example hardware partition nPAR or virtual partition vPAR, as will be described in more detail below). This means a model needs to expose the underlying configurability of the next generation computer
25 fabrics - an nPAR consists of a particular hardware partition. This makes the models so complex that it becomes increasingly difficult for automated tools (and humans) to understand and process the models, to enable design and management of: a) the business process, b) the application and application configuration, and c) the infrastructure and infrastructure configuration.

30 The need to model the full reconfigurability and recursive nature of a system is exemplified in the DMTF's profile for "System Virtualization, Partitioning and Clustering": <http://www.dmtf.org/apps/org/workgroup/redundancy/>

-3-

Another example of difficulties in modelling is WO2004090684 which relates to modeling systems in order to perform processing functions. It says "The potentially large number of components may render the approach impractical. For example, an IT system with all of its hardware components, hosts, switches, routers, desktops, operating systems, applications, business processes, etc. may include millions of objects. It may be difficult to employ any manual or automated method to create a monolithic model of such a large number of components and their relationships. This problem is compounded by the typical dynamic nature of IT systems having frequent adds/moves/changes. Secondly, there is no abstraction or hiding of details, to allow a processing function to focus on the details of a particular set of relevant components while hiding less relevant component details. Thirdly, it may be impractical to perform any processing on the overall system because of the number of components involved."

15 Summary of the Invention

An object is to provide improved apparatus or methods. In one aspect the invention provides:

A method of deploying a computer based process on shared physical infrastructure of a service provider, the process having a number of functional steps, the method having the steps of:

generating a software model of the process, suitable for automated deployment on the shared physical infrastructure, to meet given non functional requirements, by generating for the model a representation of an arrangement of software application components, for carrying out the functional steps, and a design of computing infrastructure, for running the software application components,

generating a three dimensional view of at least the computing infrastructure design of the model for operators,

translating operator inputs associated with the three dimensional view into changes to corresponding parts of the model, and

carrying out the automated deployment of the process based on the model, on the shared physical infrastructure.

-4-

By having a 3D visual interface, operators can manage more complex systems, which become feasible with modelling sufficiently rich to model the computing infrastructure sufficiently for automated deployment.

This helps enable the service provider or enterprise to ensure the resulting process is
5 closer to the real needs of the enterprise, expressed in the non functional requirements and the functional steps, at the initial design or after subsequent revisions through the lifecycle of the process. In turn this improved customisation is likely to lead to more efficient use of resources, in terms of the enterprise paying for just what they need, and in terms of the service provider having more resources to offer to other
10 enterprises for example. Such customisation is traditionally too expensive or too slow for service providers to provide manually. It can become more cost effective and faster to provide by means of the automation of the design of the computing infrastructure for the process. This helps to reduce the need for scarce knowledgeable human operators. Shared infrastructure helps reduce costs and enables more rapid
15 changes in scale as resources can be reallocated.

Embodiments of the invention can have any additional features, without departing from the scope of the claims, and some such additional features are set out in dependent claims and in embodiments described below.

Another aspect provides software on a machine readable medium which when
20 executed carries out the above method.

Another aspect provides a method of using a visual interface to a system for deploying a computer based process on shared physical infrastructure of a service provider, the process having a number of functional steps, the system having:
a model generating part for generating a software model of the process, suitable for
25 automated deployment on the shared physical infrastructure, to meet given non functional requirements, by generating for the model a representation of an arrangement of software application components, for carrying out the functional steps, and a design of computing infrastructure, for running the software application components, the method having the steps of:
30 using an operator terminal to view a three dimensional view of at least the computing infrastructure design of the model,

-5-

causing changes to the model by inputting actions associated with entities in the three dimensional view, for translation into changes to corresponding parts of the model, and

5 causing the system to carry out the automated deployment of the process based on the model, on the shared physical infrastructure.

Another aspect provides a system for deploying a computer based process on shared physical infrastructure of a service provider, the process having a number of functional steps, the system having:

10 a model generator part for generating a software model of the process, suitable for automated deployment on the shared physical infrastructure, to meet given non functional requirements, by generating for the model a representation of an arrangement of software application components, for carrying out the functional steps, and a design of computing infrastructure, for running the software application components, the system also having;

15 a three dimensional visual rendering part coupled to the model and arranged to generate a three dimensional view of at least the computing infrastructure design of the model for operators,

an input interpreter coupled to the model generator to enable operator input to the model generation by translating operator inputs associated with the three dimensional
20 view into changes to corresponding parts of the model, and

a deployment part for carrying out the automated deployment of the process based on the model, on the shared physical infrastructure.

Other aspects can encompass corresponding apparatus or parts of the system such as
25 some of the software for the system and steps by human operators using the system. These aspects are useful to get direct infringement or inducing of direct infringement in cases where the infringers system is partly or largely located remotely and outside the jurisdiction covered by the patent, as is feasible with many such systems, yet the human operator is using the system and gaining the benefit, from within the
30 jurisdiction. Other advantages will be apparent to those skilled in the art, particularly over other prior art. Any of the additional features can be combined together, and combined with any of the aspects, as would be apparent to those skilled in the art. The

embodiments are examples only, the scope is not limited by these examples, and many other examples can be conceived within the scope of the claims.

Brief Description of the Figures

- 5 Specific embodiments of the invention will now be described, by way of example, with reference to the accompanying Figures, in which:
- Figure 1 shows a schematic view of an embodiment showing models, adaptive infrastructure and a management system,
- 10 Figure 2 shows a schematic view of some operation steps by an operator and by the management system, according to an embodiment,
- Figure 3 shows a schematic view of some of the principal actions and models according to an embodiment,
- Figure 4 shows a schematic view of a sequence of steps from business process to deployed model in the form of a model information flow, MIF, according to another
- 15 embodiment,
- Figure 5 shows a sequence of steps and models according to another embodiment,
- Figure 6 shows steps in deriving a grounded model according to an embodiment,
- Figure 7 shows an arrangement of master and slave application servers for a distributed design, according to an embodiment,
- 20 Figure 8 shows parts of a master application server for the embodiment of figure 7,
- Figure 9 shows an arrangement of virtual entities on a server, for use in an embodiment,
- Figure 10 shows an example of a sales and distribution business process (SD) Benchmark Dialog Steps and Transactions,
- 25 Figure 11 shows an example Custom Model Instance for SD Benchmark,
- Figure 12 shows a class diagram for an Unbound Model Class,
- Figure 13 shows an example of a template suitable for a decentralised SD example,
- Figure 14 shows a Grounded Model instance for a decentralized SD,
- Figure 15 shows another example of a template, suitable for a centralised secure SD
- 30 example,
- Figure 16 shows an overview of an embodiment having a 3-D visual interface for the operator according to an embodiment,
- Figure 17 shows some of the principal steps according to an embodiment,

-7-

Figure 18 shows an overview of an architecture of a system according to an embodiment,

Figure 19 shows example of a 3D view of part of a model, and

Figures 20, 21 and 22 show steps of methods according to embodiments .

5

Description of Specific Embodiments

Definitions:

“non-functional requirements” can encompass how well the functional steps are achieved, in terms such as performance, security properties, cost, availability and others. It is explained in Wikipedia (http://en.wikipedia.org/wiki/Non-functional_requirements) for non-functional requirements as follows– “In systems engineering and requirements engineering, non-functional requirements are requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that specify specific behavior or functions. Typical non-functional requirements are reliability, scalability, and cost. Non-functional requirements are often called the ilities of a system. Other terms for non-functional requirements are "constraints", "quality attributes" and "quality of service requirements".”

Functional steps can encompass any type of function of the business process, for any purpose, such as interacting with an operator receiving inputs, retrieving stored data, processing data, passing data or commands to other entities, and so on, typically but not necessarily, expressed in human readable form....

“Deployed” is intended to encompass a modelled business process for which the computing infrastructure has been allocated and configured, and the software application components have been installed and configured ready to become operational. According to the context it can also encompass a business process which has started running.

“suitable for automated deployment” can encompass models which provide machine readable information to enable the infrastructure design to be deployed, and to enable the software application components to be installed and configured by a deployment service, either autonomously or with some human input guided by the deployment service.

-8-

“business process” is intended to encompass any process involving computer implemented steps and optionally other steps such as human input or input from a sensor or monitor for example, for any type of business purpose such as service oriented applications, for sales and distribution, inventory control, control or scheduling of manufacturing processes for example. It can also encompass any other process involving computer implemented steps for non business applications such as educational tools, entertainment applications, scientific applications, any type of information processing including batch processing, grid computing, and so on. One or more business process steps can be combined in sequences, loops, recursions and branches to form a complete Business Process. Business process can also encompass business administration processes such as CRM, sales support, inventory management, budgeting, production scheduling and so on, and any other process for commercial or scientific purposes such as modelling climate, modelling structures, or modelling nuclear reactions.

“application components” is intended to encompass any type of software element such as modules, subroutines, code of any amount usable individually or in combinations to implement the computer implemented steps of the business process. It can be data or code that can be manipulated to deliver a business process step (BPStep) such as a transaction or a database table. The Sales and Distribution (SD) product produced by SAP is made up of a number of transactions each having a number of application components for example.

“unbound model” is intended to encompass software specifying in any way, directly or indirectly, at least the application components to be used for each of the computer implemented steps of the business process, without a complete design of the computing infrastructure, and may optionally be used to calculate infrastructure resource demands of the business process, and may optionally be spread across or contain two or more sub-models. The unbound model can also specify the types or versions of corresponding execution components such as application servers and database servers, needed by each application component, without specifying how many of these are needed for example.

“grounded model” is intended to encompass software specifying in any way, directly or indirectly, at least a complete design of the computing infrastructure suitable for automatic deployment of the business process. It can be a complete specification of a

computing infrastructure and the application components to be deployed on the infrastructure.

“bound model” encompasses any model having a binding of the Grounded Model to physical resources. The binding can be in the form of associations between
5 ComputerSystems, Disks, StorageSystems, Networks, NICS that are in the Grounded Model to real physical parts that are available in the actual computing infrastructure.

“infrastructure design template” is intended to encompass software of any type which determines design choices by indicating in any way at least some parts of the computing infrastructure, and indicating predetermined relationships between the
10 parts. This will leave a limited number of options to be completed, to create a grounded model. These templates can indicate an allowable range of choices or an allowable range of changes for example. They can determine design choices by having instructions for how to create the grounded model, or how to change an existing grounded model.

“computing infrastructure” is intended to encompass any type of resource such as hardware and software for processing, for storage such as disks or chip memory, and for communications such as networking, and including for example servers, operating systems, virtual entities, and management infrastructure such as monitors, for monitoring hardware, software and applications. All of these can be “designed” in the
15 sense of configuring and/or allocating resources such as processing time or processor hardware configuration or operating system configuration or disk space, and instantiating software or links between the various resources for example. The resources may or may not be shared between multiple business processes. The configuring or allocating of resources can also encompass changing existing
20 configurations or allocations of resources. Computing infrastructure can encompass all physical entities or all virtualized entities, or a mixture of virtualized entities, physical entities for hosting the virtualized entities and physical entities for running the software application components without a virtualized layer.

“parts of the computing infrastructure” is intended to encompass parts such as servers,
30 disks, networking hardware and software for example.

“server” can mean a hardware processor for running application software such as services available to external clients, or a software element forming a virtual server

-10-

able to be hosted by a hosting entity such as another server, and ultimately hosted by a hardware processor.

“AIService” is an information service that users consume. It implements a business process.

5 “Application Constraints Model” can mean arbitrary constraints on components in the Customized Process, Application Packaging and Component Performance Models. These constraints can be used by tools to generate additional models as the MIF progresses from left to right.

10 “ApplicationExecutionComponent” is for example a (worker) process, thread or servlet that executes an Application component. An example would be a Dialog Work Process, as provided by SAP.

15 “ApplicationExecutionService” means a service which can manage the execution of ApplicationExecutionComponents such as Work Processes, servlets or data-base processes. An example would be an Application Server as provided by SAP. Such an application server includes the collection of dialog work processes and other processes such as update and enqueue processes as shown in the diagram of the master application server. (figure 8).

20 “Application Packaging Model” is any model which describes the internal structure of the software: what products are needed and what modules are required from the product, and is typically contained by an unbound model.

“Application Performance Model” means any model which has the purpose of defining the resource demands, direct and indirect, for each Business process (BP) step. It can be contained in the unbound model.

25 “Component Performance Model” can mean any model containing the generic performance characteristics for an Application Component. This can be used to derive the Application Performance Model (which can be contained in the unbound model), by using the specific Business process steps and data characteristics specified in the Custom Model together with constraints specified in the Application Constraints Model.

30 “Custom Model” means a customized general model of a business process to reflect specific business requirements.

“Deployed Model” means a bound model with the binding information for the management services running in the system.

-11-

“Candidate Grounded Model” can be an intermediate model that may be generated by a tool as it transforms the Unbound Model into the Grounded Model.

“Grounded Component” can contain the installation and configuration information for both Grounded Execution Components and Grounded Execution Services, as well as
5 information about policies and start/stop dependencies.

“Grounded Execution Component” can be a representation in the Grounded Model of a (worker) process, thread or servlet that executes an Application Component.

“Grounded Execution Service” is a representation in the Grounded Model of the entity that manages the execution of execution components such as Work Processes,
10 servlets or database processes.

“Infrastructure Capability Model” can be a catalogue of resources that can be configured by the utility such as different computer types and devices such as firewalls and load balancers.

MIF (Model Information Flow) is a collection of models used to manage a business
15 process through its entire lifecycle.

The present invention can be applied to many areas, the embodiments described in detail can only cover some of those areas. It can encompass modeling dynamic or static systems, such as enterprise management systems, networked information
20 technology systems, utility computing systems, systems for managing complex systems such as telecommunications networks, cellular networks, electric power grids, biological systems, medical systems, weather forecasting systems, financial analysis systems, search engines, and so on. The details modelled will generally depend on the use or purpose of the model. So a model of a computer system may
25 represent components such as servers, processors, memory, network links, disks, each of which has associated attributes such as processor speed, storage capacity, disk response time and so on. Relationships between components, such as containment, connectivity, and so on can also be represented.

An object-oriented paradigm can be used, in which the system components are
30 modeled using objects, and relationships between components of the system are modeled either as attributes of an object, or objects themselves. Other paradigms can be used, in which the model focuses on what the system does rather than how it operates, or describes how the system operates. A database paradigm may specify

-12-

entities and relationships. Formal languages for system modelling include text based DMTF Common Information Model (CIM), Varilog, NS, C++, C, SQL, or graphically expressed based schemes.

Embodiments of this invention as described make use of a 3D tool/game for the design and management of complex IT systems. The tool can be implemented to be similar in appearance to the Electronic Arts TM computer game called SimCity TM : it allows IT administrators to design IT systems using building blocks and connectors linking those blocks together.

Note: the distinction between tool and game is blurred on purpose, as it is envisioned that embodiments of this invention can look like and be operated like a video game. In the remainder of this document, the embodiments will be referred to as a “game” or “tool”.

As building is happening within the game environment, models of the IT system and the applications running on it are created (or updated if they already exist) and in turn deployed on shared infrastructure in a data centre. Physical machines get provisioned, virtual machines (vm) created and software deployed and configured in accordance to the building plans drawn in the tool. At the same time, the tool can be used to represent the real time state of the system, showing for instance network data packets being transmitted, operating system processes memory growing or shrinking or virtual machines being migrated between physical hosts. The tool can also allow for a 3D representation and control of higher level concepts, such as business logic and business processes running inside the application servers.

Once the IT system is running, the tool supports the execution of management operations. Actions triggered on the in-game building blocks to affect their properties (appearance, shape, location) are translated into management commands that modify the state of the running system. For example, a vm can be represented in the tool as a cube. Stretching the cube to make it grow is translated as growing the amount of memory made available to the vm in the data centre.

To make it easier to learn and use, the tool can map the components of IT systems into familiar architectural terms: buildings (for physical machine), roads (for network links between racks of machines). Fluid entities, such as network data packets or migratable vms can be represented in the tool as 3D objects free to move along certain paths between designated sources and destinations.

-13-

The tool can be made available online, so that IT administrators based in different physical locations can collaborate to build systems or participate to solve issues that affect running systems.

There are two main advantages over the existing management tools. The use of 3D visualization (for objects and the terrain) makes it possible to handle complex systems more easily. This is observed in real-time strategy games that allow users to control large numbers of in-game characters. SimCity game allows users to build cities of great complexity. The use of 3D is a step beyond the in-web-browser table or tree view representation for management of data center resources. The current approach without 3D visualisation is unlikely to scale as the number of resources to manage reaches a higher order of magnitude. This is likely to happen soon with the uptake of Service Oriented Architecture (many more services to manage) or multi-core computing architectures (many more virtual machines per hardware resources). References to three dimensional views are intended to encompass projections of three dimensional bodies having a depth dimension, onto a two dimensional screen such as a conventional LCD or other screen. The view may be isometric or perspective for example, to show the depth. It is not intended to exclude true three dimensional displays such as holographic displays or those made to appear three dimensional using stereoscopic spectacles.

The fact that the tool is online means that two or more people can collaborate in building IT systems and also work together when it comes to diagnosing problems. This tool can also be used for training purposes in the context of data center operations. The game like feel to the interface should not be neglected as it may render somewhat mundane task a little more fun and rewarding. For example, attention of the operator can be drawn to items needing attention or approaching overload by suitable graphical symbols such as rainclouds, or flames, or showing a block with cartoon like signs of shaking or sweating.

Additional features

Some examples of additional features for dependent claims are as follows:
The input interpreter of the system can be arranged to determine if the changes are allowable changes.

-14-

The model generation part can be arranged to use one of a number of predetermined templates of computing infrastructure design, the input interpreter being arranged to use the template to determine if the changes are allowable changes.

The operator interface can be suitable for multiple operators, and generate three dimensional views for the multiple operators simultaneously, and broadcast changes to the model to all operators. This can help enable collaborative input on complex models. For example one operator may make a change to a functional step, and another operator, being an expert in infrastructure, can view the effects of that change on the infrastructure, and respond appropriately.

This use of an infrastructure design template can reduce the number of options to be evaluated and can help reduce the complexity of the task of generating a model which can be deployed and which makes efficient use of resources. This in turn can help enable a more highly automated method, or can enable more complex business processes to be deployed more efficiently, and managed more efficiently. Where the infrastructure includes virtualized entities, the increased flexibility tends to increase the complexity of the task of infrastructure design, and the use of templates becomes even more appropriate and valuable in this case. Furthermore, it enables human operators to be more productive with less knowledge, experience or skills, or enables operators of a given level of skill, knowledge or experience to handle more complex business processes for example.

The computing infrastructure can comprise virtualized entities. This is one way of increasing flexibility of allocation of resources, and thus enabling improved efficiency of use of the resources. It helps to provide more flexibility since resources such as storage or processing can be divided more finely for sharing, and can be reallocated more quickly.

The input interpreter can be arranged to restrict allowable changes according to a class of the operator. This can enable some operators to change only higher level entities or only change entities for which they have appropriate skills, so for example only a database expert may be allowed to change a given database to add a table or change what others are authorised to do.

The model can represent a structure and behaviour of the deployment sufficiently to enable simulation of the deployment to verify how well the non functional requirements are met.

-15-

The operator input can comprise a change to any one or more of functional steps, software application components, computing infrastructure, development or monitoring tools for the deployment.

5 The model generator can be arranged to determine consequential changes to the rest of model from the operator input. This is useful to enable the model to be maintained consistently and efficiently, and reduce the level of skill and knowledge needed by the operator.

The operator input associated with the view can comprise changing any one or more of shape, appearance, and location of entities in the view.

10 The system can have monitors to monitor behaviour of the deployment, the visual rendering part being coupled to the monitor to present a visual representation of the monitored behaviour in the view. This helps enable an operator to see the effects of changes to the model or changes in usage of the process for example, to enable more efficient management of the deployment.

15 The visual rendering part can have a stored mapping of modelled entities to rendered items, and the input interpreter have a stored mapping of operator actions on rendered items to changes to the model.

The non functional requirements can comprise any one or more of: a number of concurrent users, an indication of desired response time, an availability level, a security level. These are typically significant to facilitate automated deployment.

20 Where the enterprise desires to deploy on dedicated hardware local to the enterprise, yet have the benefits of management by a service provider, then this can add another layer of complexity for operators. Reference is made to above referenced copending application number 200702144 for more details of examples of this. In these circumstances, assisting operators by providing a 3-D visual interface to enable them to navigate complex models more quickly, can be even more valuable.

25 Setting up of a development environment can be facilitated by providing a predetermined mapping of which tools are appropriate for a given development purpose and given part of the model, or by including models of tools to be deployed with the model. Reference is made to above referenced copending application numbers 200702145, and 200702601 for more details of examples of this. Combining this with assisting developers by providing the 3D interface can enable the advantages of both to be enhanced.

-16-

An enterprise interface can be provided to enable the enterprise to customise the non functional requirements independently of each other. Reference is made to above referenced copending application number 200702363 for more details of examples of this. Combining this with providing the 3D interface can assist the enterprise and the service provider to navigate complex models more quickly and see the effects of each others actions for example.

The operation of the business process can be simulated and multiple test deployments can be made in parallel. Reference is made to above referenced copending application number 200702377 for more details of examples of these. Combining this with the 3D visual interface can assist developers.

Annotations can be inserted in the source code to assist in modelling or in documentation. Reference is made to above referenced copending application number 200702600 for more details of examples of this. Combining this with the 3D visual interface can enable the advantages of both to be enhanced.

In the embodiments described, a general aim of the model based approach described is to enable matched changes to all three parts: the functional steps of the process, the applications used to implement the functional steps of the process, and configuration of the computing infrastructure used by the applications. Such changes are to be carried out automatically by use of appropriate software tools interacting with software models modelling the above mentioned parts. Until now there has not been any attempt to link together tools that integrate business process, application and infrastructure management through the entire system lifecycle.

Model based approach

A model-based approach for management of such complex computer based processes will be introduced, before discussing the visual interface. Such models can have structured data models in CIM/UML to model the following:

- Infrastructure elements, such as physical machines, vms, network links.
- Application elements, such as Databases, application servers.
- Business level elements, such as functional steps of business processes running in the application servers.

-17-

A model is an organized collection of elements. Elements are modelled in UML. A goal of some embodiments is to use these data models for the automated on-demand provision of enterprise applications following a Software as a service (SaaS) paradigm. Model-Based technologies to automatically design and manage Enterprise Systems –see “Adaptive Infrastructure meets Adaptive Applications”, by Brand et al, published as an external HP Labs Tech Report:

<http://www.hpl.hp.com/techreports/2007/HPL-2007-138.html>

and incorporated herein by reference, can provide the capability to automatically design, deploy, modify, monitor, and manage a running system to implement a business process, while minimizing the requirement for human involvement.

Problem statement

The design of the hardware infrastructure and software landscape for large business processes such as enterprise applications is an extremely complex task, requiring human experts to design the software and hardware landscape. Once the enterprise application has been deployed, there is an ongoing requirement to modify the hardware and software landscape in response to changing workloads and requirements. This manual design task is costly, time-consuming, error-prone, and unresponsive to fast-changing workloads, functional requirements, and non-functional requirements. The embodiments describe mechanisms to automatically create an optimised design for an enterprise application, monitor the running deployed system, and dynamically modify the design to best meet the non-functional requirements.

There are two basic inputs to the design process:

- Specification of functional requirements. Typically, this is in the form of a set of business steps that the application is to support. These describe what the system is intended to do from the perspective of end users. The specification will specify the set of standard business steps required from a standard catalogue, and any system-specific customisations of these steps. This specification will determine the set of products and optional components that must be included in the design of a suitable software landscape for the enterprise application.

- Specification of non-functional requirements. This defines the requirements that the design must meet, such as performance, security, reliability, cost, and

-18-

maintainability. Examples of performance could include the total and concurrent number of users to be supported, transaction throughput, or response times.

The design process involves the creation of a specification of the hardware and software landscape of the enterprise application that will meet the functional and non-functional requirements described above. This can consist of:

- A set of physical hardware resources, selected from an available pool. The infrastructure would consist of computers, memory, disks, networks, storage, and other appliances such as firewalls.

- A virtual infrastructure to be deployed onto the physical resources, together with an assigned mapping of virtual infrastructure to physical infrastructure. The virtual infrastructure must be configured in such a way to best take advantage of the physical infrastructure and support the requirements of the software running on it. For example, the amount of virtual memory or priority assigned to a virtual machine.

- A selection of appropriately configured software components and services, distributed across the virtual and physical infrastructure. The software must be configured to meet the system specific functional requirements, such as customisations of standard business processes. Additionally, the software must be configured to best make use of the infrastructure it is deployed on, while meeting both the functional and non-functional requirements. Configuration parameters could include the level of threading in a database, the set of internal processes started in an application server, or the amount of memory reserved for use by various internal operations of an application server.

A design for the enterprise application consists of:

- Selection of appropriate quantities and types of physical and virtual infrastructure and software components

- Configuration parameters for the infrastructure and software components and services.

The embodiments described below are concerned with an automated mechanism to create an optimised design for an enterprise application by modelling the enterprise application in order to simulate the effect of various design parameters, such that the most appropriate selections and configurations can be made. A model manager in the form of a Model-Based Design Service (MBDS) is responsible for the creation of a set of models of the system, each with slightly different parameters for selection,

configuration, and evaluation possibilities. The design process can be simply regarded as a search for and selection of the best model, usually in terms of finding the least expensive model which meets the functional and non-functional requirements of the system.

5 Figs 16,17

Figure 16 shows an overview of an embodiment of the invention. This shows a model store 820, shared physical infrastructure 815 of the service provider, and various other parts involved in managing the models and the deployment of the models on the physical infrastructure. These parts for managing would typically be implemented as software services. A deployment part 837 takes the model and deploys it on the physical infrastructure. A model generation part 844 generates the model 827 of the computer based process. A visual interface for operators is provided by a 3-D visual rendering part 830, coupled to the model store, and an input interpreter 832. This is arranged to translate management actions input by the operator associated with rendered entities in the view, into changes to corresponding entities in the model.

The model generation part receives non functional requirements 824 for the process, and functional steps 826 of the process. The model of the process comprises a model of software applications 829, for implementing the functional steps, and a model of computing infrastructure 834 for running the software applications. The model is shown in deployed form as deployed software applications 818 and deployed computing infrastructure 813 in the shared physical infrastructure.

Figure 17 shows some of the principal steps carried out by the system of figure 1 according to an embodiment. Functional steps and non functional requirements (NFRs) for the deployment of the process are received at step 850. The model generation part models the arrangement of software application components to carry out functional steps at step 857. Then the same part models a design of computing infrastructure for running the software application components to meet the NFRs at step 859. A 3-dimensional view of the model is rendered for the operator at step 861. At step 862 operator input of actions associated with entities in the 3-dimensional view of the model into changes to corresponding entities in the model at step 862. The updated model of the process is deployed on shared infrastructure at step 863.

-20-

More details of an example of using a series of models for such purposes will now be described. If starting from scratch, a business process is designed using a business process modeling tool. The business process is selected from a catalog of available business processes and is customized by the business process modeling tool. An
5 available business process is one that can be built and run. There will be corresponding templates for these as described below. Then non-functional characteristics such as reliability and performance requirements are specified.

Next the software entities such as products and components required to implement the business process are selected. This is done typically by searching through a catalog of
10 product models in which the model for each product specifies what business process is implemented. This model is provided by an application expert or the product vendor.

Next the computing infrastructure such as virtual machines, operating systems, and underlying hardware, is designed. This can use templates as described in more detail
15 below, and in above referenced previously filed application serial number 11/741878 "Using templates in automated model-based system design" incorporated herein by reference. A template is a model that has parameters and options: by filling in the parameters and selecting options a design tool transforms the template into a complete model of a deployable system. This application shows a method of modelling a
20 business process having a number of computer implemented steps using software application components, to enable automatic deployment on a computing infrastructure, the method having the steps of:

automatically deriving a grounded model of the business process from an unbound model of the business process, the unbound model specifying the application
25 components to be used for each of the computer implemented steps of the business process, without a complete design of the computing infrastructure, and the grounded model specifying a complete design of the computing infrastructure suitable for automatic deployment of the business process,

the deriving of the grounded model having the steps of providing an infrastructure
30 design template having predetermined parts of the computing infrastructure, predetermined relationships between the parts, and having a limited number of options to be completed, generating a candidate grounded model by generating a completed candidate infrastructure design based on the infrastructure design template,

-21-

and generating a candidate configuration of the software application components used by the unbound model, and evaluating the candidate grounded model, to determine if it can be used as the grounded model.

Next the physical resources from the shared resource pool in the data center are
5 identified and allocated. Finally the physical resources are configured and deployed and ongoing management of the system can be carried out.

All of this can use SAP R/3 as an example, but is also applicable to other SAP systems or non-SAP systems. Templates as discussed below can include not only the components needed to implement the business process and the management
10 components required to manage that business process, but also designs for computing infrastructure.

The model generation part can be implemented in various ways. One way is based on a six stage model flow called the Model Information Flow (MIF). This involves the model being developed in stages or phases which capture the lifecycle of the process
15 from business requirements all the way to a complete running system. The six phases are shown in figure 4 described below in more detail and each has a corresponding type of model which can be summarised as follows:

- General Model: The starting point, for example a high level description of business steps based on the “out-of-the-box” functionalities of software
20 packages the user can choose from.
- Custom Process Model: defined above, and for example a specialization of the previous model (General Model) with choices made by the enterprise. This model captures non-functional requirements such as response time, throughput and levels of security. Additionally, it can specify modifications to the generic
25 business processes for the enterprise.
- Unbound Model: defined above, and for example an abstract logical description of a system capable of running the business process with the requirements as specified by the enterprise.
- Grounded Model: defined above and for example can be a transformation of
30 the previous model (Unbound Model) to specify infrastructure choices, such as the quantities and types of hardware and virtualization techniques to use, and also the structure and configuration of the software to run the business process.

-22-

- Bound Model: a grounded model for which resources in the data centre have been reserved.
- Deployed Model: a grounded model where the infrastructure and the software components have been deployed and configured. At this point, the service is up and running.

Each stage of the flow has corresponding types of model which are stored in a Model Repository.

10 Figure 18 Conceptual Architecture

Figure 18 shows a view of an architecture for a system according to an embodiment of the invention. Physical infrastructure is shown in the form of data centre resources 902, which can be physical and virtual machines for processing, virtual and physical network elements such as routers, switches, links, and virtual and physical storage elements for example. A model store in the form of a model repository 911 is shown, for storing models of any of the types mentioned above. Examples of models 913 are shown, together with a stored design template for the computing infrastructure 917, which serves to limit changes or other management operations on the models, as will be described below. An example of an implementation of the models, and services is described in more detail below, with reference to figures 1 to 15.

Core services consume the models provided by the Model Repository and execute management actions to realize the transitions between phases, to generate the next model in the MIF. Those services can be for example :

- Template-based Design Service (TDS) (not shown and an example of a model based design service): translates non-functional requirements into design choices for a Grounded Model based on the template 917.
- Resource Acquisition Service (RAS) 904: its purpose is to allocate physical resources prior to the deployment of virtual resources, such as vms.
- Resource Configuration Service (RCS) 906: its role is to create/update the virtual and physical infrastructure.
- Software Deployment Service (SDS) 908 : installs and configures the applications needed to run the business processes and potentially other software.

-23-

- Monitoring Services (MS) 909 deploys Probes to monitor behaviour of a Deployed Model. This can include monitoring at any one or more of these three levels:

- Infrastructure: e.g. to monitor CPU, RAM, network I/O usage regardless of which application or functional step is executing.
- Application: e.g. to monitor time taken or CPU consumption of a given application such as a DB process on the operating system, regardless of which particular infrastructure component is used.
- Business process: e.g. count the number of sales order per hour, regardless of which infrastructure components or applications are used.

Figure 18 also shows a Game Server 921 connected to the Model Repository and Game Clients A, 933, and B, 937 connected to the Game Server. These game clients can be local terminals, or remote clients with a connection via the Internet for example, using well established techniques. Each remotely connected client runs a networking layer to communicate with the server to retrieve and send game updates. Each client uses a 3D engine to render 3D objects and terrain on the screen. There exist widely available implementations of 3D engines, such as :

- OpenSceneGraph: which is shown in more detail at:

<http://www.openscenegraph.com/>.

- Java3D: as shown in more detail at:

<https://java3d.dev.java.net/>.

The input translator and the 3-D visual rendering part can be implemented in the form of software run by the Game Server and the game clients in some cases. The game server can be arranged to do part of the task of generating a 3D view. It can convert the modelled entities into a description of locations and types of blocks for the view, for use by the 3D rendering engine. The Game server can be arranged to use four types of models. The Building Blocks Model 931 is a pre-defined set of 3D objects that can be rendered on screen by the Game Clients. Used together, the building blocks can form complex structures. A Connector is a special type of 3D object whose purpose is to connect objects together. This can be rendered as a line, a dotted line, a tube, a roadway, a river channel or similar. Building blocks can be similar to Lego

-24-

blocks: cubes, spheres, cones, etc... An example of a screenview of an environment for altering an entity using a palette of shapes is shown in a screenshot from the SimCity building architect tool which can be seen at:

<http://en.wikipedia.org/wiki/Image:Sc3kBuildingArchitectPlus.png>

5 The Actions Model 929 captures the set of actions that can be executed on the building blocks. An action can be the modification of the object's attributes such as its shape (stretching, shrinking), its location within a 3D space, or its connection to other objects. Actions have a visual appearance in the form of icons. The actions model can limit the actions to allowed actions. For example, the design template can have an
10 entity such as a virtual machine and an attached set of allowable operations to the virtual memory to change its size within a range from 1GB to 4GB, in increments of 250MB. In this case, the actions model can be arranged to limit modifications to such an entity to be consistent with the limits in the design template.

The Model to Building Block Mapping 923 defines how modelled elements (and their
15 relationship) in the Model Repository (such as vm, physical machines, etc...) are rendered as 3D objects on the screen. An example of this mapping could indicate that a physical machine is rendered as a cuboid (as shown for example at <http://en.wikipedia.org/wiki/Cuboid>) whose volume is proportional to the disk space of the physical machine. Another example of a mapping could indicate that the link
20 between a virtual machine and its physical host is represented by a 3D connector of a certain width. This mapping can be arranged to render only allowable attributes, or allowable changes to such attributes.

The Management Operations to Actions Mapping 927 specifies how allowable management operations –at least some of which are defined in the templates– translate
25 into actions affecting the 3D objects (and vice versa).

Figure 19, examples of 3D blocks for 3D view

An example of a management operation is the dynamic modification of the amount of memory made available to a virtual machine. Figure 19 shows a number of blocks and a connector. It shows a 3D representation of a VM labelled xy as a container shaped
30 like a cuboid . The cuboid contains within it others objects, including a cube RAM whose dimension is proportional to the vm memory and a cube labelled CPU% whose dimension is proportional to the VM CPU share on the physical host.

-25-

An action in the Actions Model allows stretching or shrinking 3D cubes. The meaning of the mapping will be the following: shrinking or stretching the cube is **(A)** a legal operation and **(B)** translates to dynamically changing the VM memory.

Figure 19 also shows an example of a disk zx shown as half full by darker colour or shading at the bottom in this example. It is shown connected to the RAM by a network link labelled net yz. The usage of the network link could be shown visually in some way to represent whether it is a bottleneck, or for what percentage of the time it is overloaded, or under utilised, and warrants changing.

10 Figures 20, 21 game server operations

At start-up, the Game Server connects to the Model Repository to read the set of models and elements. The Game Server derives the set of 3D building blocks that can be used using the Model to Building Blocks Mapping. The building blocks are displayed on a palette toolbar for example by all the clients connected to the Game Server. Similarly the set of possible actions is derived from the set of management operations -stored in the template- combined with the Management Operations to Action Mappings. The set of allowed actions can be displayed on a palette toolbar by each connected client.

To design a model using a Game Client, the user drag-and-drops from the palette 3D building blocks and connectors to link objects together (if allowed) as shown by step 702. These operator actions on rendered entities, in the form of in-game actions, are translated by the Game Server into changes to the model, also called requests for changes (RFCs) at step 703 for operations such as:

- Creation of new elements in the Model Repository.
- Or modification of existing elements -or their relationships- if already existing.
- Or execution of management operations on selected elements.

The game server can be arranged to interrogate the model and use the core services to determine consequential changes on other parts or other layers of the model, at step 704 to supplement the change. Another option is to generate a new view showing the consequential changes, and generate and evaluate a simulation of the operation of the changed model before implementing the change in the data centre. Once the operator is satisfied, the completed change or changes are handed over to the Core Services to

-26-

modify the data center resources accordingly at step 707. Once data center resources have been modified, their status is reflected back in the models in the repository. The Game Server in turns translates the new states of the model elements into new states of the 3D objects (different shape, new position, etc....). Monitoring tools, also called probes, in the deployed process in the data centre infrastructure then pass details of behaviour of the process to the game server, as shown in figure 19 and at step 709 of figure 19. This behaviour can be rendered in the 3D view, for example by showing graphical indications of CPU usage, disk usage, database access delay times, average time taken by an application, number of users served, and so on. As in online games, the modified statuses of the 3D objects are then broadcasted to all connected clients so all users get a consistent view on the system at the same time

To visualize an existing model, the Game Server connects to the Model Repository, fetches the model and translates its structural elements and their relationships into 3D building blocks and appropriate connectors as defined in the Model to Building Block mappings. The set of building blocks is passed onto the Game Client for rendering.

An example of actions by the game server generates the 3D view is shown in figure 21. At step 732, a list of the modelled entities in the part of the model in the required view is obtained from the model, and the building block mapping is used to identify which types of blocks and connectors correspond to the listed entities. At step 734, a description of how to render each type of block or connector is obtained using the building blocks model. At step 736, any monitored behaviours to be rendered are received from the tools linked to the deployed process, and interpreted by the game server, which determines which blocks they apply to and where and how to show them. At step 737, for each client, the game server determines locations of items in the view, depending on e.g. zoom, perspective, viewing location, orientation, level of detail, which layers, and so on.

Figure 22 shows an example of actions by the game server to handle operator actions. At step 739, operator inputs such as screen coordinates of a cursor and mouse click inputs are used to detect which rendered building block an operator is manipulating. Then the building block to model mapping is used to identify which modelled entity is being operated on. At step 744, from a determination that an operator is trying to change the shape, for example by a click and drag operation, the actions model is used to display allowed values for example, either graphically or using text in a menu or

-27-

palette, and to restrict the alteration to allowed values. These allowed values can be determined from the design infrastructure template, and the actions model used to obtain instructions on how to display the allowed values. At step 746, the actions to management operations mapping is used to determine the actual change selected by the operator. At step 747, a change management service is used to determine consequential changes to other layers, or other entities and to check all are within change management policies and within the changes allowed by the template, before the changes are deployed in the physical infrastructure.

There exist tools for creating online collaboration environments, such the open source project OpenCroquet: (http://www.opencroquet.org/index.php/Main_Page). Open Croquet can be used for parts of the implementation of the game Server and Clients. Techniques used in SecondLife™ (<http://secondlife.com/>), a 3D online environment where most content is user generated, could also be used. SecondLife has an interface that allows users to create complex structures from building blocks. Blocks are scriptable: their attributes (appearance shapes) can be modified by the execution of easy-to-write, user-friendly code. SecondLife "resident" can use a feature that allows virtual blocks to interact with the Web (and this also means Web Services) via HTTP invocations. This way, linking what happens in SecondLife with the Internet world (and this can include data center management services) is feasible, as shown at <http://mashable.com/2006/05/30/second-life-web-20-virtual-world-mashups/>.

LinCity NG is an Open Source implementation of the Sim City™ game. http://lincity-ng.berlios.de/wiki/index.php/Main_Page. Parts of this could be used for most of the implementation of the Game Clients. Support for a networked mode can be added by modifying the source code.

Templates for the computing infrastructure design

Templates are used to capture designs that are known to instantiate successfully (using the core services mentioned above). An example of template describes a SAP module running on a Linux vm with a certain amount of memory. The templates also capture management operations that it is known can be executed, for instance migration of vm of a certain kind, increasing the memory of a vm, deploying additional application server to respond to high load, etc... If a change management

-28-

service refers to the templates, then the templates can be used to restrict the types of change (deltas) that can be applied to the models.

Templates sometimes have been used in specific tools to restrict choices. Another approach is to use constraints which provide the tool and user more freedom. In this

5 approach constraints or rules are specified that the solution must satisfy. One example might be that there has to be at least one application server and at least one database in the application configuration. These constraints on their own do not reduce the complexity sufficiently for typical business processes, because if there are few constraints, then there are a large number of possible designs (also called a large
10 solution space). If there are a large number of constraints (needed to characterize a solution), then searching and resolving all the constraints is really hard – a huge solution space to explore. Also it will take a long time to find which of the constraints invalidates a given possible design from the large list of constraints.

Templates might also contain instructions for managing change. For example they can
15 contain reconfiguration instructions that need to be issued to the application components to add a new virtual machine with a new slave application server.

The deriving of the grounded model can involve specifying all servers needed for the application components. This is part of the design of the adaptive infrastructure and one of the principal determinants of performance of the deployed business process.

20 The template may limit the number or type of servers, to reduce the number of options, to reduce complexity of finding an optimised solution for example.

The deriving of the grounded model from the unbound model can involve specifying a mapping of each of the application components to a server. This is part of configuring the application components to suit the design of adaptive infrastructure. The template
25 may limit the range of possible mappings, to reduce the number of options, to reduce complexity for example.

The deriving of the grounded model can involve specifying a configuration of management infrastructure for monitoring of the deployed business process in use. This monitoring can be at one or more different levels, such as monitoring the
30 software application components, or the underlying adaptive infrastructure, such as software operating systems, or processing hardware, storage or communications.

More than one grounded model can be derived, each for deployment of the same business process at different times. This can enable more efficient use of resources for

business processes which have time varying demand for those resources for example. Which of the grounded models is deployed at a given time can be switched over any time duration, such as hourly, daily, nightly, weekly, monthly, seasonally and so on. The switching can be at predetermined times, or switching can be arranged according to monitored demand, detected changes in resources such as hardware failures or any other factor.

Where the computing infrastructure has virtualized entities, the deriving of the grounded model can be arranged to specify one or more virtualized entities without indicating how the virtualised entities are hosted. It has now been appreciated that the models and the deriving of them can be simplified by hiding such hosting, since the hosting can involve arbitrary recursion, in the sense of a virtual entity being hosted by another virtual entity, itself hosted by another virtual entity and so on. The template can specify virtual entities, and map application components to such virtual entities, to limit the number of options to be selected, again to reduce complexity. Such templates will be simpler if they do not need to specify the hosting of the virtual entities. The hosting can be defined at some time before deployment, by a separate resource allocation service for example.

The grounded model can be converted to a bound model, by reserving resources in the adaptive infrastructure for deploying the bound model. At this point, the amount of resources needed is known, so it can be more efficient to reserve resources at this time than reserving earlier, though other possibilities can be conceived. If the grounded model is for a change in an existing deployment, the method can have the step of determining differences to the existing deployed model, and reserving only the additional resources needed.

The bound model can be deployed by installing and starting the application components of the bound model. This enables the business process to be used. If the grounded model is for a change in an existing deployment, the differences to the existing deployed model can be determined, and only the additional application components need be installed and started.

Two notable points in the modelling philosophy are the use of templates to present a finite catalogue of resources that can be instantiated, and not exposing the hosting relationship for virtualized resources. Either or both can help reduce the complexity of

-30-

the models and thus enable more efficient processing of the models for deployment or changing after deployment.

Some embodiments can use an infrastructure capability model to present the possible types of resources that can be provided by a computing fabric. An instance of an infrastructure capability model contains one instance for each type of Computer System or Device that can be deployed and configured by the underlying utility computing fabric. Each time the utility deploys and configures one of these types, the configuration will always be the same. For a Computer System this can mean the following for example.

- 10 Same memory, CPU, Operating System
- Same number of NICs with same I/O capacity
- Same number of disks with the same characteristics

The templates can map the application components to computers, while the range of both application components and computers is allowed to vary. In addition the templates can also include some or all of the network design, including for example whether firewalls and subnets separate the computers in the solution. In embodiments described below in more detail, the Application Packaging Model together with the custom (business process) model show how the various application components can implement the business process, and are packaged within the Grounded Model.

- 20 The template selected can also be used to limit changes to the system, such as changes to the business process, changes to the application components, or changes to the infrastructure, or consequential changes from any of these. This can make the ongoing management of the adaptive infrastructure a more tractable computing problem, and therefore allow more automation and thus reduced costs. In some example templates
- 25 certain properties have a range: for example 0 to n, or 2 to n. A change management tool (or wizard, or set of tools or wizards) only allows changes to be made to the system that are consistent with template. The template is used by this change management tool to compute the set of allowable changes, it only permits allowable changes. This can help avoid the above mentioned difficulties in computing
- 30 differences between models of current and next state, if there are no templates to limit the otherwise almost infinite numbers of possible configurations.

Some of the advantages or consequences of these features are as follows:

-31-

1. Simplicity: by using templates it becomes computationally tractable to build a linked tool set to integrate business process, application and infrastructure design and management through the entire lifecycle of design, deployment and change.

2. By limiting the number of possible configurations of the adaptive infrastructure, the particular computing problem of having to compute the differences between earlier and later states of complex models is eased or avoided. This can help enable a management system for the adaptive infrastructure which can determine automatically how to evolve the system from an arbitrary existing state to an arbitrary desired changed state. Instead templates fix the set of allowable changes and are used as configuration for a change management tool.

3. The template models formally relate the business process, application components and infrastructure design. This means that designs, or changes, to any one of these can be made dependent on the others for example, so that designs or changes which are inconsistent with the others are avoided.

Fig 1 overview

Fig 1 shows an overview of infrastructure, applications, and management tools and models according to an embodiment. Adaptive infrastructure 280 is coupled typically over the internet to customers 290, optionally via a business process BP call centre 300. A management system 210 has tools and services for managing design and deployment and ongoing changes to deployed business processes, using a number of models. For example as shown, the management system has initial design tools 211, design change tools 213, deployment tools 215, and monitoring and management tools 217. These may be in the form of software tools running on conventional processing hardware, which may be distributed. Examples of initial design tools and design change tools are shown by the services illustrated in fig 5 described below.

A high level schematic view of some of the models are shown, for two business processes, there can be many more. Typically the management system belongs to a service provider, contracted to provide IT services to enterprises who control their own business processes for their customers. A model 230 of business process 1 is used to develop a design 250 of software application components. This is used to create and infrastructure design 270 for running the application components to implement the business process. This design can then be deployed by the management

-32-

system to run on the actual adaptive infrastructure, where it can be used for example by customers, a call centre and suppliers (not shown for clarity). Similarly, item 220 shows a model of a second business process, used to develop a design 240 of software application components. This is used to create and infrastructure design 260 for running the application components to implement the second business process. This design can then also be deployed by the management system to run on the actual adaptive infrastructure.

The management system has a 3D visual interface to an infrastructure management operator 200. This operator can be service provider staff, or in some cases can be trained staff of the enterprise owning the process. The service provider staff may be able to view and manage the processes of different enterprises deployed on the shared infrastructure. The operators of a given enterprise would be able to view and manage only their own processes. As discussed above, the interface can be regarded as having an input interpreter part 832 and a 3D visual rendering part, both coupled to the management system 210 to be able to interact with the various types of models, and with the infrastructure design template.

The adaptive infrastructure can include management infrastructure 283, for coupling to the monitoring and management tools 217 of the management system. The models need not be held all together in a single repository, in principle they can be stored anywhere.

Fig 2 operation

Figure 2 shows a schematic view of some operation steps by an operator and by the management system, according to an embodiment. Human operator actions are shown in a left hand column, and actions of the management system are shown in the right hand column. At step 500 the human operator designs and inputs a business process (BP). At step 510 the management system creates an unbound model of the BP. At step 520, the operator uses the 3D visual interface to select a template for the design of the computing infrastructure. At step 530, the system uses the selected template to create a grounded model of the BP from the unbound model and the selected template. In principle the selection of the template might be automated or guided by the system. The human operator of the service provider then causes the grounded model to be deployed, either as a live business process with real customers, or as a test

-33-

deployment under controlled or simulated conditions. The suitability of the grounded model can be evaluated before being deployed as a live business process: an example of how to do this is described below with reference to figure 3.

At step 550, the system deploys the grounded model of the BP in the adaptive
5 infrastructure. The deployed BP is monitored by a monitoring means of any type, and monitoring results are passed to the 3D visual interface to be viewed by the human operator. Following review of the monitoring results at step 570, the operator of the enterprise can design changes to the BP or the service provider can design changes to the infrastructure at step 575. These are input to the system, and at step 580 the
10 system decides if changes are allowed by the same template. If no, at step 585, the operator decides either for a new template, involving a return to step 520, or for a redesign within the limitations of the same template, involving at step 587 the system creating a grounded model of the changes, based on the same template.

At step 590 the operator causes deployment of the grounded model for test or live
15 deployment. At step 595 the system deploys the grounded model of the changes. In principle the changes could be derived later, by generating a complete grounded model, and later determining the differences, but this is likely to be more difficult.

Fig 3 operation

20 Fig 3 shows an overview of an embodiment showing some of the steps and models involved in taking a business process to automated deployment. These steps can be carried out by the management system of figure 1, or can be used in other embodiments.

A business process model 15 has a specification of steps 1-N. There can be many
25 loops and conditional branches for example as is well known. It can be a mixture of human and computer implemented steps, the human input being by customers or suppliers or third parties for example. At step 65, application components are specified for each of the computer implemented steps of the business process. At step 75, a complete design of computing infrastructure is specified automatically, based on
30 an unbound model 25. This can involve at step 85 taking an infrastructure design template 35, and selecting options allowed by the template to create a candidate infrastructure design. This can include design of software and hardware parts. At step 95, a candidate configuration of software application components allowed by the

template is created, to fit the candidate infrastructure design. Together these form a candidate grounded model.

At step 105, the candidate grounded model is evaluated. If necessary, further candidate grounded models are created and evaluated. Which of the candidates is a best fit to the requirements of the business process and the available resources is identified. There are many possible ways of evaluating, and many possible criteria, which can be arranged to suit the type of business process. The criteria can be incorporated in the unbound model for example.

There can be several grounded models each for different times or different conditions.

For example, time varying non-functional requirements can lead to different physical resources or even a reconfiguration: a VM might have memory removed out-of-office hours because fewer people will be using it. One might even shutdown an underused slave application server VM. The different grounded models would usually but not necessarily come from the same template with different parameters being applied to generate the different grounded models.

The template, grounded and subsequent models can contain configuration information for management infrastructure and instructions for the management infrastructure, for monitoring the business process when deployed. An example is placing monitors in each newly deployed virtual machine which raise alarms when the CPU utilization rises above a certain level – e.g. 60%.

Fig 4 MIF

Figure 4 shows some of the principal elements of the MIF involved in the transition from a custom model to a deployed instance. For simplicity, it does not show the many cycles and iterations that would be involved in a typical application lifecycle – these can be assumed. The general model 15 of the business process is the starting point and it is assumed that an enterprise or consultant has designed a customized business process. That can be represented in various ways, so a preliminary step in many embodiments is customising it. A custom model 18 is a customization of a general model. So it is likely that a General Model could be modelled using techniques similar to the ones demonstrated for modelling the Custom Model: there would be different business process steps. A custom model differs from the general model in the following respects. It will include non-functional

-35-

requirements such as number of users, response time, security and availability requirements. In addition it can optionally involve rearranging the business process steps: new branches, new loops, new steps, different/replacement steps, steps involving legacy or external systems.

- 5 The custom model is converted to an unbound model 25 with inputs such as application performance 31, application packaging 21, and application constraints 27. The unbound model can specify at least the application components to be used for each of the computer implemented steps of the business process, without a complete design of the computing infrastructure. The unbound model is converted to a
- 10 grounded model 55 with input from models of infrastructure capability 33, and an infrastructure design template 35.

Deployment of the grounded model can involve conversion to a bound model 57, then conversion of the bound model to a deployed model 63. The bound model can have resources reserved, and the deployed model involves the applications being installed

15 and started.

Fig 5 MIF

- Figure 5 shows a sequence of steps and models according to another embodiment. This shows a model repository 310 which can have models such as templates (TMP),
- 20 an unbound model (UM), a bound model (BM), a partially deployed model (PDM), a fully deployed model (FDM). The figure also shows various services such as a service 320 for generating a grounded model from an unbound model using a template. Another service is a resource acquisition service 330 for reserving resources using a resources directory 340, to create a bound model.
- 25 An adaptive infrastructure management service 350 can configure and ignite virtual machines in the adaptive infrastructure 280, according to the bound model, to create a partially deployed model. Finally a software deployment service 360 can be used to take a partially deployed model and install and start application components to start the business process, and create a fully deployed model.

30

Figure 6 deriving grounded model

Figure 6 shows steps in deriving a grounded model according to an embodiment. At step 400, a template is selected from examples such as centralised or decentralised

-36-

arrangements. A centralised arrangement implies all is hosted on a single server or virtual server. Other template choices may be for example high or low security, depending for example on what firewalls or other security features are provided. Other template choices may be for example high or low availability, which can imply redundancy being provided for some or all parts.

At step 410, remaining options in the selected template are filled in. This can involve selecting for example disk sizes, numbers of dialog processes, number of servers, server memory, network bandwidth, server memory, network bandwidth, database time allowed and so on. At step 420, a candidate grounded model is created by the selections. Step 430 involves evaluating the candidate grounded model e.g. by building a queuing network, with resources represented, and with sync points representing processing delays, db delays and so on. Alternatively the evaluation can involve deploying the model in an isolated network with simulated inputs and conditions.

At step 440, the evaluation or simulation results are compared with goals for the unbound model. These can be performance goals such as maximum number of simultaneous users with a given response time, or maximum response time, for a given number of users. At step 450, another candidate grounded model can be created and tested with different options allowed by the template. At step 460 the process is repeated for one or more different templates. At step 470, results are compared to identify which candidate or candidates provides the best fit. More than one grounded model may be selected, if for example the goals or requirements are different at different times for example. In this case, the second or subsequent grounded model can be created in the form of changes to the first grounded model.

Fig 7 Master and Slave application servers

Figure 7 shows an arrangement of master and slave application servers for a decentralised or distributed design of computing infrastructure, according to an embodiment. A master application server 50 is provided coupled by a network to a database 60, and to a number of slave application servers 70. Some of the slaves can be implemented as virtual slave application servers 72. Each slave can have a number of dialog worker processes 80. The master application server is also coupled to remote users using client software 10. These can each have a graphical user interface GUI on

-37-

a desktop PC 20 coupled over the internet for example. The slaves can be used directly by the clients once the clients have logged on using the master.

Fig 8 Master Application Server

- 5 Figure 8 shows parts of a master application server for the embodiment of figure 7. An enqueue process 110 is provided to manage locks on the database. A message server 120 is provided to manage login of users and assignment of users to slave application servers for example. An update server 130 is provided for managing committing work to persistent storage in a database. A print server 140 can be
10 provided if needed. A spool server 150 can be provided to run batch tasks such as reports. At 160 dialog worker processes are shown for running instances of the application components.

Fig 9 Virtual entities

- 15 Figure 9 shows an arrangement of virtual entities on a server, for use in an embodiment. A hierarchy of virtual entities is shown. At an operating system level there are many virtual machines VM. Some are hosted on other VMs. Some are hosted on virtual partitions VPARs 610 representing a reconfigurable partition of a hardware processing entity, for example by time sharing or by parallel processing
20 circuitry. A number of these may be hosted by a hard partitioned entity nPAR 620 representing for example a circuit board mounting a number of the hardware processing entities. Multiple nPARs make up a physical computer 630 which is typically coupled to a network by network interface 650, and coupled to storage such as via a storage area network SAN interface 640.
- 25 There are many commercial storage virtualization products on the market from HP, IBM, EMC and others. These products are focused on managing the storage available to physical machines and increasing the utilization of storage. Virtual machine technology is a known mechanism to run operating system instances on one physical machine independently of other operating system instances. It is known, within a
30 single physical machine, to have two virtual machines connected by a virtual network on this machine. VMware is a known example of virtual machine technology, and can provide isolated environments for different operating system instances running on the same physical machine.

-38-

There are also many levels at which virtualization can occur. For example HP's cellular architecture allows a single physical computer to be divided into a number of hard partitions or nPARs. Each nPAR appears to the operating system and applications as a separate physical machine. Similarly each nPAR can be divided into a number of virtual partitions or vPARs and each vPAR can be divided into a number of virtual machines (e.g. HPVM, Xen, VMware).

Figures 10 to 15

The next part of this document describes in more detail with reference to figs 10 to 15 examples of models that can be used within the Model Information Flow (MIF) shown in figs 1 to 9, particularly fig 4. These models can be used to manage an SAP application or other business process through its entire lifecycle within a utility infrastructure. The diagrams are shown using the well known UML (Unified Modelling Language) that uses a CIM (common information model) style. The implementation can be in Java or other software languages.

A custom model can have a 1-1 correspondence between an instance of an AIService and a BusinessProcess. The AIService is the information service that implements the business process.

A business process can be decomposed into a number of business process steps (BPsteps), so instances of a BusinessProcess class can contain 1 or more BPSteps. An instance of a BPStep may be broken into multiple smaller BPSteps involving sequences, branches and loops for example. Once the BusinessProcess step is decomposed into sufficient detail, each of the lowest level BPSteps can be matched to an ApplicationComponent. An ApplicationComponent is the program or function that implements the BPStep. For SAP, an example would be the SAP transaction named VA01 in the SD (Sales and Distribution package) of SAP R/3 Enterprise. Another example could be a specific Web Service (running in an Application Server).

BPStep can have stepType and stepParams fields to describe not only execution and branching concepts like higher-level sequences of steps, but also the steps themselves. The stepType field is used to define sequential or parallel execution, loops, and if-then-else statements. The stepParams field is used to define associated data. For example, in the case of a loop, the stepParams field can be the loop count or

-39-

a termination criterion. The set of BPSteps essentially describes a graph of steps with various controls such as loops, if-then-else statements, branching probabilities, etc.

The relation BPStepsToApplicationComponentMapping is a complex mapping that details how the BPStep is mapped to the ApplicationComponent. It represents, in a condensed form, a potentially complex mix of invocations on an Application Component by the BPStep, such as the specific dialog steps or functions invoked within the ApplicationComponent or set of method calls on a Web Service, and provided details of parameters, such as the average number of line items in a sales order.

A BPStep may have a set of non-functional requirements (NonFunctionalRequirements) associated with it: performance; availability, security, and others. In the current version availability and security requirements are modelled by a string: "high", "medium", "low". Performance requirements are specified in terms of for example a number of registered users (NoUsersReq), numbers of concurrent users of the system, the response time in seconds and throughput requirement for the number of transactions per second. Many BPSteps may share the same set of non-functional requirements. A time function can be denoted by a string. This specifies when the non-functional requirements apply, so different requirements can apply during office-hours to outside of normal office hours. Richer time varying functions are also possible to capture end of months peaks and the like.

Figs 10, 11 Custom Model

For an example of a Custom Model the well-known Sales and Distribution (SD) Benchmark will be discussed. This is software produced by the well known German company SAP. It is part of the SAP R/3 system, which is a collection of software that performs standard business functions for corporations, such as manufacturing, accounting, financial management, and human resources. The SAP R/3 system is a client server system able to run on virtually any hardware/software platform and able to use many different database management systems. For example it can use an IBM AS/400 server running operating system OS/400 using database system DB2; or a Sun Solaris (a dialect of Unix) using an Oracle database system; or an IBM PC running Windows NT using SQL Server.

-40-

SAP R/3 is designed to allow enterprises to choose their own set of business functions, and to customize to add new database entities or new functionality. The SD Benchmark simulates many concurrent users using the SD (Sales and Distribution) application to assess the performance capabilities of hardware. For each user the interaction consists of 16 separate steps (Dialog Steps) that are repeated over and over. The steps and their mapping to SAP transactions are shown in Figure 10. A transaction here is an example of an Application Component. Each transaction is shown as a number of boxes in a row. A first box in each row represents a user invoking the transaction e.g. by typing /nva01 to start transaction VA01. As shown in figure 10, transaction VA01 in the top row involves the business process steps of invoking the create sales order transaction, then filling order details, then saving the sold-to party, and completing with the "back" function F3 which saves the data.

A next transaction VL01N is shown in the second row, and involves steps as follows to create an outbound delivery. The transaction is invoked, shipping information is filled in, and saved. A next transaction VA03 is shown in the third row for displaying a customer sales order. This involves invoking the transaction, and filling subsequent documents. A fourth transaction is VL02N in the fourth row, for changing an outbound delivery. After invoking this transaction, the next box shows saving the outbound delivery. A next transaction shown in the fifth row is VA05, for listing sales orders. After invoking this transaction, the next box shows prompting the user to fill in dates and then a third box shows listing sales orders for the given dates. Finally, in a sixth row, the transaction VF01 is for creating a billing document, and shows filling a form and saving the filled form.

Figure 11 shows an example of a custom model instance for the SD Benchmark. The top two boxes indicate that the business process "BPModel" contains one top level BPStep: "SD Benchmark", with stepType=Sequence. Two lines are shown leading from this box, one to the non-functional requirements associated with this top-level BPStep, and shown by the boxes at the left hand side. In this particular case only performance requirements have been specified – one for 9am-5pm and the other for 5pm-9am. Other types of non-functional requirements not shown could include security or availability requirements for example. In each case the performance requirements such as number of users, number of concurrent users, response time required, and throughput required, can be specified as shown. These are

-41-

only examples: other requirements can be specified to suit the type of business process. A box representing the respective time function is coupled to each performance requirement box as shown. One indicates 9am to 5pm, and the other indicates 5pm to 9am in this example.

- 5 On the right hand side a line leads from the SD Benchmark BPStep to the functional requirements shown as six BPSteps, with stepType=Step – one for each SAP transaction shown in Figure 10 (VA01, VL01N, etc). For convenience the name of the first dialog step for each transaction shown in Figure 10 is used as the name of the corresponding BPStep shown in Figure 11 (“Create sales order”, “Create outbound
10 delivery”, “Display customer sales order”, “Change outbound delivery”, “List sales order”, and “Create delivery document”). For each of these steps the BPStepToApplicationComponentMapping relation specifies the details of the dialog steps involved. For example in the case of CreateSalesOrder, Figure 10 shows that the BPStepToApplicationComponentMapping needs to specify the following dialog steps
15 are executed in order: “Create Sales Order”, “Fill Order Details”, “Sold to Party” and “Back”. In addition it might specify the number of line items needed for “Fill Order Details”. At the right hand side of the figure, each BP step is coupled to an instance of its corresponding ApplicationComponent via the respective mapping. So BPstep “Create Sales order” is coupled to ApplicationComponent VA01, via mapping having
20 ID:001. BPstep “Create outbound delivery” is coupled to ApplicationComponent VL01N via mapping having ID:002. BPstep “Display customer sales order” is coupled via mapping having ID:003 to ApplicationComponent VA03. BPstep “Change outbound delivery” is coupled via mapping having ID:004 to ApplicationComponent VL02N. BPstep “List sales order” is coupled via mapping
25 having ID:005 to ApplicationComponent VA05. BPstep “Create delivery document” is coupled via mapping having ID:006 to ApplicationComponent VF01.

Fig 12, The Unbound Model

- The Unbound Model is used to calculate resource demands. As shown in
30 Figure 12 this model can be made up of four models: the Custom Model (labelled CustomizedProcessingModel), Application Packaging, Application Constraints and Application Performance models, an example of each of which will be described below (other than the Custom Model, an example of which has been described above

with respect to Figure 11). Other arrangements can be envisaged. No new information is introduced that is not already contained in these four models.

Fig 12, Application packaging model

5 The Application Packaging Model describes the internal structure of the software: what products are needed and what modules are required from the product. An ApplicationComponent can be contained in an ApplicationModule. An ApplicationModule might correspond to a JAR (Java archive) file for an application server, or a table in a database. In the case of SAP it might be the module to be loaded
10 from a specific product into an application server such as SD or FI (Financials). The application packaging model can have a DiskFootPrint to indicate the amount of disk storage required by the ApplicationModule. In the case of the ApplicationComponent VA01 in Figure 10, it is from SD with a DiskFootPrint of 2MB for example.

One or more ApplicationModules are contained within a product. So for example SAP
15 R/3 Enterprise contains SD. ApplicationModules can be dependent on other ApplicationModules. For example the SD Code for the Application Server depends on both the SD Data and the SD Executable code being loaded into the database.

The custom model can have an ApplicationExecutionComponent for executing an ApplicationComponent. This could be a servlet running in an application server or a
20 web server. It could also be a thread of a specific component or a process. In the case of SD's VA01 transaction it is a Dialog Work Process. When it executes, the ApplicationComponent may indirectly use or invoke other Application-Components to run: a servlet may need to access a database process; SD transactions need to access other ApplicationComponents such as the Enqueue Work Process and the Update
25 Work Process, as well as the Database ApplicationExecutionComponent. The ApplicationExecutionComponent can be contained by and executed in the context of an ApplicationExecutionService (SAP application server) which loads or contains ApplicationModules (SD) and manages the execution of ApplicationExecutionComponents (Dialog WP) which, in turn, execute the
30 ApplicationComponent (VA01) to deliver a BPStep.

Fig 12, Application Constraints model

The Application Constraints Model expresses arbitrary constraints on components in the Customized Process, Application Packaging and Component Performance Models. These constraints are used by tools to generate additional models as the MIF progresses from left to right. Examples of constraints include:

- How to scale up an application server – what ApplicationExecutionComponents are replicated and what are not. For example, to scale up an SAP application server to deal with more users one cannot simply replicate the first instance – the master application server 50 of figs 7 and 8, commonly known as the Central Instance. Instead a subset of the components within the Central Instance is needed. This is also an example of design practice, there may be other constraints encoding best design practice.

- Installation and configuration information for ApplicationComponents, ApplicationExecutionComponents and ApplicationExecutionServices

- Performance constraints on ApplicationExecutionServices – e.g. do not run an application server on a machine with greater than 60% CPU utilization

Other examples of constraints include ordering: the database needs to be started before the application server. Further constraints might be used to encode deployment and configuration information. The constraints can be contained all in the templates, or provided in addition to the templates, to further limit the number of options for the grounded model.

Fig 12, Application performance model

The purpose of the Application Performance Model is to define the resource demands for each BPStep. There are two types of resource demand to consider.

1. The demand for resources generated directly by the ApplicationExecutionComponent (e.g. Dialog WP) using CPU, storage I/O, network I/O and memory when it executes the BPStep – the ComponentResourceDemand

2. The demand for resources generated by components that the above ApplicationExecutionComponent causes when it uses, calls or invokes other components (e.g. a Dialog WP using an Update WP) – the IndirectComponentResourceDemand

-44-

The IndirectComponentResourceDemand is recursive. So there will be a tree like a call-graph or activity-graph.

A complete Application Performance Model would contain similar information for all the BPSteps shown in Figure 11. For example the set of dialog steps in the BPStep "Create Sales Order" might consume 0.2 SAPS. Further it consists of 4 separate invocations (or in SAP terminology Dialog Steps). The calls are synchronous.

The following are some examples of attributes that can appear in IndirectComponentResourceDemands and ComponentResourceDemands.

- delayProperties: Any delay (e.g. wait or sleep) associated with the component's activity which does not consume any CPU, NetIOProperties and DiskIOProperties.
- NumInvocation: The number of times the component is called during the execution of the BPStep.
- InvocationType: synchronous if the caller is blocked; asynchronous if the caller can immediately continue activity.
- BPStepToAppCompID: This is the ID attribute of the BPStepToApplicationComponentMapping. The reason for this is that a particular ApplicationExecutionComponent is likely to be involved in several different BPSteps.
- ApplicationEntryPoint: This is the program or function being executed. In the case of "Create Sales Order" this is VA01 for the DialogWP. It could also be a method of a Web Service.

CPUProperties can be expressed in SAPs or in other units. There are various ways to express MemProperties, NetIOProperties and DiskIOProperties.

Fig 12, Component Performance Model

There is one instance of an Application Performance Model for each instance of a Custom Model. This is because, in the general case, each business process will have unique characteristics: a unique ordering of BPSteps and/ or a unique set of data characteristics for each BPStep. The DirectComponentResourceDemands and IndirectComponentResourceDemands associations specify the unique resource demands for each BPStep. These demands need to be calculated from known

-45-

characteristics of each ApplicationComponent derived from benchmarks and also traces of installed systems.

The Component Performance Model contains known performance characteristics of each ApplicationComponent. A specific Application Performance Model is calculated by combining the following:

- The information contained in the BPStepToApplicationComponentMapping associations in the Custom Model
- Any performance related constraints in the Application Constraints Model
- The Component Performance Model

Taken together, the models of the Unbound Model specify not only the non-functional requirements of a system, but also a recipe for how to generate and evaluate possible software and hardware configurations that meet those requirements. The generation of possible hardware configurations is constrained by the choice of infrastructure available from a specific Infrastructure Provider, using information in an Infrastructure Capability Model, and by the selected template.

A general principle that applies to deployable software elements described in the Unbound Model, such as the ApplicationExecutionComponent or ApplicationExecutionService, is that the model contains only the minimum number of instances of each type of element necessary to describe the structure of the application topology. For example, in the case of SD only a single instance of a Dialog Work Process ApplicationExecutionComponent associated with a single instance of an Application Server ApplicationExecutionService is needed in the Unbound Model to describe the myriad of possible ways of instantiating the grounded equivalents of both elements in the Grounded Model. It is the template and packaging information that determines exactly how these entities can be replicated and co-located.

The Infrastructure Capability Model

As discussed above, two notable features of the modelling philosophy described are:

1. Present a template having a finite catalogue of resources that can be instantiated, so that there are a fixed and finite number of choices. For example, small-xen-vm 1-disk, medium-xen-vm 2-disk, large-xen-vm 3-disk, physical-hpux-machine etc. This makes the selection of resource type by any capacity planning tool

-46-

simpler. It also makes the infrastructure management easier as there is less complexity in resource configuration – standard templates can be used.

2. Do not expose the hosting relationship for virtualized resources. The DMTF Virtualization System Profile models hosting relationship as a “HostedDependency” association. This does not seem to be required if there is only a need to model a finite number of resource types, so it does not appear in any of the models discussed here. This keeps the models simpler since there is no need to deal with arbitrary recursion. It does not mean that tools that process these models can’t use the DMTF approach internally if that is convenient. It may well be convenient for a Resource Directory Service and Resource Assignment Service to use this relationship in their internal models.

An instance of an infrastructure capability model contains one instance for each type of ComputerSystem or Device that can be deployed and configured by the underlying utility computing fabric. Each time the utility deploys and configures one of these types the configuration will always be the same. For a ComputerSystem this means the following.

- Same memory, CPU, Operating System
- Same number of NICs with same I/O capacity
- Same number of disks with the same characteristics

Fig 13 template example

Fig 13 shows an example of an infrastructure design template having predetermined parts of the computing infrastructure, predetermined relationships between the parts, and having a limited number of options to be completed. In this case it is suitable for a decentralised SD business process, without security or availability features. The figure shows three computer systems coupled by a network labelled “AI_network”, the right hand of the three systems corresponding to a master application server, and the central one corresponds to slave application servers as shown in figure 7. Hence it is decentralized. AI is an abbreviation of Adaptive Infrastructure. The left hand one of the computer systems is for a database. The type of each computer system is specified, in this case as a BL20/Xen. The central one, corresponding to slave application servers has an attribute “range = 0..n”. This means the template allows any number of these slave application servers.

-47-

The master application server is coupled to a box labelled AI_GroundedExecutionService:AppServer, indicating it can be used to run such a software element. It has an associated AIDeploymentSetting box which contains configuration information and deployment information sufficient to allow the

5 AI_GroundedExecutionService to be automatically installed, deployed and managed. The AI_GroundedExecutionService:AppServer is shown as containing three components, labelled AI_GroundedExecutionComponents, and each having an associated AIDeploymentSetting box. A first of these components is a dialog work process, for executing the application components of steps of the business process,

10 another is an update process, responsible for committing work to persistent storage, and another is an enqueue process, for managing locks on a database. As shown, the range attribute is 2..n for the update and the dialog work process, meaning multiple instances of these parts are allowed.

The slave application server has a GroundedExecutionService having only one type of

15 AI_GroundedExecutionComponent for any number of dialog work processes. The slave application server is shown having a rangePolicy = Time function, meaning it is allowed to be active at given times. Again the service and the execution component each have an associated AIDeploymentSetting box.

The master and slave application servers and the database computer system have an

20 operating system shown as AI_disk: OSDisk. The master application server is shown with an AI_Disk: CIDisk as storage for use by the application components. For the network, each computer system has a network interface shown as AI_Nic1, coupled to the network shown by AI_Network :subnet1.

The database computer system is coupled to a box labelled

25 AI_GroundedExecutionService: Database, which has only one type of AI_GroundedExecutionComponent, SD DB for the database. Again the service and the execution component each have an associated AIDeploymentSetting box. AIDeploymentSetting carries the configuration and management information used to deploy, configure, start, manage and change the component. Further details of an

30 example of this are described below with reference to figure 14. This computer system is coupled to storage for the database labelled AI_Disk: DBDisk.

Optionally the template can have commands to be invoked by the tools, when generating the grounded model, or generating a changed grounded model to change an

existing grounded model. Such commands can be arranged to limit the options available, and can use as inputs, parts of the template specifying some of the infrastructure design. They can also use parts of the unbound model as inputs.

5 Fig 14 Grounded Model

The Grounded Model may be generated by a design tool as it transforms the Unbound Model into the Grounded Model. It can be regarded as a candidate Grounded Model until evaluated and selected as the chosen Grounded Model. The following are some of the characteristics of the example Grounded Model of figure 14 compared to the template shown in Fig 13, from which it is derived.

- The number of instances of GroundedExecutionComponent has been specified.
- The GroundedExecutionComponents are executed by a GroundedExecution-Service. The execution relationship is consistent with that expressed in the Application Packaging Model.
- The GroundedExecutionServices are run on a ComputerSystem whose type has been selected from the Infrastructure Capability Model.
- There are two update components, Update1 and Update2. There are two DialogWorkProcesses, DialogWorkProcess1 and DialogWorkProcess2.
- The number of slave application servers has been set at zero.

The management system is arranged to make these choices to derive the Grounded Model from the template using the Unbound Model. In the example shown, the criteria used for the choice includes the total capacity of the system, which must satisfy the time varying Performance Requirements in the Custom Model. The required capacity is determined by combining these Performance Requirements with the aggregated ResourceDemands [Direct and Indirect] of the Application Performance Model. If the first choice proves to provide too little capacity, or perhaps too much, then other choices can be made and evaluated. Other examples can have different criteria and different ways of evaluating how close the candidate grounded model is to being a best fit.

In some examples the server may only have an OS disk attached; that is because the convention in such installations is to NFS mount the CI disk to get its SAP executable files. Other example templates could have selectable details or options such as details

-49-

of the CIDisk and the DBDisk being 100 GB, 20MB/sec, non Raid, and so on. The OS disks can be of type EVA800. The master and slave application servers can have 2 to 5 dialog work processes. Computer systems are specified as having 3 GB storage, 2.6 GHz CPUs and SLES 10-Xen operating system for example. Different parameters can be tried to form candidate Grounded Models which can be evaluated to find the best fit for the desired performance or capacity or other criteria.

The Grounded Model therefore specifies the precise number and types of required instances of software and hardware deployable entities, such as GroundedExecutionComponent, GroundedExecutionService, and AIComputerSystem. AIDeploymentSettings can include for example:

- InfrastructureSettings such as threshold information for infrastructure management components, for example MaxCPUUtilization – if it rises above the set figure, say 60%, an alarm should be triggered.
- Management policy can specify further policy information for the management components – e.g. flex up if utilization rises above 60%
- GroundedDeploymentSettings which can include all command line and configuration information so that the system can be installed, configured and started in a fully functional state.
- SettingData which can provide additional configuration information that can override information provided in the GroundedDeploymentSettings. This allows many GroundedComponents to share the same GroundedDeploymentSettings (c.f. a notion of typing) with specific parameters or overrides provided by SettingData. Both the GroundedDeploymentSettings and SettingData are interpreted by the Deployment Service during deployment.
- Data related to possible changes to the component such as instructions to be carried out when managing changes to the component, to enable more automation of changes.

Not all attributes are set in the Grounded Model. For example, it does not make sense to set MAC addresses in the Grounded Model, since there is not yet any assigned physical resource.

Fig 15, an alternative adaptive infrastructure design template

Figure 15 shows an alternative adaptive infrastructure design template, in a form suitable for a centralised secure SD business process. Compared to fig 13, this has only one computer system, hence it is centralised. It shows security features in the form of a connection of the network to an external subnet via a firewall. This is shown by an interface AI_Nic:nicFW, and a firewall shown by AI_Appliance: FireWall.

Other templates can be envisaged having any configuration. Other examples can include a decentralised secure SD template, a decentralised highly available SD template, and a decentralised, secure and highly available SD template.

Bound Model

A Bound Model Instance for a SD system example could have in addition to the physical resource assignment, other parameters set such as subnet masks and MAC addresses. A Deployed Model could differ from the Bound Model in only one respect. It shows the binding information for the management services running in the system. All the entities would have management infrastructure in the form of for example a management service. The implementation mechanism used for the interface to the management services is not defined here, but could be a reference to a Web Service or a SmartFrog component for example. The management service can be used to change state and observe the current state. Neither the state information made available by the management service, nor the operations performed by it, are necessarily defined in the core of the model, but can be defined in associated models.

One example of this could be to manage a virtual machine migration. The application managing the migration would use the management service running on the PhysicalComputerSystem to do the migration. Once the migration is completed, the management application would update the deployed model and bound models to show the new physical system. Care needs to be taken to maintain consistency of models. All previous model instances are kept in the model repository, so when the migration is complete, there would be a new instance (version) of the bound and deployed models.

Information Hiding and the Model Information Flow

It is not always the case that for the MIF all tools and every actor can see all the information in the model. In particular it is not the case for deployment services having a security model which requires strong separation between actors. For example, there can be a very strong separation between the utility management plane and farms of virtual machines. If a grounded model is fed to the deployment services of the management plane for an enterprise, it will not return any binding information showing the binding of virtual to physical machines, that information will be kept inside the management plane. That means there is no way of telling to what hardware that farm is bound or what two farms might be sharing. What is returned from the management plane is likely to include the IP address of the virtual machines in the farms (it only deals with virtual machines) and the login credentials for those machines in a given farm. The management plane is trusted to manage a farm so that it gets the requested resources. Once the deployment service has finished working, one could use application installation and management services to install, start and manage the applications. In general different tools will see projections of the MIF. It is possible to extract from the MIF models the information these tools require and populate the models with the results the tools return. It will be possible to transform between the MIF models and the data format that the various tools use.

Implementation:

The software parts such as the models, the model repository, and the tools or services for manipulating the models, can be implemented using any conventional programming language, including languages such as Java, or C compiled following established practice. The servers and network elements can be implemented using conventional hardware with conventional processors. The processing elements need not be identical, but should be able to communicate with each other, e.g. by exchange of IP messages.

The foregoing description of embodiments of the invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed, and modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention. The embodiments were chosen and described in order to explain the

-52-

principles behind the invention and its practical applications to enable one skilled in the art to utilize the invention in various embodiments and with various modifications as are suited to the particular use contemplated. Other variations can be conceived within the scope of the claims.

CLAIMS

1. A method of providing an operator interface for developing a computer based business process for deployment by a service provider, the business process having a number of functional steps, the method having the steps of:
 - 5 providing model of the business process, the model specifying the functional steps, specifying an arrangement of software application components for carrying out the functional steps, and specifying a design of computing infrastructure, for running the software application components, to meet given non functional requirements, and suitable for automated deployment,
 - 10 generating a three dimensional view of at least part of the model for the operator, translating operator inputs to manipulate representations in the three dimensional view into commands to change corresponding parts of the functional steps, or software application components or computing infrastructure design in the model.
- 15 2. The method of claim 1, having the step of deploying the model with the changes.
3. The method of claim 1, having the step of determining if the changes are allowable changes, before they are deployed.
- 20 4. The method of claim 1, the operator interface being suitable for multiple operators, the method having the steps of generating three dimensional views for the multiple operators simultaneously, and broadcasting changes to the model to all operators.
5. The method of claim 3, having the step of using one of a number of predetermined
25 templates of computing infrastructure design, and using the template to determine if the changes are allowable changes.
6. The method of claim 1, the design of the computing infrastructure comprising virtualized entities.
- 30 7. The method of claim 3, having the step of restricting allowable changes according to a class of the operator.

-54-

8. The method of claim 1, having the steps of generating the model by generating an unbound model specifying the application components to be used for each of the computer implemented steps of the business process, without a complete design of the computing infrastructure, and from the unbound model generating a grounded model specifying a complete design of the computing infrastructure suitable for automatic deployment of the business process.

9. The method of claim 1, having the step of determining consequential changes to other parts of the model from the operator input.

10. The method of claim 1, the operator input to manipulate the representations comprising changing any one or more of shape, appearance, and location of representations in the view.

11. The method of claim 1, having the step of monitoring behaviour of the deployment, and presenting a visual representation of the monitored behaviour in the view.

12. A system for providing an operator interface for developing a computer based business process for deployment by a service provider, the business process having a number of functional steps, the system having:

a model of the business process, the model specifying the functional steps, specifying an arrangement of software application components, for carrying out the functional steps, and specifying a design of computing infrastructure, for running the software application components to meet given non functional requirements, and suitable for automated deployment, the system also having;

a three dimensional visual rendering part coupled to the model store and arranged to generate a three dimensional view of at least part of the model for the operator,

an input interpreter arranged to receive and translate operator inputs manipulating representations in the three dimensional view into commands to change corresponding parts of the functional steps, or software application components or computing infrastructure design in the model.

-55-

13. The system of claim 12, having a deployment part for carrying out the automated deployment of the process based on the model.

5 14. The system of claim 12, the input interpreter being arranged to determine if the changes are allowable changes.

10 15. The system of claim 12, the operator interface being suitable for multiple operators, the visual rendering part being arranged to generate three dimensional views for the multiple operators simultaneously, and to broadcast changes to the model to all operators.

15 16. The system of claim 14, the model generation part being arranged to use one of a number of predetermined templates of computing infrastructure design, the input interpreter being arranged to use the template to determine if the changes are allowable changes.

17. The system of claim 12, the computing infrastructure comprising virtualized entities.

20 18. The system of claim 14, the input interpreter being arranged to restrict allowable changes according to a class of the operator.

25 19. The system of claim 12, having a model generator arranged to generate the model by generating an unbound model specifying the application components to be used for each of the computer implemented steps of the business process, without a complete design of the computing infrastructure, and from the unbound model generate a grounded model specifying a complete design of the computing infrastructure suitable for automatic deployment of the business process.

30 20. The system of claim 12, the model generator being arranged to determine consequential changes to the rest of model from the operator input.

-56-

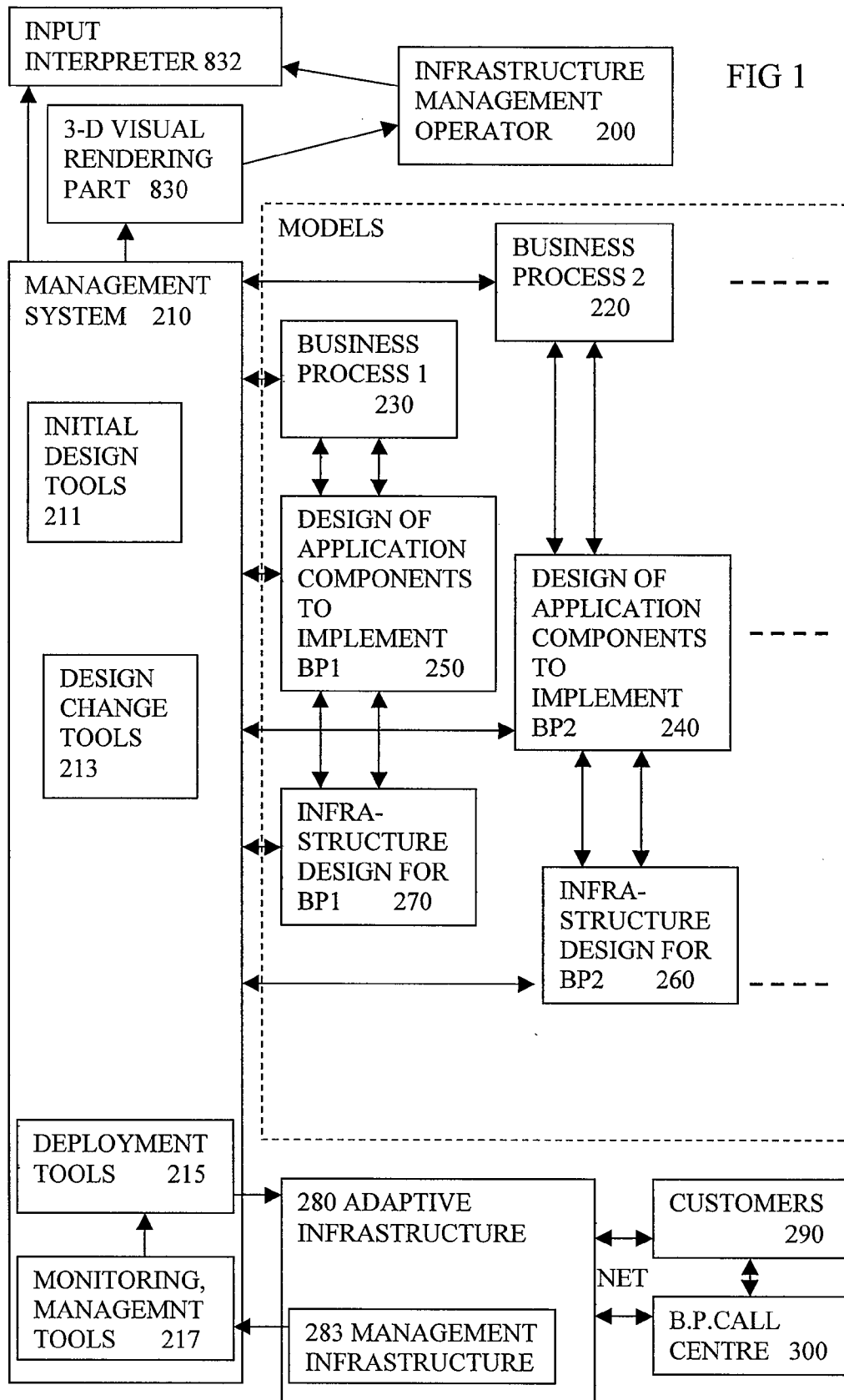
21. The system of claim 12, having monitors to monitor behaviour of the deployment, the visual rendering part being coupled to the monitor to present a visual representation of the monitored behaviour in the view.

5 22. The system of claim 12, the visual rendering part having a stored mapping of modelled entities to rendered items, and the input interpreter having a stored mapping of operator actions on rendered items to changes to the model.

10 23. Software on a machine readable medium which when executed carries out the method of claim 1.

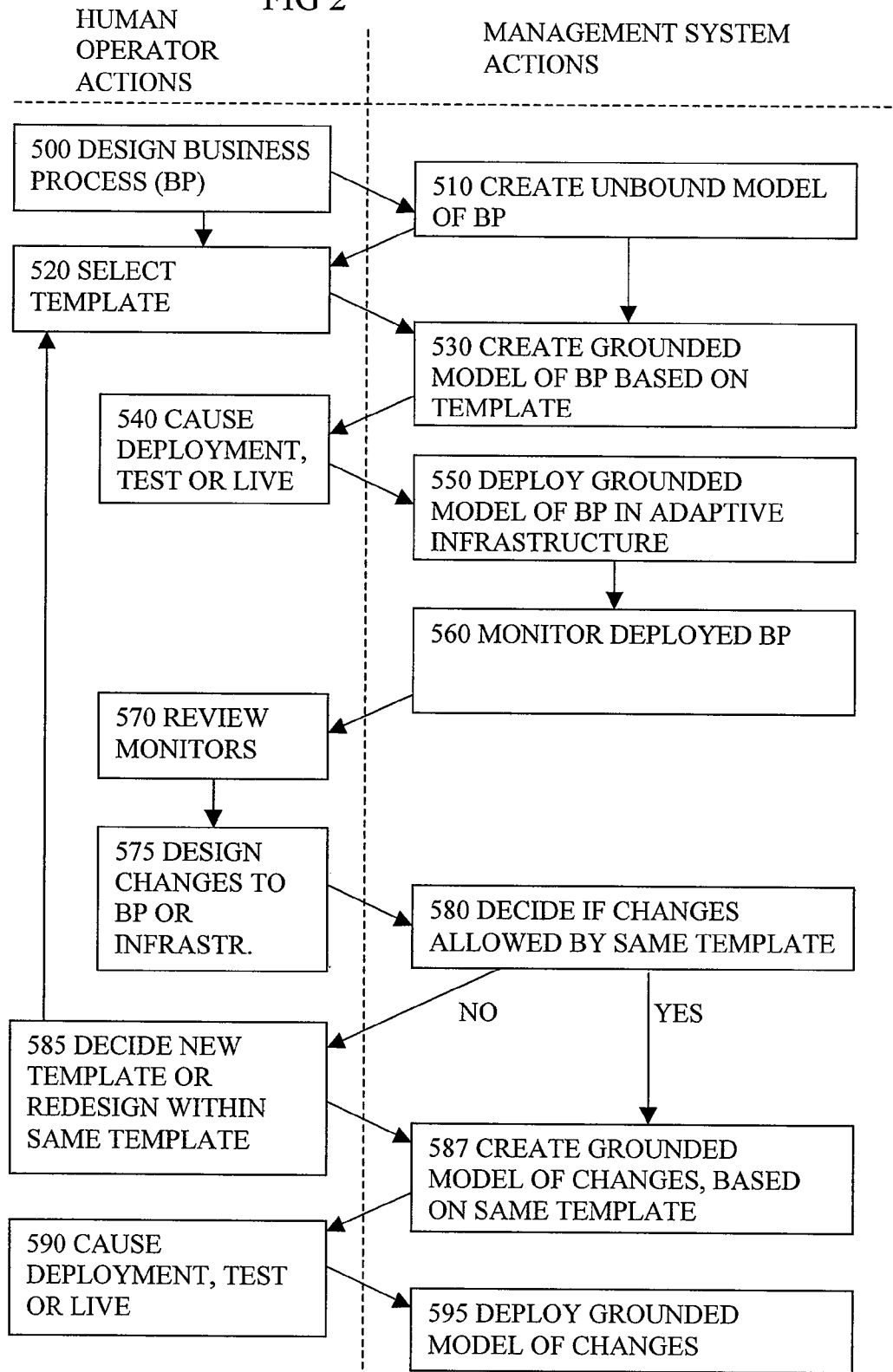
24. A method of using an operator interface for developing a computer based business process on physical infrastructure of a service provider, the business process having a number of functional steps, the system having a model generating part for generating
15 a software model of the process, suitable for automated deployment on the shared physical infrastructure, to meet given non functional requirements, by generating for the model a representation of an arrangement of software application components, for carrying out the functional steps, and a design of computing infrastructure, for running the software application components, the method having the steps of:
20 using an operator terminal to view a three dimensional view of at least the computing infrastructure design of the model,
causing changes to the model by inputting actions associated with entities in the three dimensional view, for translation into changes to corresponding parts of the model, and
25 causing the system to carry out the automated deployment of the process based on the model, on the shared physical infrastructure.

1/21



2/21

FIG 2



3/21

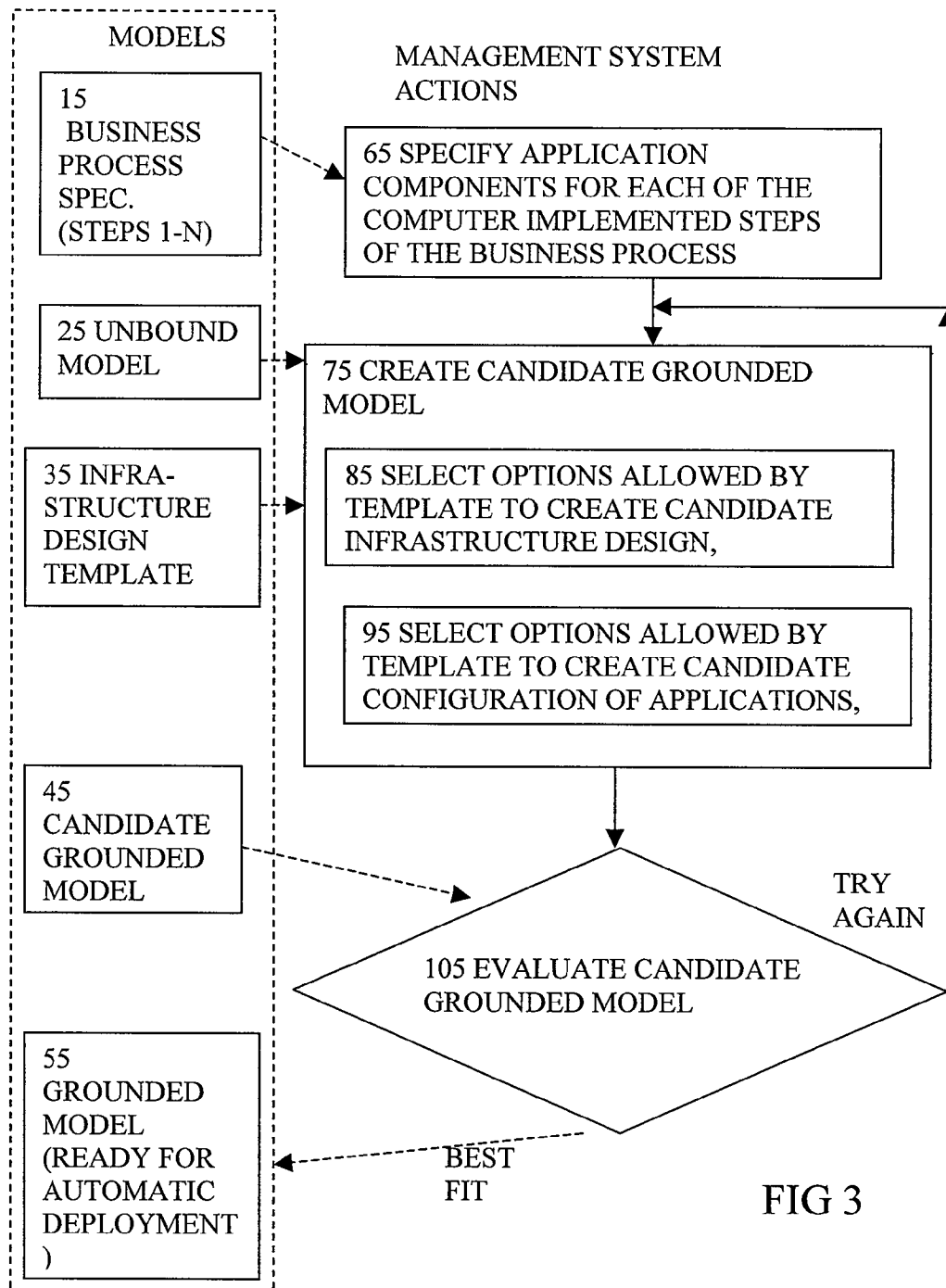
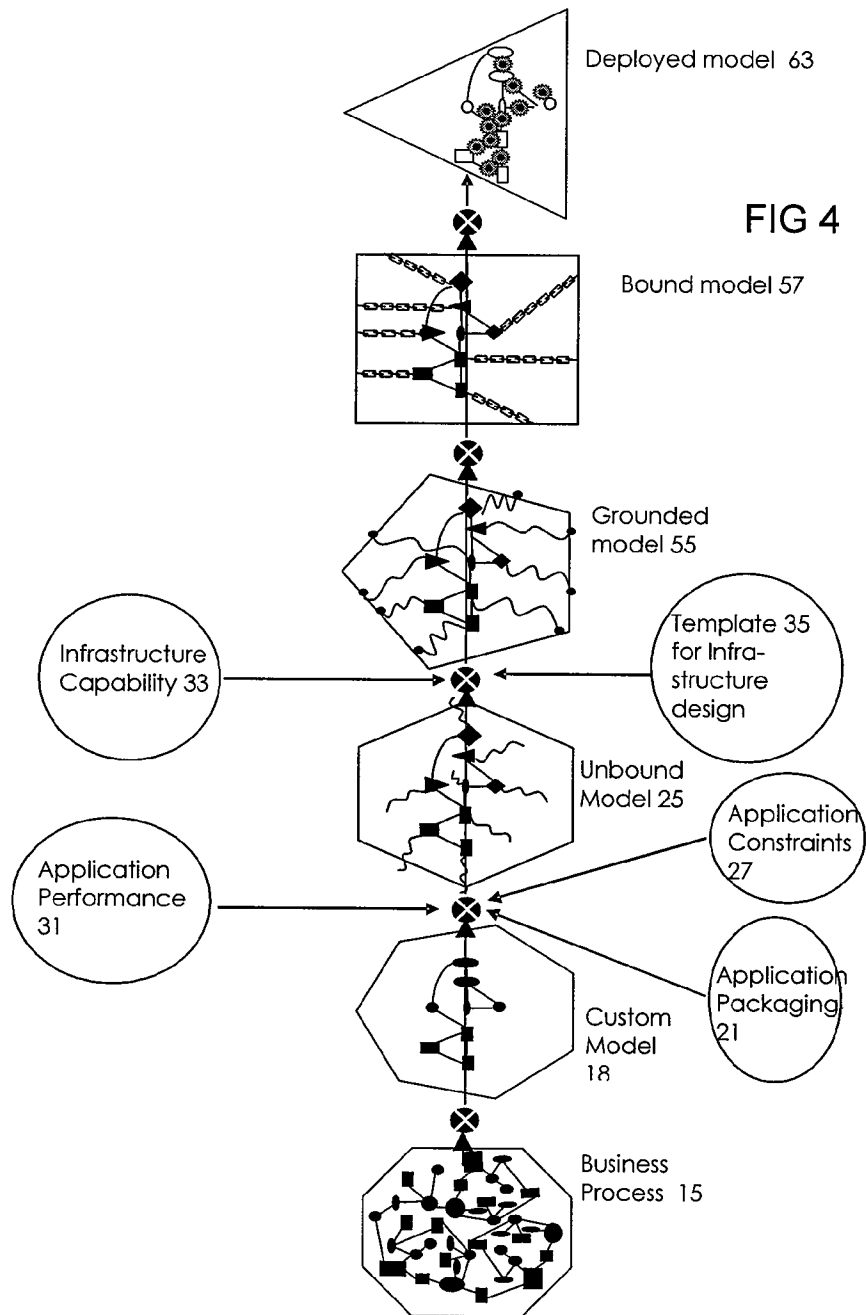


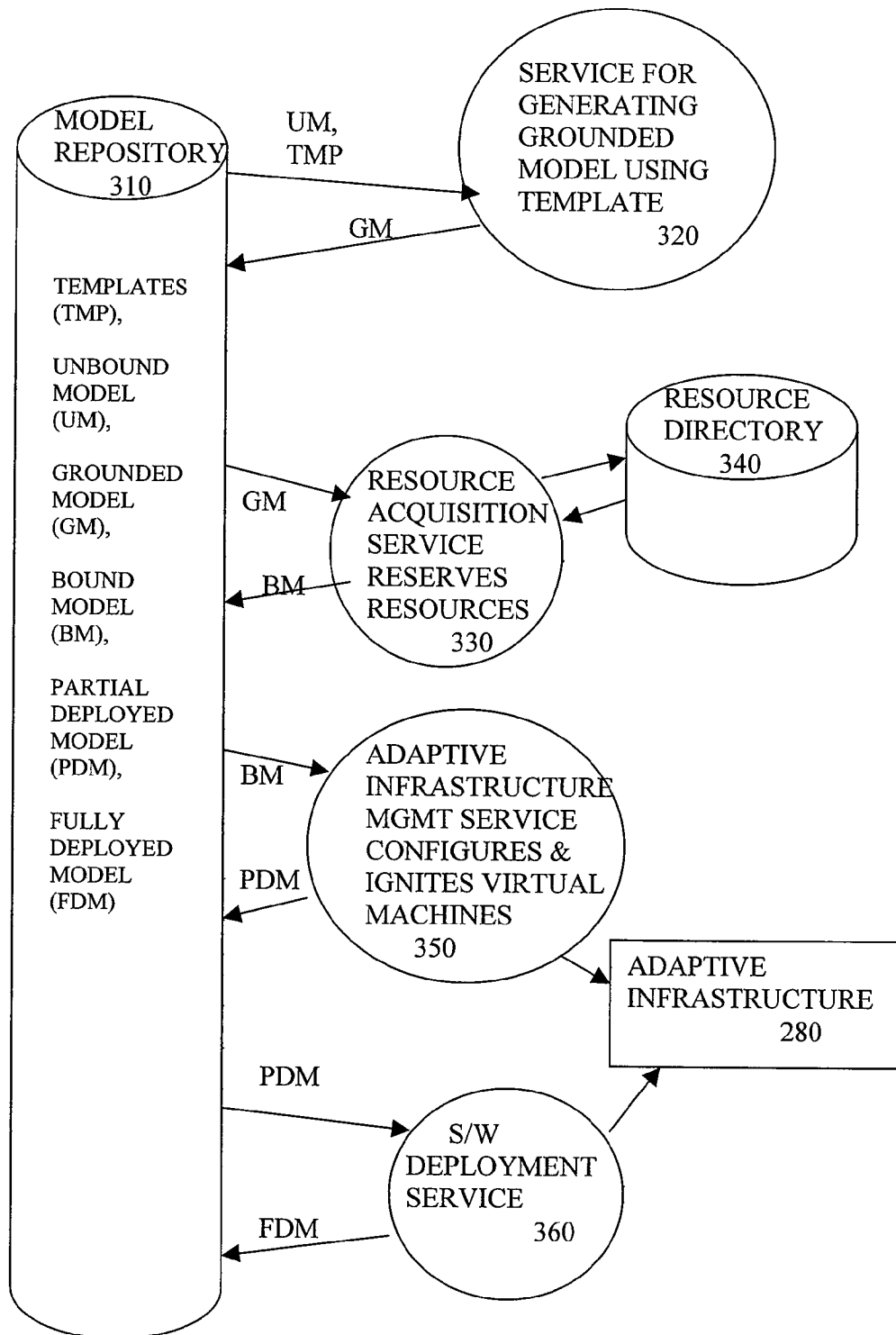
FIG 3

4/21



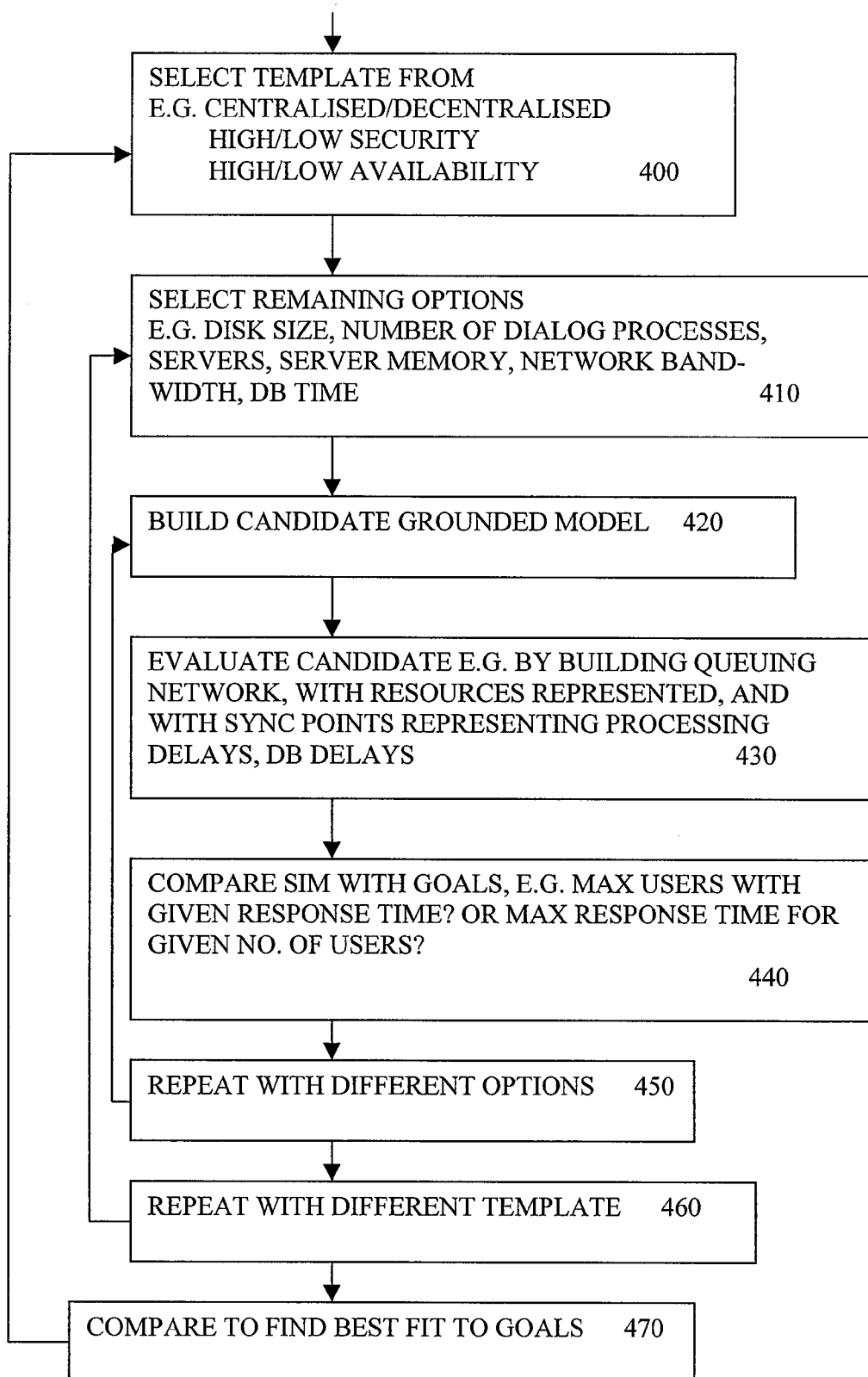
5/21

FIG 5



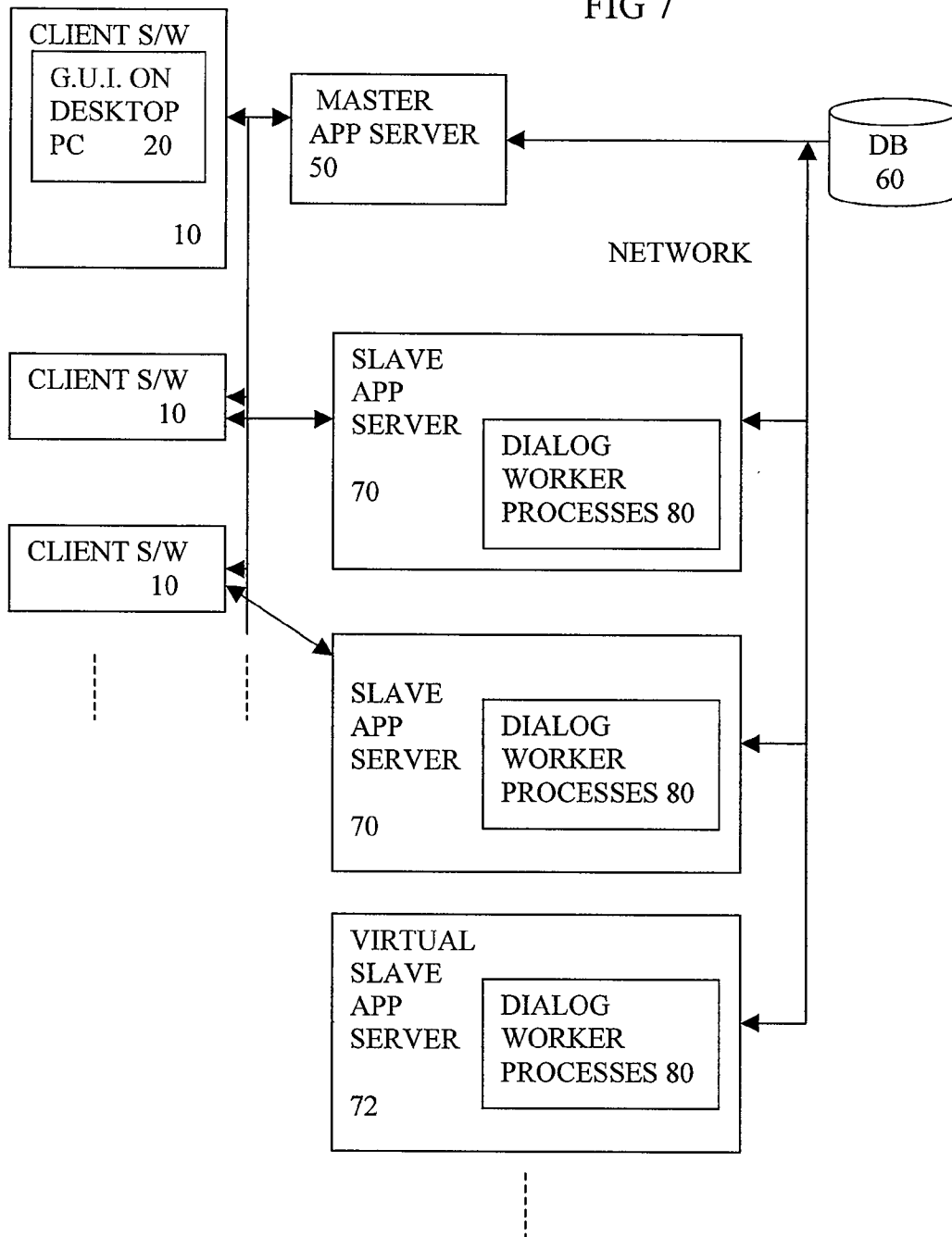
6/21

FIG 6



7/21

FIG 7



8/21

FIG 8

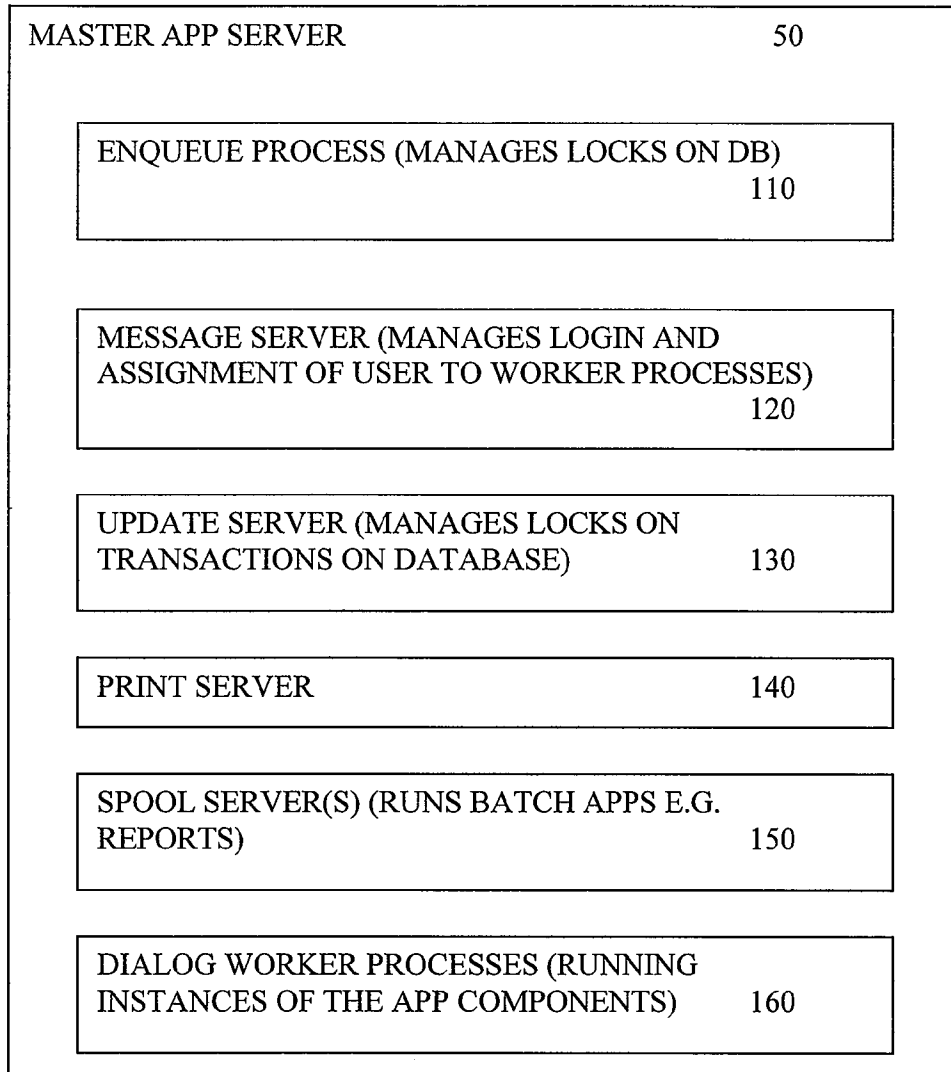
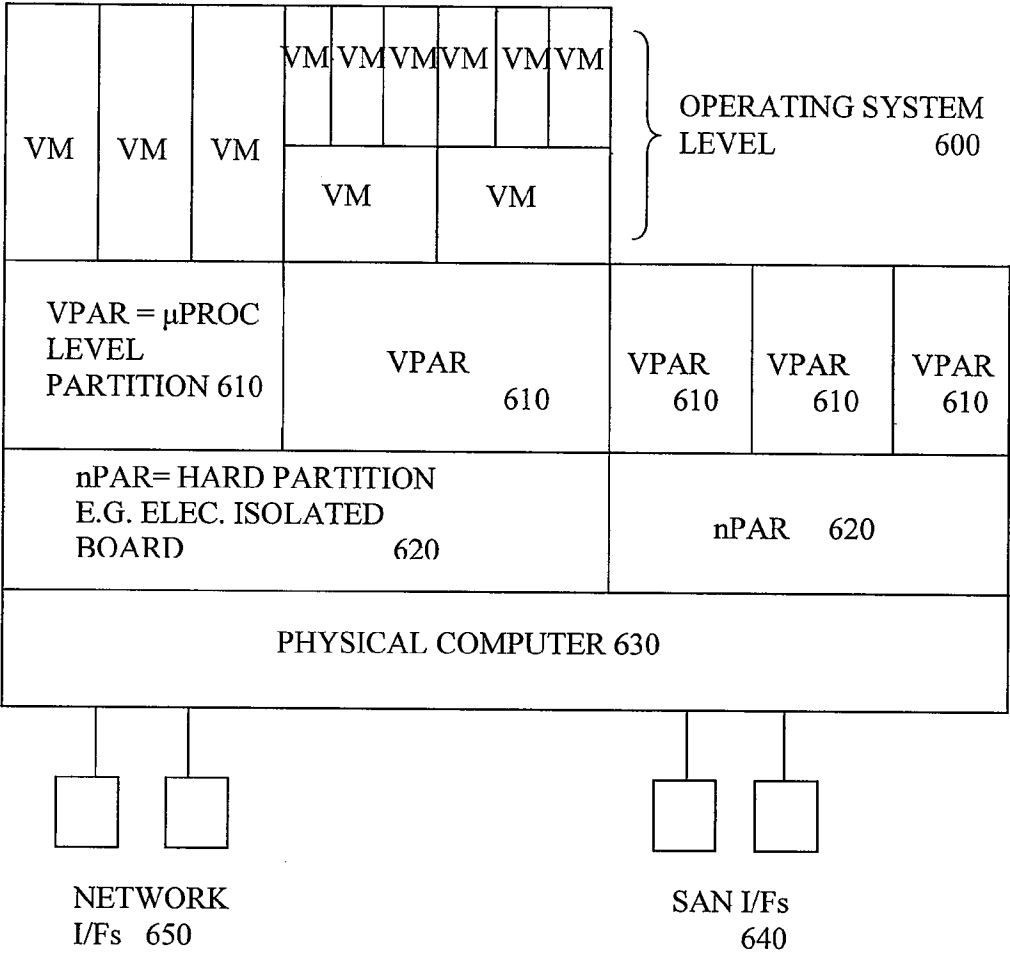


FIG 9



10/21

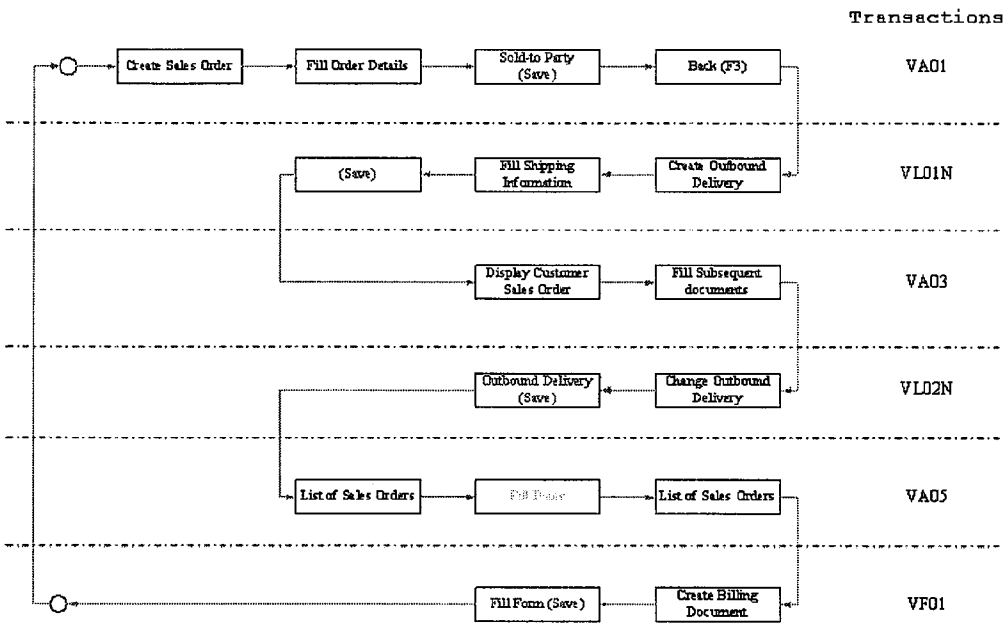


FIG 10

11/21

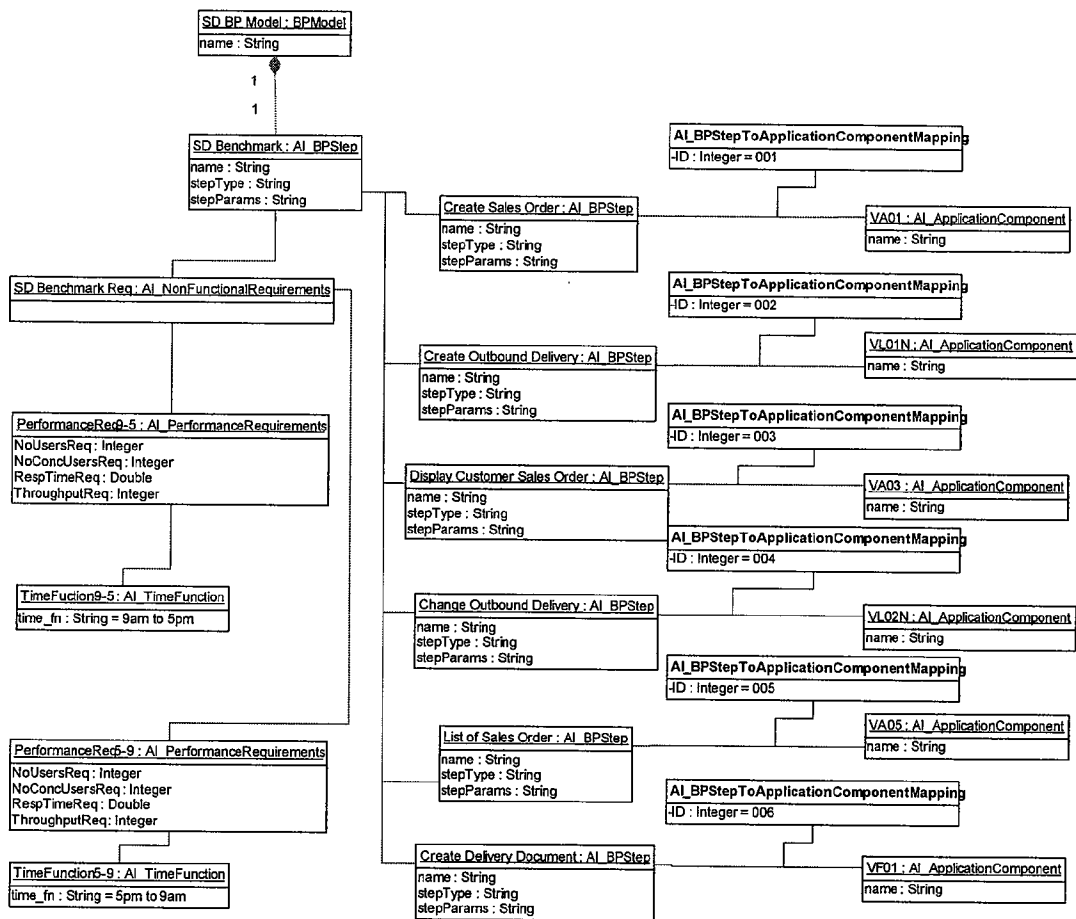


FIG 11

12/21

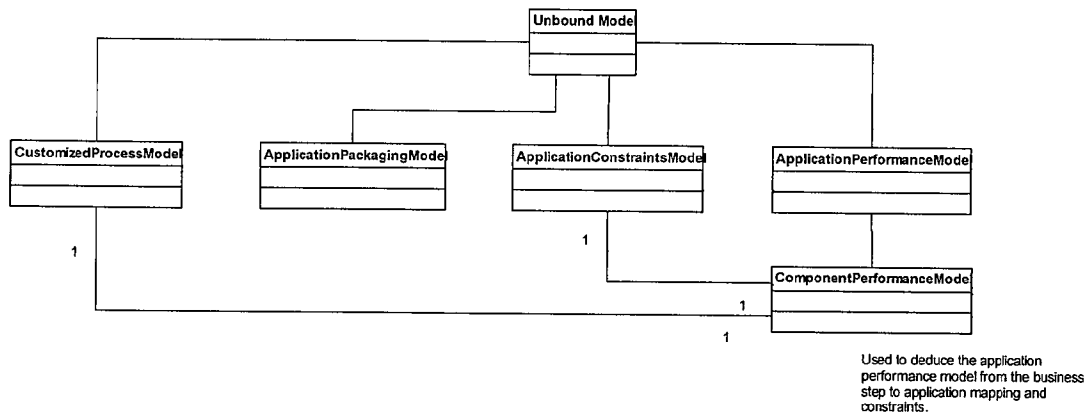


FIG 12

13/21

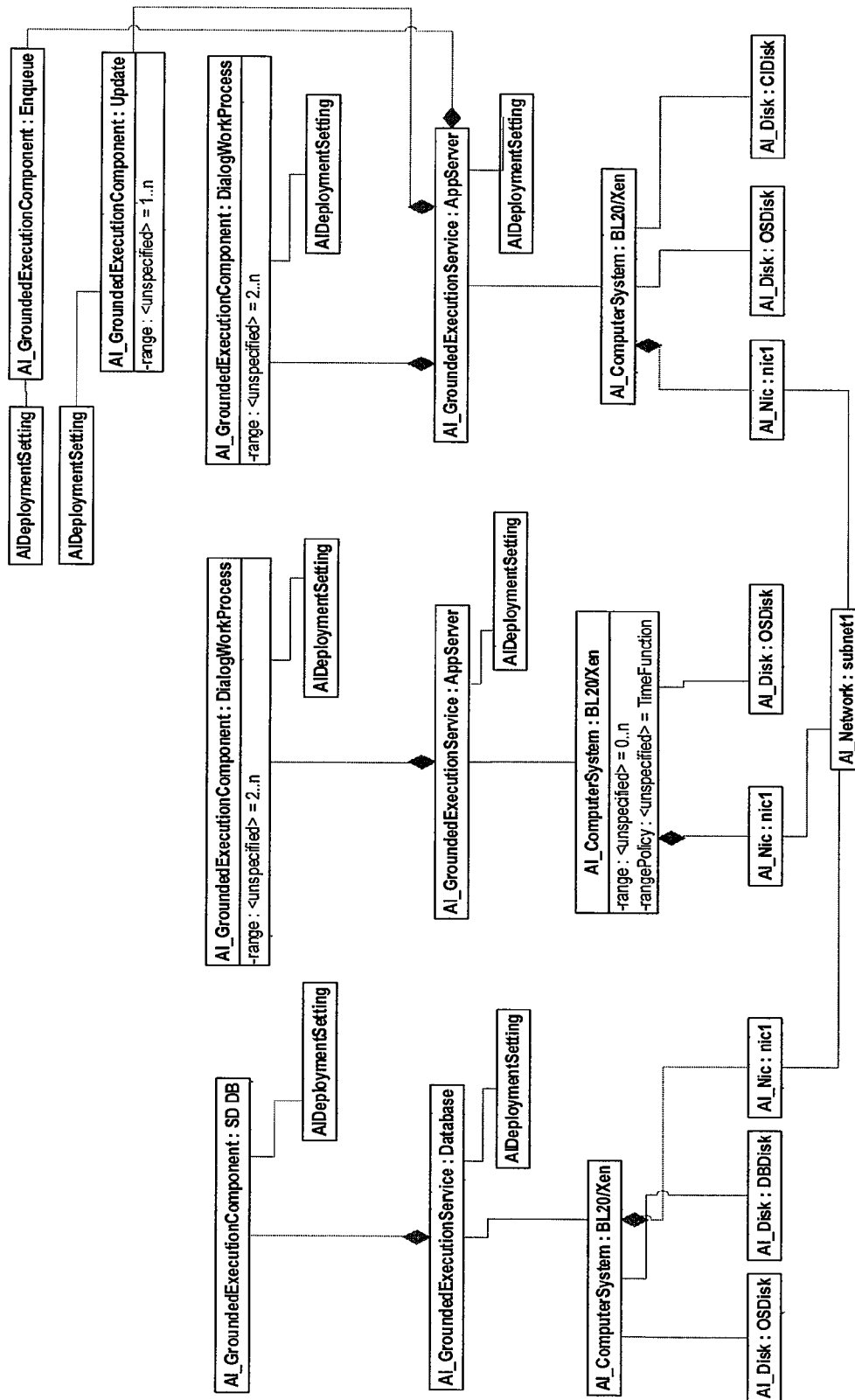


FIG 13

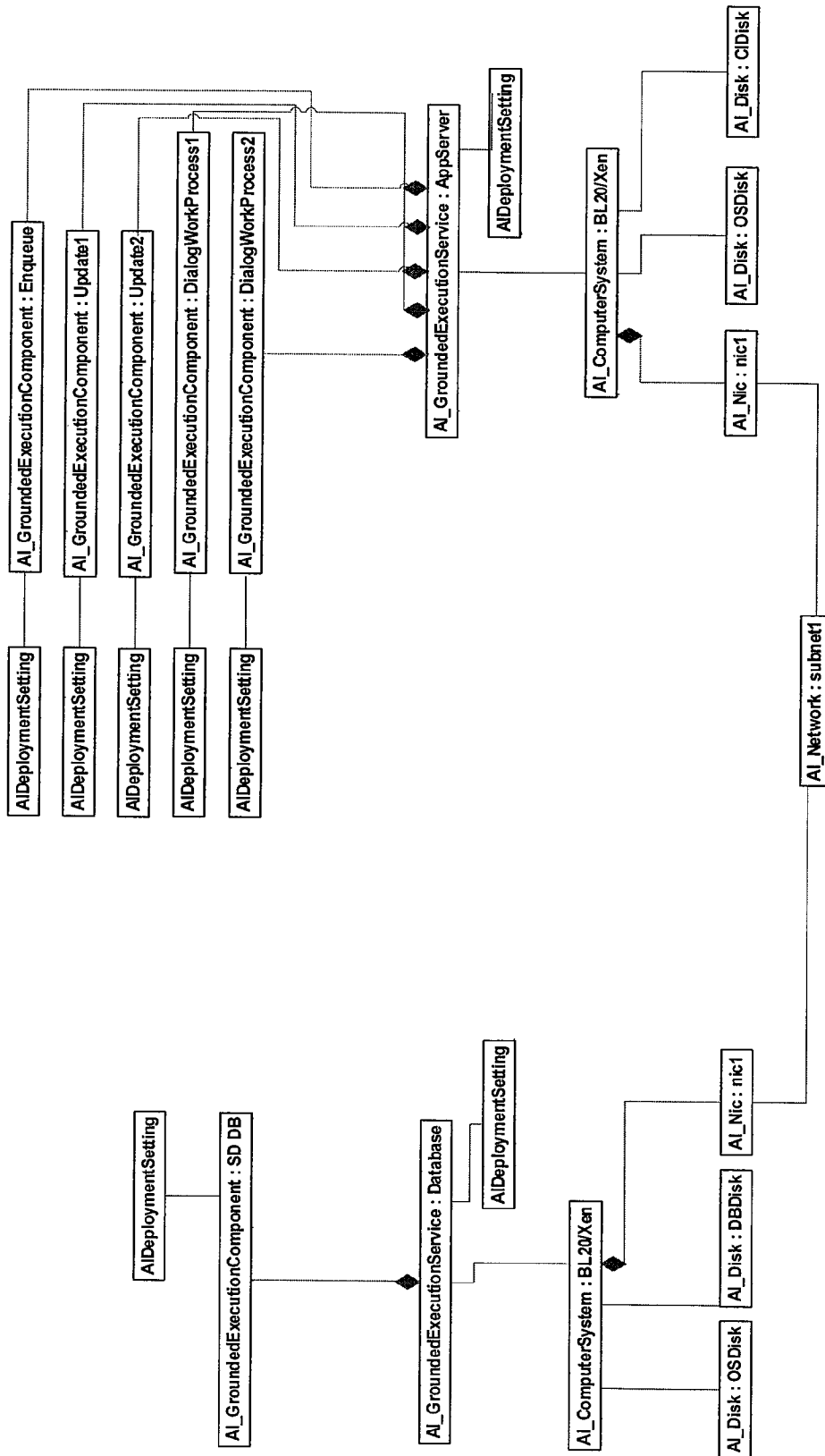


FIG 14

15/21

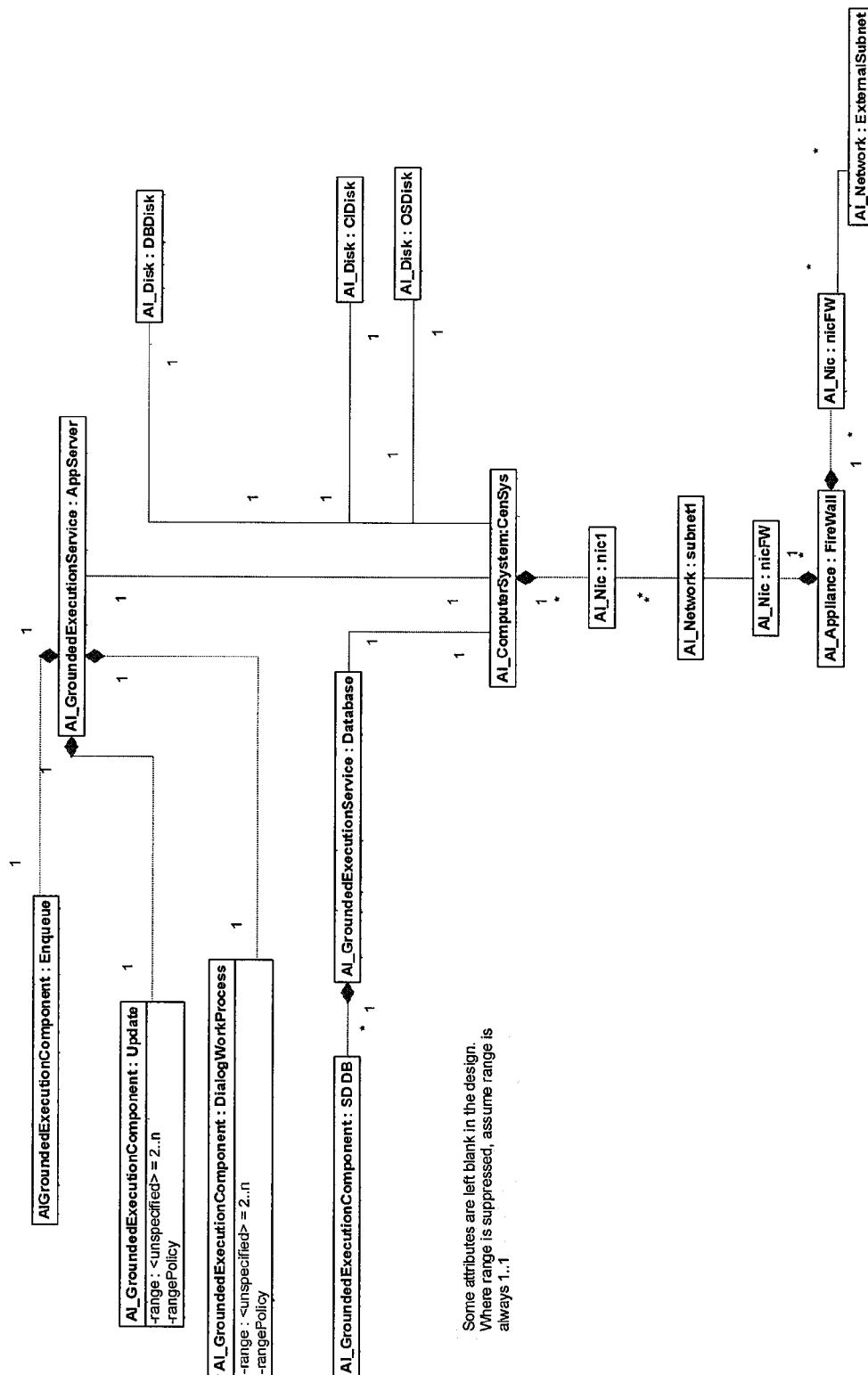
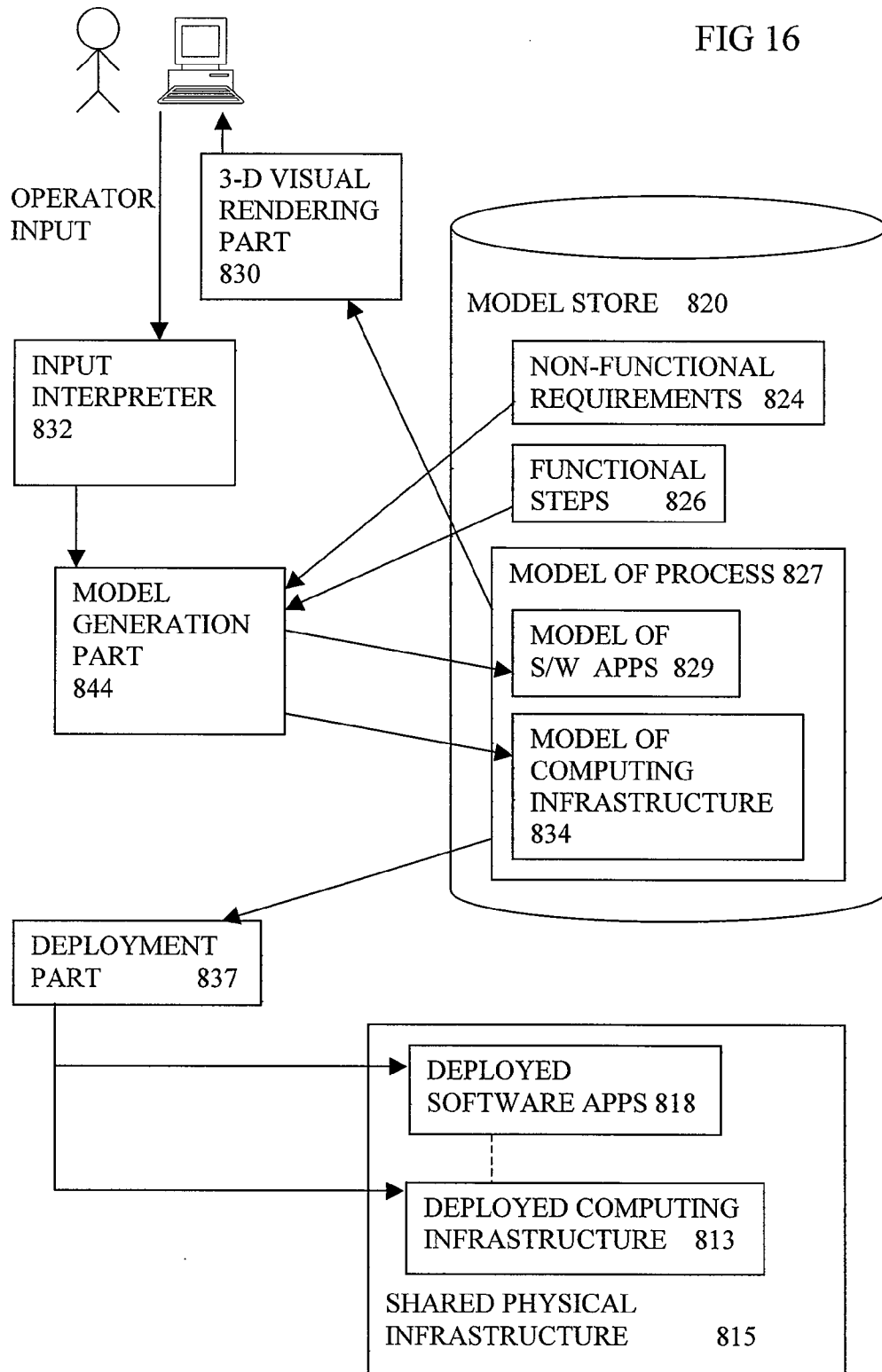


FIG 15

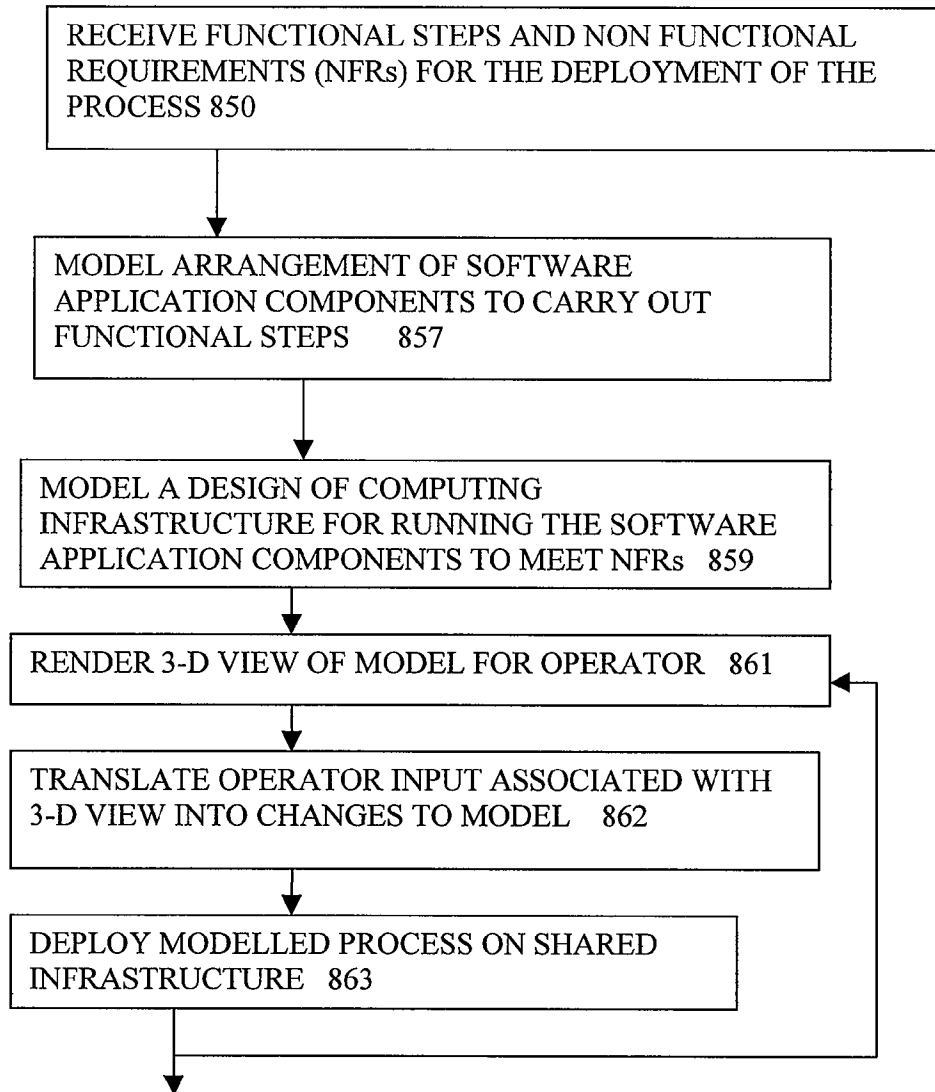
16/21

FIG 16



17/21

FIG 17



18/21

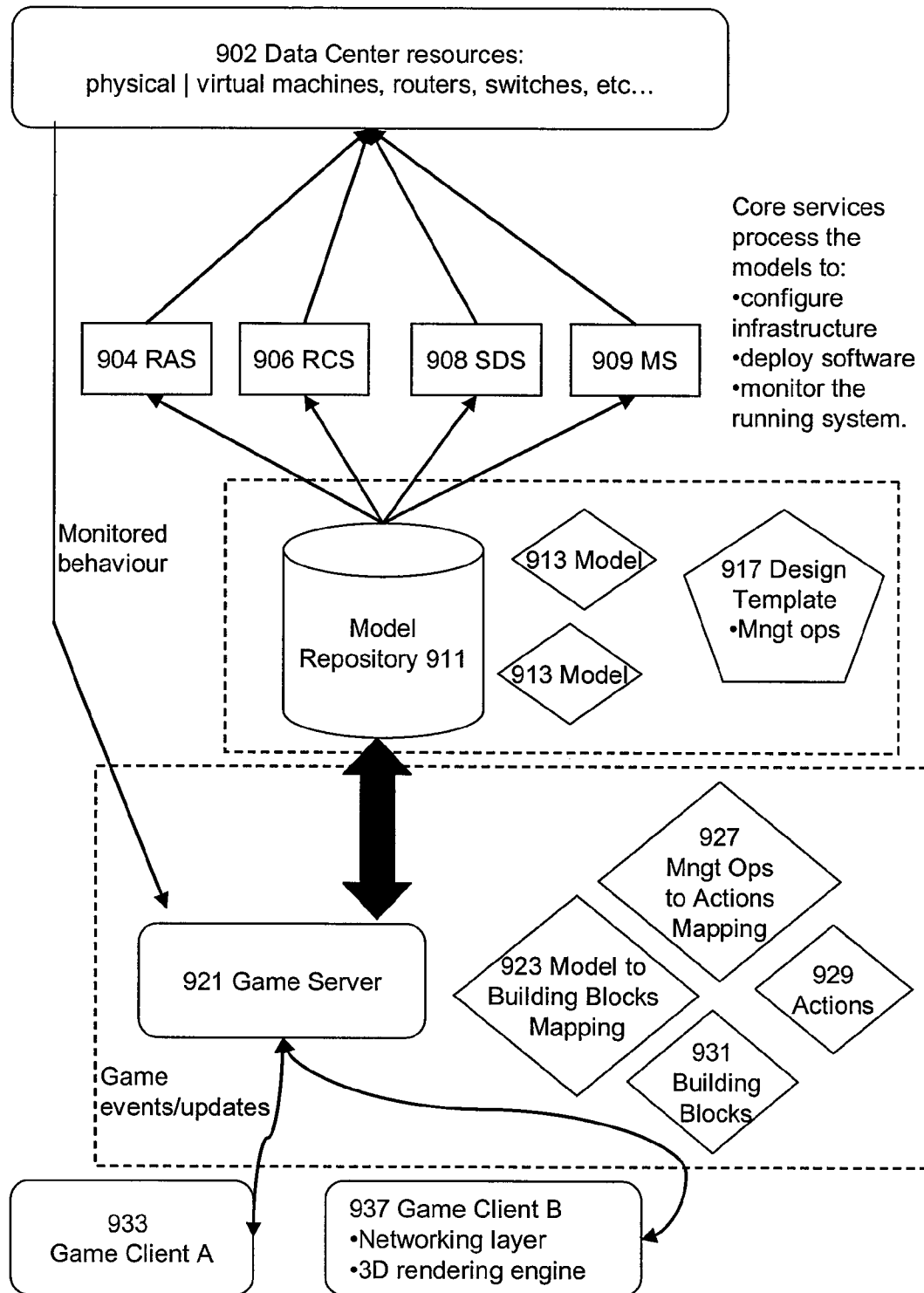


FIG 18

19/21

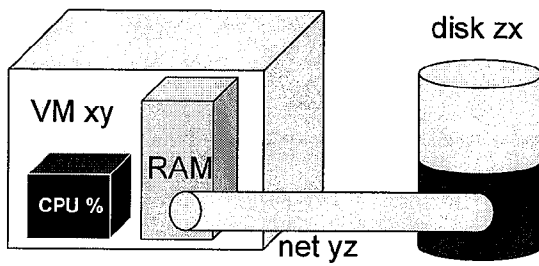
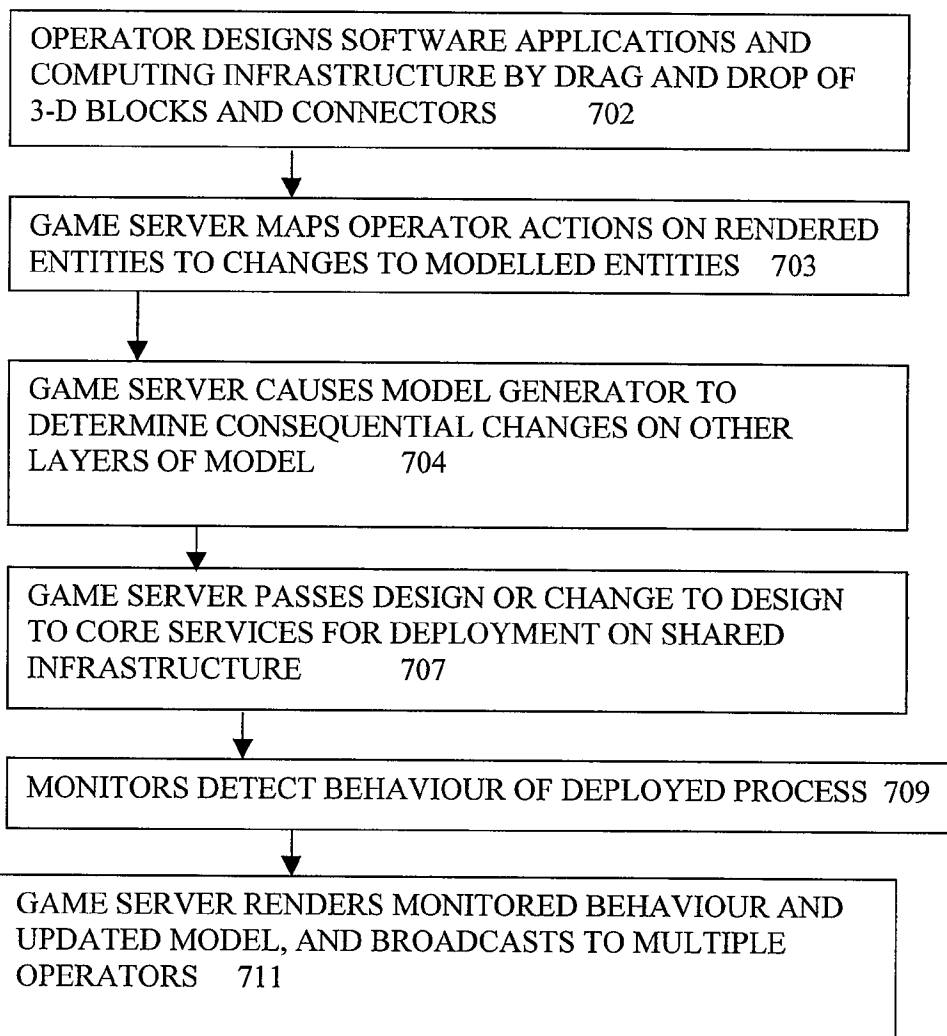


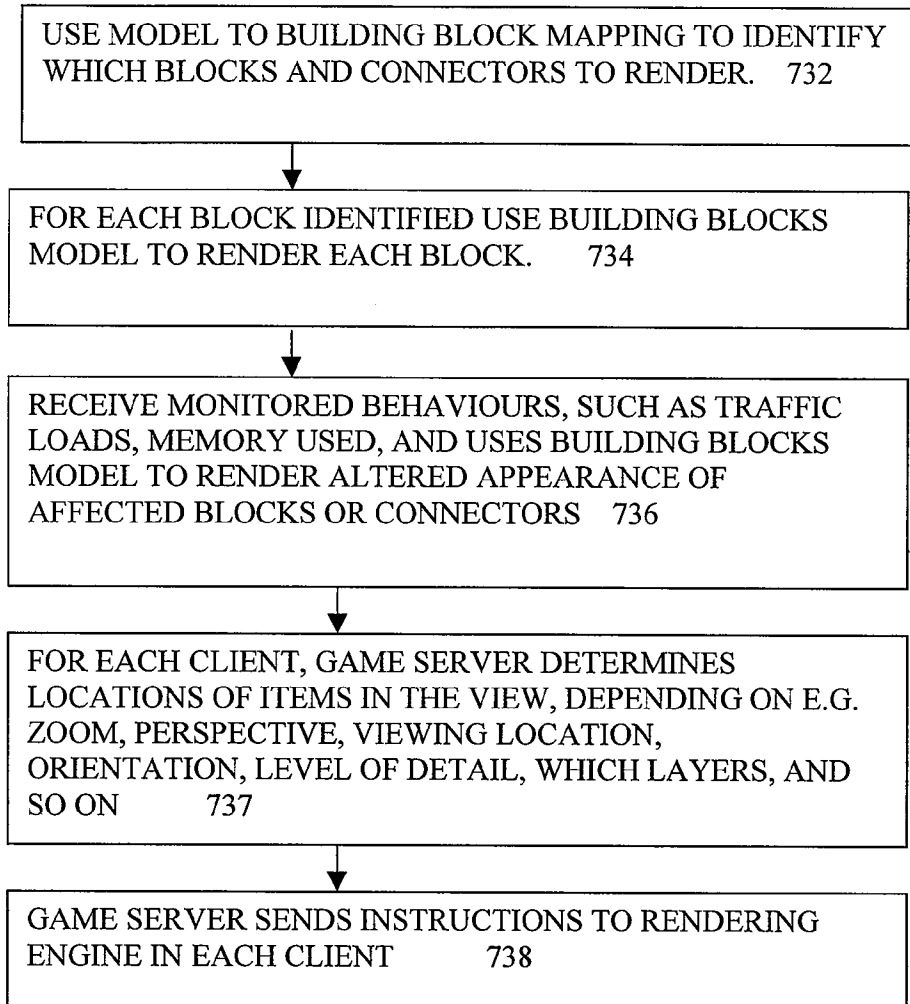
FIG 19

FIG 20



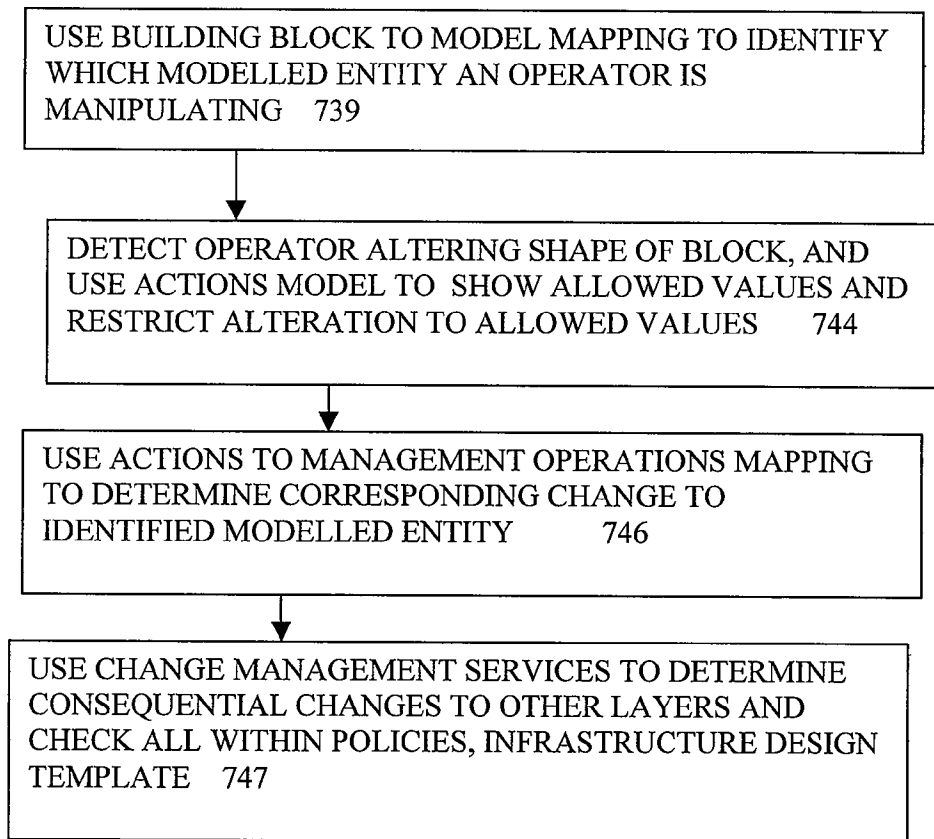
20/21

FIG 21



21/21

FIG 22



A. CLASSIFICATION OF SUBJECT MATTER**G06Q 50/00(2006.01)i**

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)

IPC 8 : G05B 19/418, G06F 17/60, G06F 15/173, G06T 15/00, G06T 17/00

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

Korean Utility models and applications for Utility models since 1975

Japanese Utility models and applications for Utility models since 1975

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

eKIPASS "BUSINESS PROCESS, MODEL, GENERATING, TRANSLATING OPERATOR, THREE DIMENSIONAL VIEW, SOFTWARE, INFRASTRUCTURE, USER INTERFACE"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2004-0002891 A1 (KAY-YUT CHEN et al.) 01 January 2004 See the Abstract, Claims 1, 11, 15 and Figures 1-4	1-24
A	US 2007-0179828 A1 (ALEX ELKIN et al.) 02 August 2007 See the Abstract, Claims 1, 15, 26, 27 and Figures 1-21	1-24
A	US 7239311 B2 (RICHARD S. DUNN et al.) 03 July 2007 See the Abstract and Figure 1	1-24
A	US 2004-19688 A1 (RAND B. NICKERSON et al.) 29 January 2004 See the Abstract	1-24



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

22 AUGUST 2008 (22.08.2008)

Date of mailing of the international search report

25 AUGUST 2008 (25.08.2008)

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
Government Complex-Daejeon, 139 Seonsa-ro, Seo-
gu, Daejeon 302-701, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

LEE, Joon Sung

Telephone No. 82-42-481-8544



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2007/088334

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004-0002891 A1	01.01.2004	NONE	
US 2007-0179828 A1	02.08.2007	NONE	
US 7239311 B2	03.07.2007	US 2006-146053 AA	06.07.2006
US 2004-19688 A1	29.01.2004	NONE	